Algorithms for Trust-Region Subproblems with Linear Inequality Constraints

Melanie L. Bain Gratton

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Statistics and Operations Research.

Chapel Hill
2012

Approved by:

Jon W. Tolle
Gabor Pataki
Shu Lu
Scott Provan
Douglas Kelly
Abstract

MELANIE L. BAIN GRATTON: Algorithms for Trust-Region Subproblems with Linear Inequality Constraints. (Under the direction of Jon W. Tolle.)

In the trust-region framework for optimizing a general nonlinear function subject to nonlinear inequality constraints, sequential quadratic programming (SQP) techniques generate subproblems in which a quadratic function must be minimized over a spherical region subject to linear inequality constraints. An interior-point algorithm proposed by Kearsley approximately solves these subproblems when the objective functions are large-scale and convex. Kearsley’s algorithm handles the inequality constraints with a classical log-barrier function, minimizing quadratic models of the log-barrier function for fixed values of the barrier parameter subject to the trust-region constraint. Kearsley recommends the LSTRS algorithm of Rojas et al. for minimizing these models. For the convex case, we prove convergence of Kearsley’s algorithm and suggest alternatives to the LSTRS algorithm. These alternatives include the new annulus algorithm of Griffin et al., which blends the conjugate gradient and sequential subspace minimization methods to yield promising numerical results. For the nonconvex case, we propose and test a new interior-point algorithm that incorporates the annulus algorithm into an SQP framework with trust regions.
To Michael Gratton, who makes everything wonderful
Acknowledgments

I would like to express my gratitude to my advisor, Jon W. Tolle, for taking on just one more graduate student. I could not have made it this far without his support and guidance (and patience – LOTS of patience). I would also like to thank my committee members Gábor Pataki, Shu Lu, Scott Provan, and Douglas Kelly for their feedback and advice. I am very grateful to Joshua Griffin and Wenwen Zhou of SAS Institute, who made great suggestions and always gave generously of their time when I had questions.

I would like to thank my parents Eileen and Robert, my uncle Ron, my Mamaw, and my siblings Timothy, Christina, Ryan, and Kathleen for their love and support.

I would like to thank the UNC Board of Governors, the Statistical and Applied Mathematical Sciences Institute, the Department of Statistics and Operations Research at UNC, and SAS Institute for providing me with employment and learning opportunities. Many thanks to Gehan Corea, Manoj Chari, and Radhika Kulkarni of SAS for helping me to refine my skills and find my confidence.

I would like to thank all the faculty and staff who contributed to my education, especially the professors and staff in the UNC STOR department, for their hard work and dedication.

Finally, I’d like to thank all the amazing friends, extended family, and coworkers who have enriched my life over the years. I am particularly grateful to Aaron Jackson and Kevin Kesseler for helping me through the toughest times.

Thank you all.
# Table of Contents

Abstract ................................................................. iii

List of Figures ........................................................... ix

1 Introduction ............................................................ 1
   1.1 General Problem ................................................... 1
   1.2 Line-search and Trust-region Approaches ....................... 2
   1.3 Sequential Quadratic Programming .............................. 5
       1.3.1 Merit Functions and Filters ............................... 6
   1.4 Penalty Function Methods ...................................... 7
   1.5 Large-scale Problems .......................................... 7
   1.6 Problem of Interest ............................................. 8
   1.7 Outline ............................................................ 9

2 Analysis of Trust-region Subproblems ............................... 10
   2.1 Necessary and Sufficient Conditions ........................... 10
   2.2 Properties of the Parameterized Solution ...................... 13
       2.2.1 Case 1 ..................................................... 18
       2.2.2 Case 2 ..................................................... 18
       2.2.3 Case 3 ..................................................... 19
       2.2.4 Case 4 ..................................................... 22
   2.3 Necessary and Sufficient Conditions for the Inequality-constrained Sub-
       problem .......................................................... 22
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 Positive Definite Case</td>
<td>84</td>
</tr>
<tr>
<td>B.2 Indefinite Case</td>
<td>98</td>
</tr>
<tr>
<td>Bibliography</td>
<td>112</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Line-search and trust-region steps ................................................. 4

2.1 Case 1, second subcase; Case 2 ................................................ 19

2.2 Case 3, second subcase .......................................................... 20

2.3 Case 3, third subcase (the hard case) .......................................... 21

3.1 $\Phi(\lambda)$ with $\|x^N\| > \Delta$, hard case not in effect ...................... 28

3.2 $\Phi(\lambda)$ with $\|x^N\| = \|x(0)\| < \Delta$, hard case not in effect ........... 28

3.3 $\Phi(\lambda)$, hard case in effect ................................................. 29

3.4 Annulus algorithm: restart indicated .......................................... 46

4.1 Proposed algorithm: local trust region centered at $x^k$ ..................... 63

4.2 Flowchart for proposed algorithm .............................................. 70

5.1 Positive definite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.05$ ........... 73

5.2 Positive definite $H$, $\Delta = 1$, $m = n$, density of $A = 0.15$ .............. 74

5.3 Positive definite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.05$ .......... 75

5.4 Indefinite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.05$ ............... 76

5.5 Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.05$ ................. 77

5.6 Indefinite $H$, $\Delta = 10$, $m = n$, density of $A = 0.15$ .................. 78

B.1 Positive definite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.01$ .......... 84

B.2 Positive definite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.05$ .......... 85

B.3 Positive definite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.15$ .......... 85

B.4 Positive definite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.01$ .......... 86

B.5 Positive definite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.05$ .......... 86
B.6 Positive definite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.15$ . . . . . . . . 87
B.7 Positive definite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.01$ . . . . . . . . 87
B.8 Positive definite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.05$ . . . . . . . . 88
B.9 Positive definite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.15$ . . . . . . . . 88
B.10 Positive definite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.01$ . . . . . . . 89
B.11 Positive definite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.05$ . . . . . . . 89
B.12 Positive definite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.15$ . . . . . . . 90
B.13 Positive definite $H$, $\Delta = 1$, $m = n$, density of $A = 0.01$ . . . . . . . . 90
B.14 Positive definite $H$, $\Delta = 1$, $m = n$, density of $A = 0.05$ . . . . . . . . 91
B.15 Positive definite $H$, $\Delta = 1$, $m = n$, density of $A = 0.15$ . . . . . . . . 91
B.16 Positive definite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.01$ . . . . . . . . 92
B.17 Positive definite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.05$ . . . . . . . . 92
B.18 Positive definite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.15$ . . . . . . . . 93
B.19 Positive definite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.01$ . . . . . . . 93
B.20 Positive definite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.05$ . . . . . . . 94
B.21 Positive definite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.15$ . . . . . . . 94
B.22 Positive definite $H$, $\Delta = 10$, $m = n$, density of $A = 0.01$ . . . . . . . . 95
B.23 Positive definite $H$, $\Delta = 10$, $m = n$, density of $A = 0.05$ . . . . . . . . 95
B.24 Positive definite $H$, $\Delta = 10$, $m = n$, density of $A = 0.15$ . . . . . . . . 96
B.25 Positive definite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.01$ . . . . . . . . 96
B.26 Positive definite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.05$ . . . . . . . . 97
B.27 Positive definite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.15$ . . . . . . . . 97
B.28 Indefinite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.01$ . . . . . . . . 98
B.29 Indefinite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.05$ . . . . . . . . 99
B.30 Indefinite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.15$ . . . . . . . . 99
B.31 Indefinite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.01$ . . . . . . . . . . 100
B.32 Indefinite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.05$ . . . . . . . . . . . . . 100
B.33 Indefinite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.15$ . . . . . . . . . . . . . 101
B.34 Indefinite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.01$ . . . . . . . . . . . . . 101
B.35 Indefinite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.05$ . . . . . . . . . . . . . 102
B.36 Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.01$ . . . . . . . . . . . . . 103
B.37 Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.05$ . . . . . . . . . . . . . 103
B.38 Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.15$ . . . . . . . . . . . . . 104
B.39 Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.01$ . . . . . . . . . . . . . 104
B.40 Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.05$ . . . . . . . . . . . . . 105
B.41 Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.15$ . . . . . . . . . . . . . 105
B.42 Indefinite $H$, $\Delta = 1$, $m = n$, density of $A = 0.01$ . . . . . . . . . . . . . . . 106
B.43 Indefinite $H$, $\Delta = 1$, $m = n$, density of $A = 0.05$ . . . . . . . . . . . . . . . 106
B.44 Indefinite $H$, $\Delta = 1$, $m = n$, density of $A = 0.15$ . . . . . . . . . . . . . . . 107
B.45 Indefinite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.01$ . . . . . . . . . . . . . 107
B.46 Indefinite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.05$ . . . . . . . . . . . . . 108
B.47 Indefinite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.15$ . . . . . . . . . . . . . 108
B.48 Indefinite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.01$ . . . . . . . . . . . . . 109
B.49 Indefinite $H$, $\Delta = 10$, $m = n$, density of $A = 0.01$ . . . . . . . . . . . . . . . 109
B.50 Indefinite $H$, $\Delta = 10$, $m = n$, density of $A = 0.05$ . . . . . . . . . . . . . . . 109
B.51 Indefinite $H$, $\Delta = 10$, $m = n$, density of $A = 0.15$ . . . . . . . . . . . . . . . 110
B.52 Indefinite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.01$ . . . . . . . . . . . . . . . 110
B.53 Indefinite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.05$ . . . . . . . . . . . . . . . 111
B.54 Indefinite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.15$ . . . . . . . . . . . . . . . 111
Chapter 1

Introduction

This thesis proposes a new algorithm for minimizing a large-scale quadratic function in a spherical region, subject to linear inequality constraints. Such minimization problems arise frequently in the context of trust-region methods for nonlinear optimization. Kearsley [24] designed an algorithm to solve this minimization problem when the objective function is convex; we present a convergence proof for Kearsley’s algorithm and compare its performance with that of our proposed algorithm in solving positive definite and indefinite minimization problems.

This chapter introduces the problem of interest and some necessary terminology.

1.1 General Problem

We define the general inequality-constrained nonlinear optimization problem by

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to:} & \quad h(x) \leq 0
\end{align*}
\] (1.1)

where \( x \in \mathbb{R}^n \) and the functions \( f : \mathbb{R}^n \to \mathbb{R} \) and \( g : \mathbb{R}^n \to \mathbb{R}^m \) are at least twice continuously differentiable. We will also refer to this problem as a nonlinear programming
problem (NLP).

The Lagrangian function $L(x, \lambda)$ is defined as follows:

$$L(x, \lambda) = f(x) + \lambda^T h(x)$$

(1.2)

The vector $\lambda \in \mathbb{R}^m$ is the vector of Lagrange multipliers for the inequality constraints.

Assume that the point $x^*$ is a local solution of (1.1) at which a constraint qualification [31] is satisfied. Under these assumptions, there exists a vector $\lambda^* \in \mathbb{R}^m$ so that the point $(x^*, \lambda^*)$ satisfies the following equations:

$$\nabla f(x^*) + J h(x^*)^T \lambda^* = 0$$

(1.3)

$$h(x^*) \leq 0$$

(1.4)

$$\lambda^* \odot h(x^*) = 0$$

(1.5)

$$\lambda^* \geq 0$$

(1.6)

In the preceding equations, $J h(x^*)$ is the Jacobian of $h$ evaluated at $x^*$. The $\odot$ symbol indicates componentwise multiplication. These equations are known as the Karush-Kuhn-Tucker (KKT) first-order necessary conditions.

### 1.2 Line-search and Trust-region Approaches

Two basic approaches to solving optimization problems can best be described in terms of the unconstrained problem

$$\min_{x \in \mathbb{R}^n} f(x)$$

(1.7)
A line search algorithm starts from a feasible iterate $x_k$ and identifies a descent direction $\delta_k$. A descent direction $\delta_k$ is a vector that satisfies the condition

$$\nabla f(x_k)^T \delta_k < 0.$$ 

Given a descent direction, the line search algorithm approximately solves the one-dimensional minimization problem

$$\min_{\alpha} f(x_k + \alpha \delta_k)$$

to determine a suitable step length $\alpha_k$. Next, the algorithm performs the update $x_{k+1} = x_k + \alpha_k \delta_k$. This process repeats until the current iterate $x_k$ satisfies some convergence criteria.

One popular choice of descent direction is the negative gradient $\delta_k = -\nabla f(x_k)$. Algorithms employing this choice of $\delta_k$ are called steepest descent algorithms. When the objective function $f(x)$ is convex, another choice of descent direction arises from minimizing a second-order Taylor series approximation of $f(x)$ at the current iterate $x_k$. This yields the Newton direction $\delta_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$. When the function is not convex or when the Hessian $\nabla^2 f(x_k)$ is too expensive to compute at every iteration, many algorithms substitute an easily-updated positive definite approximation $B_k$ to $\nabla^2 f(x_k)$. These algorithms are known as quasi-Newton algorithms. One popular updating scheme, the BFGS method, is a rank-two update that preserves symmetric positive definiteness of $B_k$. It can be implemented in a limited-memory form that is suitable for large-scale problems.

The second general approach to solving (1.7) (and the approach of interest in this thesis) is the trust-region method. This method forms a model $m(\delta)$ of $f(x_k + \delta)$. The model is usually quadratic; common choices are a second-order Taylor series or a
quasi-Newton model. The trust-region method combines the quadratic model with a spherical constraint to form the **trust-region subproblem**

\[
\begin{align*}
\text{minimize} & \quad m(\delta) = \frac{1}{2} \delta^T H_k \delta + g_k^T \delta \\
\text{subject to:} & \quad \delta^T \delta \leq \Delta_k^2
\end{align*}
\]  

(1.8)

The spherical constraint, called the **trust-region constraint**, describes a region centered at the current iterate \(x_k\) in which the quadratic model \(m(\delta)\) is trusted to approximate the objective function \(f(x_k + \delta)\) well. Solving the trust-region subproblem effectively selects both the direction and length of the step \(\delta_k\) at the same time. One advantage of this approach is that neither the Hessian nor its approximation must be positive definite. Figure 1.1 demonstrates a current iterate \(x\) and the contours of the quadratic model, as well as the line-search (LS) and trust-region (TR) steps.

After the solution of (1.8) yields a new iterate \(x_k + \delta_k\), the trust-region method updates the **trust-region radius** \(\Delta_k\) based on a comparison of the decrease in the model \((m(x_k + \delta_k) - m(x_k))\) and the decrease in the objective function \((f(x_k + \delta_k) - f(x_k))\). If the decrease in the model accurately predicts the actual decrease in the objective,
then the trust-region radius can either expand or remain the same for the next iteration. If the decrease in the model poorly predicts the decrease in the objective, the trust-region radius contracts for the next iteration.

1.3 Sequential Quadratic Programming

Efficient algorithms for solving the more general inequality-constrained nonlinear problem (1.1) often require a quadratic model of the problem. Sequential quadratic programming (SQP) algorithms, surveyed in [6], construct a quadratic model of (1.1) at a current approximate solution $x_k$ and corresponding approximate multiplier vector $\lambda_k$. The model includes a second-order Taylor series approximation of the Lagrangian function and first-order approximations of the constraints, as follows:

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \delta_k^T H \delta_k + g^T \delta_k \\
\text{subject to:} & \quad Jh(x)\delta_k + h(x) \leq 0
\end{align*}$$

(1.9)

In this model, $H$ is the exact Hessian (or an approximate Hessian) of the Lagrangian function (1.2), $g$ is the gradient of the Lagrangian function, and $Jh(x)$ is the Jacobian of the inequality constraints. All of these quantities are evaluated at the current point $(x_k, \lambda_k)$.

One can implement the SQP approach in either a line-search framework or a trust-region framework; however, the added constraints make it much more difficult to find a feasible descent direction or to solve the trust-region subproblem. The line-search method directly solves the quadratic model (1.9) for $\delta_k$ and determines a step length $\alpha_k$, which results in the new point $x_{k+1} = x_k + \alpha_k \delta_k$. (The choice of step length $\alpha_k$ ensures that the new point $x_{k+1}$ is acceptable to a merit function or filter.) The line-search method then updates the multiplier vector $\lambda_k$ and reformulates the quadratic
model (1.9) at the new iterate. The entire process repeats until the current iterate passes some convergence test.

The trust-region method incorporates the SQP approach by adding a trust-region constraint to (1.9). If the proposed iterate $x_{k+1} = x_k + \delta_k$ does not provide sufficient decrease as measured by a merit function or filter, the method reduces the trust-region radius $\Delta$ and solves the updated subproblem. If the proposed iterate does provide sufficient decrease, then the method accepts the iterate and generates new approximate multipliers. Then the method formulates a new trust-region subproblem centered at $x_{k+1}$ in which the trust region can either stay the same or increase in size. Again, this process repeats until the iterates satisfy convergence criteria.

1.3.1 Merit Functions and Filters

Merit functions and filters provide a way to evaluate the progress of an algorithm towards feasibility and optimality. A **merit function** generally combines the objective function $f(x)$ with some measure of infeasibility. The infeasibility measure is weighted by a **penalty parameter** whose value emphasizes either satisfaction of the constraints or minimization of the objective. One choice of merit function is the $l_1$ penalty function defined as

$$\phi_1(x; \mu) = f(x) + \mu \sum_{i=1}^{p} [h_i(x)]^+$$

where $[z]^+ = \max\{0,z\}$ and $\mu > 0$. The penalty parameter $\mu$ generally increases as the sequence of iterates $\{x_k\}$ approaches the minimum of $f$. A proposed iterate $x_{k+1}$ is acceptable if it provides sufficient decrease in the merit function.

A **filter** employs the following infeasibility measure:

$$\eta(x) = \sum_{i=1}^{p} [h_i(x)]^+$$
A pair \((f(x_k), \eta(x_k))\) is said to dominate another pair \((f(x_l), \eta(x_l))\) if \(f(x_k) < f(x_l)\) and \(\eta(x_k) \leq \eta(x_l)\) or if \(f(x_k) \leq f(x_l)\) and \(\eta(x_k) < \eta(x_l)\). The filter is a set of pairs \((f(x_k), \eta(x_k))\) such that no pair dominates any other. A proposed iterate \(x_{k+1}\) is acceptable to the filter if the pair \((f(x^+), \eta(x^+))\) is not dominated by any pair in the filter.

### 1.4 Penalty Function Methods

Another approach to solving (1.1) is to use a penalty function to convert the constrained problem into an unconstrained problem. One common penalty function, also known as a log barrier function, has the form

\[
P(x, \tau) = f(x) - \tau \sum_{i=1}^{m} \log(-h_i(x))
\]

where \(\tau > 0\) is the penalty or barrier parameter.

Given a current feasible iterate \(x_k\) and a value \(\tau_k\) of the barrier parameter, the penalty method applies an unconstrained minimization technique to find an approximate solution \(x(\tau_k)\). Then the method decreases \(\tau_k\) and repeats the process. It can be shown that, under certain conditions usually including convexity, the sequence \(\{x(\tau_k)\}\) converges to a solution of (1.1) as \(\tau_k\) decreases to 0 [31]. (In the nonconvex case, it is usually difficult, if not impossible, to prove convergence.) Our proposed algorithm incorporates this approach, which is also called an interior point method.

### 1.5 Large-scale Problems

The term large-scale describes problems in which the number of decision variables or the number of constraints is large. Factoring the large matrices associated with
these problems is computationally expensive when an algorithm must update and re-factor the matrices at each iteration. For this reason, many algorithms avoid matrix factorization as much as possible. These algorithms are called matrix-free. Such algorithms require only matrix-vector multiplications and hence can exploit matrix sparsity. They use iterative methods rather than direct methods to solve the linear systems that arise.

1.6 Problem of Interest

Having defined all the necessary terminology, we define the problem of interest in this paper as follows:

\[
\begin{align*}
\text{minimize} \quad & \frac{1}{2} x^T H x + g^T x \\
\text{subject to:} \quad & Ax \leq b \\
& x^T x \leq \Delta^2
\end{align*}
\] (1.10)

We assume that the matrix \( H \) is large and sparse. We also assume that the matrix \( A \) has full row rank \( m \). Finally, we assume that the feasible region is nonempty and we start with an initial feasible point \( x_0 \). This inequality-constrained trust-region problem can arise, for example, when using a trust-region SQP method to minimize a nonlinear function subject to nonlinear inequality constraints, as well as in specific applications [24].

When a problem does not immediately satisfy our assumptions, there are some remedies available. If the matrix \( A \) does not have full row rank, a preprocessing step can remove redundant constraints. A standard linear programming solver can determine the feasibility of \( Ax \leq b \) and attempt to locate an initial feasible point (though it would
be necessary to approximate the spherical constraint with a 1-norm constraint or ∞-norm constraint). If additional information about the problem structure is available or problem scaling suggests an ellipsoid constraint, one can modify the algorithms presented in this paper by adding preconditioners for the matrix $H$ and a matrix norm (rather than the 2-norm) for the trust-region constraint. In the interest of simplicity, we do not apply preconditioners or scaling matrices in any algorithm descriptions.

1.7 Outline

The remainder of the thesis is organized in the following manner. In Chapter 2, we review the structure and optimality conditions of trust-region subproblems. In Chapter 3, we examine a variety of methods for solving the trust-region subproblem without inequality constraints. In Chapter 4, we investigate Kearsley’s algorithm for solving problem (1.10) in the large-scale convex case and we propose an extension of the algorithm to solve the problem in the large-scale nonconvex case. Chapter 5 contains numerical results that demonstrate the performance of the proposed algorithm in solving a selection of positive definite and indefinite problems of various size. Chapter 6 presents conclusions and future avenues of research.
Chapter 2

Analysis of Trust-region Subproblems

The simplest type of trust-region subproblem is the unconstrained type. For our purposes, the term “unconstrained” indicates that no constraints besides the spherical constraint are present. This yields the following minimization problem over $\mathbb{R}^n$:

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} x^T H x + g^T x \\
\text{subject to:} & \quad \|x\|^2 \leq \Delta^2
\end{align*}$$ (2.1)

Unless otherwise specified, the $\|\cdot\|$ notation indicates the 2-norm.

An appreciation of the various techniques available for solving (2.1) requires an understanding of the problem structure, which is detailed in this section. The proofs follow the analysis in Sorensen [35].

2.1 Necessary and Sufficient Conditions

In this section, we present the optimality conditions for a solution to the unconstrained problem (2.1), proving the necessary and sufficient conditions in two lemmas. For
convenience, we define
\[ \psi(x) = \frac{1}{2} x^T H x + g^T x \]
which is simply the objective function of (2.1).

**Lemma 2.1.1 (Necessary conditions)** If \( x^* \) is a (local) solution to (2.1), then there is a \( \lambda^* \geq 0 \) such that \((x^*, \lambda^*)\) solves the system

\begin{align*}
(H + \lambda^* I)x^* &= -g \\ (2.2a) \\
\lambda^*(x^T x^* - \Delta^2) &= 0 \\ (2.2b) \\
\|x^*\|^2 &\leq \Delta^2 \\ (2.2c)
\end{align*}

and \((H + \lambda^* I)\) is positive semidefinite.

**Proof:** The equations are simply part of the KKT necessary conditions for (2.1), so it remains to show that \((H + \lambda^* I)\) is positive semidefinite. Suppose first that \( x^* \neq 0 \). Since \( x^* \) solves (2.1), it also minimizes \( \psi(x) \) over all \( x \) such that \( \|x\| = \|x^*\| \). Thus \( \psi(x) \geq \psi(x^*) \) for all \( x \) such that \( \|x\| = \|x^*\| \). When combined with (2.2a), this inequality yields the following equation:

\[ -x^T (H + \lambda^* I)x + \frac{1}{2} x^T H x \geq -x^T (H + \lambda^* I)x + \frac{1}{2} x^T H x^* \]

After some algebra, we find that

\[ \frac{1}{2} (x - x^*)^T (H + \lambda^* I)(x - x^*) \geq \frac{\lambda^*}{2} (x^T x - x^* x^T x^*) = 0 \]

for all \( x \) satisfying \( \|x\| = \|x^*\| \). Since \( x^* \neq 0 \), (2.3) implies that \( H + \lambda^* I \) is positive semidefinite.

Now suppose that \( x^* = 0 \). Then \( g = 0 \), since \( (H + \lambda^* I)x^* = -g \). Thus \( x^* = 0 \).
solves \( \min \left\{ \frac{1}{2} x^T H x : \|x\|^2 \leq \Delta^2 \right\} \), so \( H \) is positive semidefinite. By (2.2b), \( \lambda^* = 0 \), so \( H + \lambda^* I \) is positive semidefinite. \( \square \)

**Lemma 2.1.2 (Sufficient conditions)** Let \((x^*, \lambda^*)\) satisfy the necessary conditions of Lemma 2.1.1. Then \( x^* \) is a global solution to (2.1).

**Proof:** If \( \|x^*\| < \Delta \), then \( \lambda^* = 0 \) by (2.2b). Hence \( H \) is positive semidefinite, so \( \psi(x) \) is convex. Thus the necessary conditions are sufficient for \( x^* \) to be a global solution.

If \( \|x^*\| = \Delta \), then since the matrix \( H + \lambda^* I \) is positive semidefinite, the function

\[
\phi(w) = \frac{1}{2} w^T (H + \lambda^* I) w + g^T w
\]

is convex for all \( w \in \mathbb{R}^n \). The function \( \phi(w) \) takes its minimum when

\[
\nabla \phi = (H^T + \lambda^* I) w + g = 0,
\]

so \( x^* \) is a global minimum of \( \phi(w) \). Thus for any \( w \in \mathbb{R}^n \), the following inequality holds:

\[
\frac{1}{2} w^T (H + \lambda^* I) w + g^T w \geq \frac{1}{2} x^*^T (H + \lambda^* I) x^* + g^T x^*
\]

Rearranging terms, we see that

\[
\psi(w) \geq \psi(x^*) + \frac{\lambda^*}{2} (x^*^T x^* - w^T w).
\]

(2.4)

and therefore for any \( w \in \mathbb{R}^n \) satisfying \( \|w\| \leq \|x^*\| = \Delta \), it is the case that \( \psi(w) \geq \psi(x^*) \). Thus \( x^* \) is a global solution to (2.1). \( \square \)
2.2 Properties of the Parameterized Solution

In this section, we parameterize the potential solutions \( x \) of the unconstrained problem by using the Lagrange multiplier \( \lambda \) for the spherical constraint. We prove several properties of these parameterized solutions.

Suppose that the smallest eigenvalue of the matrix \( H \) is \( \lambda_1 \). We define \( \hat{\lambda} = \max(0, -\lambda_1) \). If \( \lambda > \hat{\lambda} \), then \( H + \lambda I \) is positive definite and we can define

\[
x(\lambda) = -(H + \lambda I)^{-1}g
\]  

(2.5)

The following lemmas describe some useful properties of the parameterized curve \( x(\lambda) \).

**Lemma 2.2.1** If \( g \neq 0 \), then \( \|x(\lambda)\| \) strictly decreases to 0 as \( \lambda \) increases to \( \infty \) from \( \hat{\lambda} \).

**Proof:** First, we define

\[
\rho(\lambda) = \|x(\lambda)\|^2 = x(\lambda)^T x(\lambda).
\]  

(2.6)

The derivative of \( \rho(\lambda) \) with respect to the parameter \( \lambda \) is

\[
\frac{d\rho}{d\lambda} = 2x(\lambda)^T \frac{dx}{d\lambda}.
\]  

(2.7)

All that remains is to compute \( \frac{dx}{d\lambda} \) and show that \( \frac{d\rho}{d\lambda} \) is negative for all \( \lambda > \hat{\lambda} \). By definition, it is the case that

\[
(H + \lambda I)x(\lambda) = -g.
\]

Taking the derivative of this expression with respect to \( \lambda \) and rearranging terms, we find that

\[
\frac{dx}{d\lambda} = -(H + \lambda I)^{-1}x(\lambda).
\]  

(2.8)
Substituting the preceding expression into (2.7), we have
\[ \frac{d\rho}{d\lambda} = -2x(\lambda)^T(H + \lambda I)^{-1}x(\lambda). \]

From the positive definiteness of \((H + \lambda I)\), we conclude that \(\frac{d\rho}{d\lambda}\) is negative for all \(\lambda > \hat{\lambda}\). This proves the strict decrease of \(\|x(\lambda)\|\). To see that \(\|x(\lambda)\|\) decreases to 0, we can rewrite \(x(\lambda)\) as follows:
\[ x(\lambda) = -\frac{1}{\lambda} \left( \frac{H}{\lambda} + I \right)^{-1} g. \]

Clearly, as \(\lambda\) approaches \(\infty\), \(\|x(\lambda)\|\) decreases to 0.

**Lemma 2.2.2** \(\psi(\lambda) = \frac{1}{2} x(\lambda)^T H x(\lambda) + g^T x(\lambda)\) is strictly increasing for \(\lambda > \hat{\lambda}\), provided \(g \neq 0\).

**Proof:** Substituting the definition of \(x(\lambda)\) into \(\psi(\lambda)\), we get
\[ \psi(\lambda) = \frac{1}{2} g^T (H + \lambda I)^{-1} H (H + \lambda I)^{-1} g - g^T (H + \lambda I)^{-1} g. \]

Adding and subtracting the term \(\frac{1}{2} g^T (H + \lambda I)^{-2} g\), we can rewrite the preceding equation as follows:
\[ \psi(\lambda) = \frac{1}{2} g^T (H + \lambda I)^{-1} (H + \lambda I) (H + \lambda I)^{-1} g - g^T (H + \lambda I)^{-1} g - \frac{1}{2} g^T (H + \lambda I)^{-2} g \]
\[ = -\frac{1}{2} g^T (H + \lambda I)^{-1} g - \frac{1}{2} g^T (H + \lambda I)^{-2} g \]

Differentiating the preceding expression with respect to \(\lambda\) yields
\[ \frac{d\psi}{d\lambda} = \lambda g^T (H + \lambda I)^{-3} g, \]
which is positive because $H + \lambda I$ is positive definite.

The next lemma concerns the normalized tangent to the curve $x(\lambda)$. Before stating the lemma, we observe that if $T(t)$ is a parameterized $C^1$ curve in $\mathbb{R}^n$, then the normalized tangent to $T$ is $T'(t)/\|T'(t)\|$.

**Lemma 2.2.3** If $g \neq 0$, the normalized tangent to $x(\lambda)$ satisfies

\[
\lim_{\lambda \to \infty} \frac{dx/d\lambda}{\|dx/d\lambda\|} = \frac{g}{\|g\|}.
\]

**Proof:** From (2.8) and (2.5), it is clear that

\[
\frac{dx}{d\lambda} = (H + \lambda I)^{-2}g.
\]

Substituting this into the expression for $dx/d\lambda$, we arrive at

\[
\frac{dx/d\lambda}{\|dx/d\lambda\|} = \frac{(H + \lambda I)^{-2}g}{\sqrt{g^T(H + \lambda I)^{-4}g}}
\]

Finally, we divide by $\lambda$ to get

\[
\frac{(H/\lambda + I)^{-2}g}{\sqrt{g^T(H/\lambda + I)^{-4}g}},
\]

which approaches $g/\|g\|$ as $\lambda$ approaches $\infty$.

As $\lambda$ decreases to $\hat{\lambda}$, the parameterized solution $x(\lambda)$ leaves the origin tangent to the vector $-g$ (Lemma 2.2.3) and moves toward the boundary of the feasible region (Lemma 2.2.1). The following lemma locates the curve relative to the vector $-g$ for the remainder of its path.

**Lemma 2.2.4** For $g \neq 0$ and $\lambda > \hat{\lambda}$, the curve of parameterized solutions $x(\lambda)$ lies in a half-space containing the vector $-g$. This space is bounded by the hyperplane through the origin perpendicular to $-g$. 

15
Proof: For $\lambda > \hat{\lambda}$, the matrix $H + \lambda I$ and its inverse are both positive definite. Using the definition of $x(\lambda)$, we can write the inner product of $-g$ with $x(\lambda)$ as follows:

$$-g^T x(\lambda) = g(H + \lambda I)^{-1} g$$

Thus $-g^T x(\lambda) > 0$, implying that the solution curve lies in the designated half-space.

The following lemma characterizes the solution to (2.1) when this curve leaves the feasible region.

**Lemma 2.2.5** If at some point $\tilde{\lambda} \geq \hat{\lambda}$ we have $\|x(\tilde{\lambda})\| > \Delta$, then there exists a unique $\lambda^* > \tilde{\lambda}$ such that $\|x(\lambda^*)\| = \Delta$ and $x(\lambda^*)$ is a global solution to (2.1).

**Proof:** By Lemma 2.2.1, $\|x(\lambda)\|$ strictly decreases to 0 as $\lambda$ increases from $\tilde{\lambda}$ to $\infty$. Thus there exists a unique point $x(\lambda^*)$ at which $\|x(\lambda^*)\| = \Delta$. The matrix $H + \lambda^* I$ is positive definite since $\lambda^* > \hat{\lambda}$. Lemma 2.1.2 implies that the point $x(\lambda^*)$ is a global solution.

Lemma 2.2.5 states that if we can find a value of $\lambda$ in $(\tilde{\lambda}, \infty)$ for which $x(\lambda)$ is outside the feasible region, then we can find a local solution by tracing the solution curve back to the boundary of the feasible region. We will see later that this fact motivates the solution techniques for (2.1).

As the solution curve moves away from the origin, it might reach $\lambda = \hat{\lambda}$ before crossing the boundary of the feasible region. To characterize the solution in this situation, the eigenvector structure of the parameterized solution must be analyzed.

For the matrix $H$, we choose an orthonormal basis of eigenvectors associated with the eigenvalues of $H$. Let $n_1$ be the multiplicity of the eigenvalue $\lambda_1$, and let $\xi_{ij}, j =
1, \ldots, n_1$, be an orthonormal set of eigenvectors associated with $\lambda_1$. Let $\xi_i$, $i = 2, \ldots, k$, $k = n - n_1$, be eigenvectors for the remaining eigenvalues of $H$. We assume an ordering $\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_k$.

We denote the eigenspace corresponding to the smallest eigenvalue $\lambda_1$ of $H$ by $S_1$:

$$S_1 = \{z : Hz = \lambda_1 z, \quad z \neq 0\}$$

We can now write $H$ as

$$H = \lambda_1 \sum_{l=1}^{n_1} \xi_l \xi_l^T + \sum_{j=2}^{k} \lambda_j \xi_j \xi_j^T$$

and $I$ as

$$I = \sum_{l=1}^{n_1} \xi_l \xi_l^T + \sum_{j=2}^{k} \xi_j \xi_j^T$$

Therefore

$$H + \lambda I = (\lambda_1 + \lambda) \sum_{l=1}^{n_1} \xi_l \xi_l^T + \sum_{j=2}^{k} (\lambda_j + \lambda) \xi_j \xi_j^T.$$ 

The inverse matrix $(H + \lambda I)^{-1}$ exists for all $\lambda \neq -\lambda_1, \ldots, -\lambda_n$ and has the same eigenvectors as $H + \lambda I$, so we can write

$$(H + \lambda I)^{-1} = \frac{1}{(\lambda_1 + \lambda)} \sum_{l=1}^{n_1} \xi_l \xi_l^T + \sum_{j=2}^{k} \frac{1}{(\lambda_j + \lambda)} \xi_j \xi_j^T.$$ 

Finally, we observe that the parameterized solution $x(\lambda) = -(H + \lambda I)^{-1} g$ exists exactly when $(H + \lambda I)^{-1}$ does, in particular for $\lambda > -\lambda_1$. This solution can be expanded as

$$x(\lambda) = -\left( \frac{1}{(\lambda_1 + \lambda)} \sum_{l=1}^{n_1} \xi_l (\xi_l^T g) + \sum_{j=2}^{k} \frac{1}{(\lambda_j + \lambda)} \xi_j (\xi_j^T g) \right).$$

(2.9)
The expanded form of $x(\lambda)$ enables us to analyze the solution under various conditions on the vector $g$ and the eigenspace $S_1$. We consider four cases and characterize a solution in each case.

### 2.2.1 Case 1

In the first case, we assume that the matrix $H$ is positive definite ($\hat{\lambda} = 0, \lambda_1 > 0$). For convenience, we define the **Newton point** $x^N$ as follows:

$$x^N = x(0) = -H^{-1}g$$  \hspace{1cm} (2.10)

Note that if $g = 0$, then the Newton point is the origin. There are two subcases of Case 1. In the first subcase, we assume that $\|x^N\| \leq \Delta$. Since the Newton point is the unconstrained minimizer of a convex quadratic function and is feasible for (2.1), it is the unique minimum.

In the second subcase, we assume that $\|x^N\| > \Delta$. By Lemma 2.2.5, the solution to (2.1) is $x^* = x(\lambda^*)$, where $\lambda^*$ is the unique point for which $\|x(\lambda^*)\| = \Delta$. Figure 2.1 illustrates this scenario.

### 2.2.2 Case 2

In the second case, we assume that $H$ is not positive definite, $g \not\perp S_1$, and $g \neq 0$. Note that $\hat{\lambda} = -\lambda_1 \geq 0$. $x(\lambda)$ as given in (2.9) is still well-defined for $\lambda > \hat{\lambda}$, and

$$\lim_{\lambda \to -\hat{\lambda}^+_1} \|x(\lambda)\| = \infty.$$  

As in the second subcase of Case 1, Lemma 2.2.5 applies. The solution to (2.1) is $x^* = x(\lambda^*)$, where $\lambda^*$ is the unique point for which $\|x(\lambda^*)\| = \Delta$. Figure 2.1 once again serves to illustrate this situation.
2.2.3 Case 3

In the third case, we assume that $H$ is not positive definite, $g \perp S_1$, and $g \neq 0$. This situation is known as a “potential hard case.” We observe that the first term in the eigenvector expansion (2.9) is 0, so

$$x(\lambda) = -\left( \sum_{j=2}^{k} \frac{1}{\lambda_j + \lambda} \xi_j \xi_j^T g \right)$$

A consequence of this formula is that the curve $x(\lambda)$ is orthogonal to the eigenspace $S_1$. In this case, rather than going to infinity, the solution obeys

$$\lim_{\lambda \to -\lambda_1^+} x(\lambda) = x(-\lambda_1) = -\left( \sum_{j=2}^{k} \frac{1}{\lambda_j - \lambda_1} \xi_j \xi_j^T g \right)$$

We also observe that the matrix $H - \lambda I$ is positive semidefinite. To perform the analysis, we divide Case 3 into the following three subcases.

In the first subcase, $x(-\lambda_1) = \Delta$. In this unlikely situation, $x(-\lambda_1)$ is the solution to (2.1), since the sufficient conditions of Lemma 2.1.2 apply. In the second subcase, $x(-\lambda_1) > \Delta$. As in Case 2, Lemma 2.2.5 applies. The solution to (2.1) is $x^* = x(\lambda^*)$, where $\lambda^*$ is the unique point for which $\|x(\lambda^*)\| = \Delta$. An example demonstrating the
second subcase can be seen in Figure 2.2.

In the third subcase, $x(-\lambda_1) < \Delta$, that is, the parameter $\lambda$ reaches $-\lambda_1$ before $x(\lambda)$ escapes the feasible region. The matrix $H + \lambda I$ is indefinite at the boundary of the feasible region; hence, a solution in the form of $x(\lambda) = -(H + \lambda I)^{-1}g$ does not exist. This situation is known as the “hard case,” because we are unable to find a solution simply by tracing the curve $x(\lambda)$ to the boundary. However, after choosing any eigenvector $\xi$ in the eigenspace $S_1$, we can construct a solution to (2.1) by applying the following lemma.

**Lemma 2.2.6** For any $\hat{\tau} \in \mathbb{R}$ and $\xi \in S_1$ such that $\|x(-\lambda_1) + \hat{\tau}\xi\| = \Delta$, a global solution to (2.1) is given by $x^* = x(-\lambda_1) + \hat{\tau}\xi$.

**Proof:** We have already established that the matrix $H - \lambda_1 I$ is positive semidefinite. Next, we consider the KKT conditions that must be satisfied by a local solution $x^*$. First, we multiply the matrix $H - \lambda_1 I$ by $x(-\lambda_1)$, which is the first component of the
Figure 2.3: Case 3, third subcase (the hard case)

The proposed solution $x^*$. 

$$(H - \lambda_1 I)x(-\lambda_1) = -\sum_{j=2}^{k} \frac{\lambda_j}{\lambda_j - \lambda_1} (\xi_j^T g)\xi_j + \sum_{j=2}^{k} \frac{\lambda_1}{\lambda_j - \lambda_1} (\xi_j^T g)\xi_j$$

$$= -\sum_{j=2}^{k} (\xi_j^T g)\xi_j$$

$$= -g \neq 0$$

The last equation holds because we have assumed that $g \perp S_1$. Next, we consider the effect of multiplying the matrix $H - \lambda_1 I$ by $\xi$, as follows:

$$(H - \lambda_1 I)\xi = (\lambda_1 - \lambda_1)\xi = 0$$

Thus $(H - \lambda_1 I)(x(-\lambda_1) + \tau \xi) = -g$ for $\tau \in \mathbb{R}$. Finally, there exist two or more values of $\hat{\tau} > 0$ such that $\|x(-\lambda_1) + \hat{\tau} \xi\| = \Delta$. Thus, $x^* = x(-\lambda_1) + \hat{\tau} \xi$ satisfies the conditions of Lemma 2.1.1, making it a global solution to (2.1).

A diagram that illustrates the hard case can be seen in Figure 2.3.
2.2.4 Case 4

In the fourth and final case, we assume that $H$ is not positive definite and $g = 0$. In this case, $x(\lambda) = 0$ for all values of $\lambda$. We choose any eigenvector $\xi$ of the eigenvalue $\lambda_1$, and we define

$$\hat{\tau} = \frac{\Delta}{\|\xi\|}.$$ 

Then $x^* = \hat{\tau}\xi$ is a solution since $H - \lambda_1 I$ is positive semidefinite, $(H - \lambda_1 I)x^* = 0 = -g$, and $\|x^*\| = \Delta$. Consequently, when $g = 0$, the problem (2.1) is just a problem of finding the minimum eigenvalue of $H$. This relation between solving (2.1) and eigenvalue problems will appear again in Chapters 3 and 4.

Analysis of the preceding four cases demonstrates that we can always obtain a global solution to (2.1), even in the hard case.

2.3 Necessary and Sufficient Conditions for the Inequality-constrained Subproblem

When linear inequality constraints are present, the KKT conditions for the inequality-constrained trust-region subproblem (1.10) are as follows:
\[(H + \lambda^* I)x^* + g - A^T \omega^* - \lambda^* x^* = 0\]

\[\lambda^*(x^*T x^* - \Delta^2) = 0\]

\[\omega^* \odot (Ax^* - b) = 0\]

\[\|x^*\|^2 \leq \Delta^2\]

\[Ax^* - b \leq 0\]

\[\lambda^* \geq 0\]

\[\omega^* \geq 0\]

The proposed algorithm attempts to minimize the sum of the residual norms for the first three conditions while maintaining strict feasibility, complementary slackness, and nonnegative multiplier estimates.
Chapter 3

Algorithms for Trust-region Subproblems

Having investigated the structure of the unconstrained trust-region subproblem (2.1), we now present algorithms for approximating its solution. We begin with an algorithm applicable to the relatively small-scale case, then proceed to review algorithms for large-scale problems.

3.1 Small-scale Case

3.1.1 Positive Definite $H$

Most of the early work in solving trust-region subproblems emphasized the case in which the matrix $H$ is positive definite. Because the underlying general nonlinear problems were small, accuracy was not as important as it was for large-scale problems. The “dogleg,” "double dogleg,” and “hook” methods approximate the intersection of the parameterized solution curve $x(\lambda)$ with the trust region boundary by constructing piecewise linear approximations of $x(\lambda)$ based on the Newton step $x^N = -H^{-1}g$. See [23] for detailed explanations of these methods.
In the large-scale case, the factorization necessary to find a Newton step is inefficient, and it is necessary to solve trust-region subproblems more accurately to avoid an excessive number of iterations in solving the underlying NLP. Moreover, it is generally not easy to determine whether the matrix $H$ is positive definite or indefinite.

In this thesis, we investigate algorithms that can solve the indefinite problem. While our ultimate goal is to solve large-scale problems with inequality constraints, solving a small-scale indefinite problem without inequality constraints is important in our work. As a result, we devote some detail to explaining the method introduced by Moré and Sorensen.

### 3.1.2 Indefinite $H$

When the matrix $H$ is indefinite and small enough to allow factorization, we can solve (2.1) by using a method introduced by Moré and Sorensen [30]. In this section, we again define $x(\lambda)$ as

$$x(\lambda) = -(H + \lambda I)^{-1}g$$

for $\lambda > -\lambda_1$, where $\lambda_1$ is the minimum eigenvalue of $H$. We recall that when $H$ is indefinite, a solution to (2.1) must lie on the boundary of the sphere, with $H + \lambda I$ positive semidefinite.

Rather than solve the equation

$$\|x(\lambda)\| = \Delta$$

(3.1)

Moré and Sorensen’s method first attempts to find an approximate solution to the equation

$$\Phi(\lambda) \equiv \frac{1}{\Delta} - \frac{1}{\|x(\lambda)\|} = 0$$

(3.2)
by using Newton’s method. This choice of equation to solve makes use of the fact that 
\( \|x(\lambda)\|^2 \) can be rewritten as a rational function. To see this, we decompose the real
symmetric matrix \( H \) as follows:

\[
H = BAB^T \quad \text{with} \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) \quad \text{and} \quad B^TB = I.
\]

Then, recalling the derivation in (2.9), we see that

\[
\|x(\lambda)\|^2 = \|B(\Lambda + \lambda I)^{-1}B^Tg\|^2 = \sum_{j=1}^{n} \frac{\gamma_j^2}{(\lambda_j + \lambda)^2}
\]

(3.3)

where \( \gamma_i \) is the \( i \)th component of \( B^Tg \). Because \( \|x(\lambda)\|^2 \) has this structure, the function
\( \Phi(\lambda) \) defined in (3.2) is nearly linear. This allows the the solution of (3.2) to be more
easily obtained by Newton’s method than the solution to (3.1).

Having investigated the structure of \( \|x(\lambda)\| \), we next analyze the behavior of the
function \( \Phi(\lambda) \). The following lemma states the properties of \( \Phi(\lambda) \) that are essential in
Moré and Sorensen’s algorithm.

**Lemma 3.1.1** The function \( \Phi(\lambda) \) is convex and strictly decreasing for
\( \lambda \in (-\lambda_1, \infty) \).

This lemma can be proved by using equations (2.5) – (2.8) and the Cauchy-Schwarz
inequality to show \( \Phi''(\lambda) > 0 \).

Starting with \( \lambda \geq 0 \) (\( H + \lambda I \) positive definite), the most basic form of Moré and
Sorensen’s modified Newton’s method is as follows:

1. Factor \( H + \lambda I = R^TR \) (Cholesky factorization).
2. Solve \( R^TRx(\lambda) = -g \).
3. Solve \( R^Tq = x(\lambda) \).
4. Update $\lambda$:

$$\lambda_+ := \lambda + \left(\frac{\|x(\lambda)\|}{\|q\|}\right)^2 \left(\frac{\|x(\lambda)\| - \Delta}{\Delta}\right)$$  \hspace{1cm} (3.4)

The derivation of the updating step is as follows. Given $R^T R x(\lambda) = -g$ and $R^T q = x(\lambda)$, we can solve for $q$ in terms of $R$ and $g$, which yields

$$q = -R^{-T} (R^T R)^{-1} g$$

Next, using (2.8) and the fact that $R^T R = H + \lambda I$, we observe that we can rewrite $\Phi'(\lambda)$ in terms of the norm of $q$:

$$\Phi'(\lambda) = -\frac{\|q\|^2}{\|x(\lambda)\|^3}$$

Finally, we can write Newton’s method for updating $\lambda$:

$$\lambda_+ = \lambda + \frac{\Phi(\lambda)}{\Phi'(\lambda)}$$

$$= \lambda + \frac{\frac{1}{\Delta} - \frac{1}{\|x(\lambda)\|^2}}{\|q\|^2} \|x(\lambda)\|^3$$

$$= \left(\frac{\|x(\lambda)\|^2}{\|q\|^2}\right) \left(\frac{\|x(\lambda)\| - \Delta}{\Delta}\right)$$

The preceding expression is exactly the updating scheme from Step 4 of the basic algorithm.

The basic algorithm suffers from three major failings; the first of these is that it allows values of $\lambda_+$ less than $\hat{\lambda} = \max(0, -\lambda_1)$. When $\lambda_+$ falls below $-\lambda_1$, the matrix $H + \lambda I$ is indefinite, rendering the Cholesky decomposition impossible. The algorithm must immediately terminate without finding a solution. Even if $\lambda_+ > -\lambda_1$, by allowing negative values of $\lambda_+$, the algorithm becomes slower (in the best case) or misses finding a feasible Newton point (in the worst case).
The second major failing of the algorithm is that it can miss an interior solution. As shown in Figure 3.1, if $H$ is positive definite and the Newton point $x^N$ lies outside the trust region ($\|x^N\| > \Delta$), then the zero of the function $\Phi(\lambda)$ occurs for some positive $\lambda$. In this case, the algorithm should simply locate the corresponding value of $\lambda$. If the Newton point is feasible, as shown in Figure 3.2, then the algorithm should terminate with $\lambda^+ = 0$ rather than find the zero of the function.

The third failing of the algorithm is that it does not include a strategy for dealing with the hard case. In the hard case, as illustrated in Figure 3.3, the function does
not have a zero in the range of $\lambda$ for which $H + \lambda I$ is positive definite. The algorithm should stop with $\lambda$ close to $-\lambda_1$ and use a strategy based on Lemma 2.2.6 to find an approximate solution. The basic algorithm does not include any such strategy. In particular, the algorithm breaks down when $g = 0$, as the formula in Step 4 for updating $\lambda$ contains a division by zero.

To address the first two failings of the basic algorithm, Moré and Sorensen incorporate into their algorithm a safeguarding process. The safeguarding process updates $\lambda$ in such a way that $H + \lambda I$ remains positive definite, which permits Cholesky factorization. The safeguarding process also maintains an interval of decreasing length that attempts to bracket the optimal value of $\lambda$. The success of this process depends on the fact that $\Phi(\lambda)$ is convex and strictly decreasing for values of $\lambda$ between $-\lambda_1$ and $\infty$, as shown in Lemma 3.1.1.

The safeguarding process requires three parameters: $\lambda_L$ and $\lambda_U$, which bracket $\lambda^*$, and $\lambda_S$, which is a lower bound on $-\lambda_1$. $\lambda_S$ is also used to ensure that the length of the interval $[\lambda_L, \lambda_U]$ decreases as the algorithm progresses. The rules used to safeguard $\lambda$ and update (and initialize) $\lambda_L$, $\lambda_U$, and $\lambda_S$ can be found in [30].
To address the third failing of the basic algorithm, we recall that when we are in the hard case, we can still find a solution to (2.1). First, we must compute an eigenvector $\xi \in S_1$. Then, we can define a solution $x(-\lambda_1) + \tau \xi$ based on Lemma 2.2.6. While this approach is theoretically sound, determining that $g \perp S_1$ and computing $x(\lambda)$ and $\xi$ can be resource-intensive. To eliminate the need for eigenvector computations, Moré and Sorensen provide a rather complicated method for approximating the desired eigenvector. However, the methods proposed in Chapter 4 use Moré and Sorensen’s algorithm to solve only very small (four- or five-dimensional) trust-region problems, so it is efficient to approximate the eigenvector $\xi$ by using more conventional methods.

A more complete version of Moré and Sorensen’s algorithm can be written as follows:

1. Safeguard $\lambda$.

2. Attempt to compute the Cholesky factorization of $H + \lambda I$. If the factorization fails, go to 5.

3. Solve $R^T R x = -g$.

4. If $\|x\| < \Delta$, compute $\tau$ and $\xi$.

5. Update $\lambda_L$, $\lambda_U$, and $\lambda_S$.

6. Check the convergence criteria.

7. If $H + \lambda I$ is positive definite and $g \neq 0$ then update $\lambda$ by formula (3.4); otherwise, $\lambda = \lambda_S$.

Moré and Sorensen show that the revised algorithm, when used in conjunction with trust region methods, yields a version of Newton’s method that is robust and effective for solving the trust-region problem.

Having explored algorithms pertinent in the small-scale case, we now turn our attention to solving large-scale versions of (2.1).
3.2 Large-scale Case

In this section, we review techniques for solving (2.1) when \( n \) is very large, rendering matrix factorization too computationally expensive to be useful. We do not assume any knowledge about the eigenvalues of the matrix \( H \).

3.2.1 Conjugate Gradient Methods

The conjugate gradient (CG) method, designed to solve large, sparse, symmetric positive definite linear systems, is the foundation for several methods that approximately solve the trust-region subproblem (2.1). See Algorithm 7 in Appendix A for a step-by-step description of the CG algorithm. When \( H \) is positive definite, the CG method can be applied to the following linear system, which results from setting \( \nabla \psi = 0 \):

\[
Hx = -g \tag{3.5}
\]

The iterates \( \{x_k\} \) generated by the CG method minimize the function \( \psi(x) \) over a sequence of nested Krylov subspaces. The set \( \{p_0, p_1, \ldots, p_k\} \) of CG step directions is \( H \)-conjugate (that is, \( p_j^T H p_k = 0 \) when \( j \neq k \)) and spans the \( k \)th Krylov subspace \( K_k \) defined by

\[
K_k = \text{span} \{g, Hg, H^2g, \ldots, H^k g\}
\]

In exact arithmetic, when \( H \) is positive definite, the CG method converges to the solution of the linear system (which is the unconstrained minimum of (2.1)) in \( n \) iterations. In practical algorithms, CG iterations terminate when the norm of the residual of the linear system falls below a specified tolerance.

If information about the matrix \( H \) is available, it is possible to improve the efficiency of the CG method by applying a preconditioner to reduce the condition number of the
Numerous preconditioners are listed with appropriate references in [25] and [26]. Though the algorithm proposed in this thesis does not incorporate CG preconditioners, it can easily be adapted to perform preconditioned CG iterations. A thorough explanation of the CG method, including preconditioning, can be found in [25].

**Steihaug-Toint**

A simple modification to the conjugate gradient method proposed independently by Steihaug [38] and Toint [40] approximately solves the trust-region subproblem (2.1). This modification relies on the fact that when started from the origin \(x_0 = 0\), CG iterates increase monotonically in norm.

The Steihaug-Toint algorithm applies the CG method to the linear system (3.5), starting at \(x_0 = 0\). At each iteration, the algorithm checks the sign of \(p_i^T H p_i\), where \(p_i\) is the proposed step direction. If \(p_i^T H p_i < 0\), then the matrix \(H\) is indefinite, so the solution to (2.1) must be on the trust-region boundary. The step \(p_i\) is a descent direction for \(\psi(x) = \frac{1}{2}x^T H x + g^T x\), so the algorithm follows this step to the boundary of the trust region. It returns the boundary point as the approximate solution and terminates. If \(p_i^T H p_i > 0\), the algorithm computes a CG step. If the proposed iterate \(x_{i+1}\) would leave the trust region, the method takes the CG step in the direction of \(x_{i+1}\) and stops at the boundary. The method returns this boundary point as the approximate solution. If the proposed iterate stays within the trust region, the method takes the full CG step and iterates again. If the CG iteration solves the linear system to within a user-specified tolerance, the algorithm concludes that the unconstrained minimum (the Newton point \(x^N\)) is feasible with respect to the trust-region constraint. The algorithm returns the final iterate as an approximation of \(x^N\).

The Steihaug-Toint algorithm has some desirable features, but it is not ideal for all cases. The approximate solution is coarse but relatively inexpensive to compute. This
fact suggests that the algorithm can be incorporated into a trust-region framework. Unfortunately, the algorithm does not provide any means to improve the accuracy of boundary solutions. (The accuracy of interior solutions can be improved by decreasing the tolerance for terminating the CG iterations, but it should be noted that CG iterates gradually lose $H$-conjugacy in inexact arithmetic.) The Steihaug-Toint algorithm does not provide any estimate of the multiplier $\lambda^*$ for the trust-region constraint, but it is not difficult to compute an approximation. If the approximate solution is strictly inside the trust region, then $\lambda^* = 0$; otherwise, the least-squares approximation

$$
\lambda_{lsq} = -\frac{\bar{x}^T(H\bar{x} + g)}{\bar{x}^T\bar{x}}
$$

can be used, where $\bar{x}$ is the Steihaug-Toint approximate solution. The accuracy of this multiplier value depends, of course, on the accuracy of $\bar{x}$.

### 3.2.2 Lanczos Methods

The Lanczos method generates bases for the same Krylov subspaces $K_k$ as the conjugate gradient method and provides an alternative method for solving $Hx = -g$ when $H$ is positive definite. See Algorithm 8 in Appendix A.2. Instead of being $H$-conjugate, the set of Lanczos vectors $\{q_0, q_1, \ldots, q_k\}$ is orthonormal (in exact arithmetic). The Lanczos vectors satisfy the following equations:

$$
Q_k^T HQ_k = T_k
$$

$$
Q_k^T g = \gamma_0 e_1
$$

$$
g = \gamma_0 q_0
$$

where $Q_k = [q_0 q_1 \ldots q_k]$, $T_k$ is a symmetric tridiagonal matrix, $\gamma_0 = \|g\|$, $e_1$ is the first canonical unit vector in $\mathbb{R}^n$, and $q_0 = g/\|g\|$. The Lanczos vectors $\{q_k\}$ are
generated iteratively through three-term recursion relations. They are closely related to CG residuals; in fact, they are scalar multiples of the CG residuals, and the entries of the matrix $T_k$ can also be written in terms of the scalars $\alpha_k$ and $\beta_k$ generated in the CG algorithm (see Appendix A.2). The CG and Lanczos iterations break down under different conditions: the CG iteration only fails if it encounters a direction of negative curvature ($\alpha_k < 0$), but the Lanczos iteration breaks down if it encounters an invariant subspace $K_k$. The entries of $T_k$ remain the same from one iteration to the next except for the addition of a new diagonal entry and a new off-diagonal entry.

In practical implementations of the Lanczos algorithm, the vectors $\{q_k\}$ gradually lose orthogonality because the computations are not performed in exact arithmetic. (CG iterates suffer a similar loss of $H$-conjugacy, as we mentioned in the preceding section.) Sorensen [36] proposes an “implicitly restarted” Lanczos method that periodically restores the orthogonality of the vectors at a significant additional computational cost.

Several methods for solving trust-region subproblems are powered by the Lanczos method or an implicitly restarted variant. Two of these, the well-known GLTR algorithm and the LSTRS algorithm, are introduced for the purposes of theoretical comparison with our proposed algorithm. We also incorporate the GLTR algorithm into our numerical tests for practical comparison. Our experience with the LSTRS method suggests that it is not as effective as the GLTR method. For this reason we do not include the LSTRS method in our numerical tests, despite the fact that Kearsley’s method (which we will discuss in Chapter 4) suggests the LSTRS method as a subproblem solver.
GLTR

The generalized Lanczos trust region (GLTR) method of Gould, Lucidi, Toint, and Roma [19] attempts to improve on the Steihaug-Toint strategy for solving the trust-region subproblem (2.1). The GLTR method combines the Lanczos algorithm with a variant of the Moré and Sorensen method described in Section 3.1.2. The authors observe that for a Lanczos basis matrix \( Q_k = [q_0 \ q_1 \ \ldots \ q_k] \), solving the problem in the range space of \( Q_k \), that is,

\[
\min_{x \in \mathbb{R}^n \mid x = Q_k h} \quad \frac{1}{2} x^T H x + g^T x \\
\text{subject to: } \|x\|_2 \leq \Delta
\]

is equivalent to solving the problem

\[
\min_{h \in \mathbb{R}^{k+1}} \quad \frac{1}{2} h^T T_k h + \gamma_0 e_1^T h \\
\text{subject to: } \|h\|_2 \leq \Delta
\]  

(3.7)

where the matrix \( T_k \), the scalar \( \gamma_0 \), and the vector \( e_1 \) are the same as in the Lanczos equations (3.6). Because the matrix \( T_k \) is tridiagonal, the computational cost of factoring \( T_k \) is significantly smaller than the cost of factoring \( H \). This fact makes it feasible to solve the tridiagonal minimization problem (3.7) nearly exactly by using the Moré-Sorensen algorithm. The authors make an alteration to the algorithm so that the starting value of \( \lambda \) lies in the interval \([\max(0, \theta_k), \lambda_k]\), where \( \theta_k \) is the smallest eigenvalue of \( T_k \) and \( \lambda_k \) is the multiplier for the solution \( h_k \) of (3.7).

Algorithm 1 shows a step-by-step walkthrough of the GLTR algorithm. For more information about the algorithm used to solve the tridiagonal trust-region subproblem, see [19]. This reference also provides the preconditioned version of the GLTR algorithm.
Algorithm 1: GLTR algorithm

Require: $H$, $g$, $\Delta$

1: Initialize: $x_0 = 0$, $r_0 = g$, $\gamma_0 = \|g\|$, $p_0 = -g$, $\beta_{-1} = 0$
2: Initialize: INTERIOR = true
3: for $k = 0, 1, \ldots$, until convergence do
4: \hspace{1em} $\alpha_k = r_k^T r_k / p_k^T H p_k$
5: \hspace{1em} Update $T_{k-1}$ (using $\alpha_k$ and $\beta_{k-1}$) to produce $T_k$
6: \hspace{1em} if INTERIOR = true, but $\alpha_k \leq 0$ or $\|x_k + \alpha_k p_k\| \geq \Delta$ then
7: \hspace{2em} Set INTERIOR = false
8: \hspace{1em} end if
9: \hspace{1em} if INTERIOR = true then
10: \hspace{2em} $x_{k+1} = x_k + \alpha_k H p_k$
11: \hspace{1em} else
12: \hspace{3em} Solve tridiagonal subproblem (3.7) to obtain $h_k$
13: \hspace{1em} end if
14: \hspace{1em} $r_{k+1} = r_k + \alpha_k H p_k$
15: \hspace{1em} if INTERIOR = true then
16: \hspace{2em} Test convergence using $\|r_{k+1}\|$
17: \hspace{1em} else
18: \hspace{2em} $\gamma_{k+1} = \|r_{k+1}\|$
19: \hspace{2em} Test convergence using $\gamma_{k+1} |e_{k+1}^T h_k|$
20: \hspace{1em} end if
21: \hspace{1em} $\beta_k = r_{k+1}^T r_{k+1} / r_k^T r_k$
22: \hspace{1em} $p_{k+1} = -r_{k+1} + \beta_k p_k$
23: end for
24: If INTERIOR = false, compute $x_k = Q_k h_k$
Starting from the center of the trust region, the algorithm performs CG steps and builds the matrix $T_k$ by using the CG scalars $\alpha_k$ and $\beta_{k-1}$. After each CG iteration, the algorithm checks to see whether it has encountered either a direction of negative curvature or the boundary of the trust region. If the algorithm has encountered neither, it continues to the next iteration. Otherwise, it concludes that the solution must be on the boundary, so it applies the variant of the Moré-Sorensen algorithm to the tridiagonal trust-region subproblem (3.7) to compute an approximate solution $h_k$. Convergence is indicated when the norm of the current CG residual is smaller than a user-specified tolerance (the solution is interior) or the quantity $\|\nabla L(x_k, \lambda_k)\| = \|(H + \lambda_k I)x_k + g\|$ is smaller than a user-specified tolerance. The authors prove that

$$\|(H + \lambda_k I)x_k + g\| = \gamma_{k+1}|e_{k+1}^T h_k|$$

where $\gamma_{k+1}$ is the norm of the CG residual $r_{k+1}$. This equivalence makes it possible to measure the norm of $\nabla L(x, \lambda)$ without performing the change of variables $x_k = Q_k h_k$, which would cost an additional matrix-vector multiplication at each iteration and require storage of $Q_k$. If $\|\nabla L(x_k, \lambda_k)\|$ is not small enough, the algorithm computes another Lanczos vector $q_{k+1}$ and re-solves the tridiagonal system for another vector $h_{k+1}$. When the algorithm terminates, the multiplier $\lambda_k$ for the solution $h_k$ of the tridiagonal system is returned as an approximate multiplier for the original problem in the full $n$-dimensional space (2.1).

The GLTR algorithm provides some improvement over the Steihaug-Toint strategy and provides an approximate multiplier $\lambda_k$, but the algorithm does suffer from a few drawbacks. When the solution of the $n$-dimensional problem (2.1) is on the trust-region boundary, the matrix $Q_k$ must be available to make the change of variables $x_k = Q_k h_k$. This means that the Lanczos vectors must either be stored (which can be impractical...
for very large $n$) or regenerated using a set of recurrences (which requires more matrix-vector multiplications). The GLTR algorithm is an unrestarted algorithm, which means that the Lanczos vectors are not reorthogonalized at any point. Therefore, after many iterations, the Lanczos matrix $Q_k$ is not orthonormal, which can compromise the approximate solution $x_k$. The authors recommend running the algorithm no more than five to ten iterations past the Steihaug-Toint point. If the resulting $x_k$ does not have a significantly lower objective value than the Steihaug-Toint point, they recommend discarding the additional computation and returning the Steihaug-Toint solution. This saves the added computational requirement of recomputing the matrix $Q_k$.

LSTRS

When $H$ is a large, indefinite, sparse matrix, we can apply the Large-Scale Trust-Region Subproblem (LSTRS) method of Rojas, Santos, and Sorensen [34] to find a solution to (2.1). By adding a new parameter, this method converts the original problem into a scalar problem that is appropriate for the large-scale case.

The method of Rojas, Santos, and Sorensen changes the original problem into an eigenvalue problem involving a parameterized, bordered matrix. They define the bordered matrix

\[ B_\alpha = \begin{pmatrix} \alpha & g^T \\ g & H \end{pmatrix}. \]

and note that

\[ \frac{\alpha}{2} + \psi(x) = \frac{1}{2}(1, x^T)B_\alpha \begin{pmatrix} 1 \\ x \end{pmatrix}. \]

Thus, for any value of our new parameter $\alpha$, we can write (2.1) as
minimize $\frac{1}{2} y^T B_\alpha y$

subject to: $y^T y \leq 1 + \Delta^2$

$e_1^T y = 1$

where $e_1$ is the first canonical unit vector in $\mathbb{R}^{n+1}$. Finding a solution to (3.8) is equivalent to computing a special eigenpair of $B_\alpha$. First, assume that $\{-\lambda, (1, x^T)^T\}$ is an eigenpair of $B_\alpha$. Then it is the case that

$$
\begin{pmatrix}
\alpha \\ g^T \\
H \\
g
\end{pmatrix}
\begin{pmatrix}
1 \\
x
\end{pmatrix} = -
\begin{pmatrix}
1 \\
x
\end{pmatrix} \lambda.
$$

Equivalently, we have

$$
\alpha + \lambda = -g^T x
$$

(3.9)

and

$$(H + \lambda I)x = -g.$$

(3.10)

Using the decomposition in (2.9), we can define the function $\Theta(\lambda)$ for $\lambda \geq 0$ as follows:

$$
\Theta(\lambda) = -g^T x = \sum_{j=1}^n \beta_j^2 / \lambda_j + \lambda
$$

(3.11)

Differentiating with respect to $\lambda$ and recalling the definition of $x(\lambda)$, we obtain

$$
\Theta'(\lambda) = x^T x.
$$

(3.12)

Cauchy’s interlace theorem and (3.11) imply that $\lambda_1(\alpha) \leq \lambda_1$, where $\lambda_1(\alpha)$ is the
smallest eigenvalue of $B_\alpha$. Thus the matrix $H + \lambda_1(\alpha)I$ is positive semidefinite for all values of $\alpha$. If the value of $\alpha$ is updated until the associated $x$ value satisfies $\Theta(\lambda) = \alpha + \lambda$ and $\Theta'(\lambda) = \Delta^2$, then the first order sufficient conditions $(H + \lambda I)x = -g$ and $\lambda(\Delta - \|x\|) = 0$ are satisfied with the matrix $H + \lambda I$ positive semidefinite. If $\lambda \geq 0$, then the solution $x$ lies on the boundary. If we encounter $\lambda < 0$ with $\|x\| < \Delta$, then the Newton point is feasible.

The analysis just described assumes that at least one of the eigenvectors corresponding to the smallest eigenvalue of $B_\alpha$ has a nonzero first component that can be normalized to have the value one. Rojas, Santos, and Sorensen show that in a potential hard case (described in Case 3 in the section “Problem Structure”), for all values of $\alpha$ greater than a “critical value” $\tilde{\alpha}_1$, the eigenvectors of the smallest eigenvalue of $B_\alpha$ all have first component 0. This result does not invalidate the algorithm in a potential hard case; it merely requires that an alternative eigenvector be used. The authors show that given any value of $\alpha$, it is possible to find an eigenvector of $B_\alpha$ whose first component can be normalized to 1. If the potential hard case is not in effect, or if the potential hard case is in effect and $\alpha \leq \tilde{\alpha}_1$, then an eigenvector associated with $\lambda_1(\alpha)$ is used. If the potential hard case is in effect and $\alpha$ is slightly larger than the critical value $\tilde{\alpha}_1$, then an eigenvector corresponding to the second smallest eigenvalue of $B_\alpha$ is used. Since an implicitly restarted Lanczos method can compute the two smallest eigenvalues along with corresponding eigenvectors at relatively low cost (compared to the cost of computing only the smallest eigenpair), the overall algorithm always computes both eigenvalues and eigenvectors whenever a new eigenvector is needed.

Rojas, Santos, and Sorensen’s method uses $\lambda$, $\Theta(\lambda)$, and $\Theta'(\lambda)$ to adjust the parameter $\alpha$ by means of rational interpolation. After finding an interpolant $\hat{\Theta}$, the method locates a point $\hat{\lambda}$ for which $\hat{\Theta}'(\hat{\lambda}) = \Delta^2$. Then they update the parameter $\alpha$ via the
The following equation:
\[ \alpha_+ = \hat{\lambda} + \hat{\Theta}(\hat{\lambda}). \]

As a part of the overall algorithm, the value of \( \alpha_+ \) is safeguarded, which guarantees that the algorithm will converge. The authors prove that their algorithm is globally convergent at a superlinear rate.

The LSTRS method offers advantages that made it Kearsley’s choice of subproblem solver, but it has some drawbacks that caused difficulty in our numerical tests. The algorithm is “matrix-free,” provides an approximate Lagrange multiplier for the trust region, can handle indefinite \( H \), and does not require modification in the hard case. However, the method uses a large number of matrix-vector products for eigenvector computations and can occasionally miss interior solutions in favor of boundary solutions. In fact, our numerical experiments indicate that the default parameters for the LSTRS algorithm emphasize boundary solutions.

**SSM**

The sequential subspace minimization (SSM) technique, which was proposed by Hager [21] and analyzed by Hager and Park [22], iteratively projects a trust-region problem into a smaller subspace and solves the reduced problem with an appropriate small-scale algorithm (such as Moré and Sorensen’s algorithm). After changing variables back to the full space, the solution to the reduced problem is considered an approximate solution to the full problem. Unlike the GLTR method, which optimizes over a sequence of nested subspaces, the SSM method optimizes over a sequence of subspaces of fixed, low dimension. The SSM method is designed to solve the subproblem when the solution is known to be on the boundary, that is, \( \|x\| = \Delta \).
To generate an approximate solution \( x_{k+1} \), the SSM method solves the problem

\[
\begin{align*}
\text{minimize}_{x \in S_k} & \quad \frac{1}{2} x^T H x + g^T x \\
\text{subject to:} & \quad \|x\|_2 = \Delta
\end{align*}
\] (3.13)

where \( S_k \) is the subspace spanned by a small number of vectors (four or five, in practice).

A single iteration of the SSM method is presented in Algorithm 2. The columns of the matrix \( W_k \) span the subspace \( S_k \); \( W_k \) is updated after every SSM iteration. In this algorithm, \((v_{k+1}, \lambda_{k+1})\) is an approximate minimum eigenpair of \( H \). The vector \( x_{k+1} \) is an approximate solution to (3.13) and \( \sigma_{k+1} \) is the corresponding approximate multiplier.

The columns of \( W \) include the previous iterate \( x_k \), which guarantees monotonic decrease of the objective function; the gradient \( H x_k + g \) of the objective function at \( x_k \), which guarantees descent if the previous iterate \( x_k \) did not satisfy the KKT conditions; and an approximate eigenvector \( \xi_1 \) corresponding to the minimum eigenvalue of \( H \), which promotes convergence. This choice of vectors is motivated by the following theorem.

**Theorem 3.2.1 (Hager and Park [22])** *If in each step of SSM \( S_k \) contains the vectors \( x_k, H x_k + g, \) and \( \xi_1 \), an eigenvector associated with the smallest eigenvalue \( \lambda_1 \) of \( H \), then SSM converges to a solution of (3.13).*

Note that in practice the exact minimum eigenpair is not known, so convergence is not guaranteed.
The Moré-Sorensen method is suitable for performing the minimization because the problem is of very small scale (four or five dimensions). The cost of computing an approximate eigenvector $z^*$ is also negligible compared to the cost of one matrix-vector multiplication with the matrix $H$. Because the vector $v_k$ approximates an eigenvector $\xi_1$ for the smallest eigenvalue of $H$, for large problems it is preferable to improve the approximation over several SSM iterations rather than compute a very accurate (and expensive) approximation once before commencing SSM iterations. A change of variables is necessary to transform the SSM trust-region constraint $\|W_k u\|_2 \leq \Delta$ into the form $\|u\|_2 \leq \Delta$ for compatibility with the Moré-Sorensen method. This transformation requires the Cholesky factorization of the matrix $W_k^T W_k$ and the inversion of the Cholesky factor, but the small scale of the problem makes this computation quite inexpensive.

In numerical experiments, Hager observed very slow convergence of the SSM method when he included only the preceding three vectors as a basis for $S_k$. By introducing an appropriately chosen vector to the basis, he was able to accelerate convergence. The original choice of acceleration vector was computed by applying Newton’s method to the nonlinear system

$$(H + \sigma I)x - b = 0, \quad \frac{1}{2} x^T x - \frac{1}{2} \Delta^2 = 0$$

This choice dramatically improves convergence, but it requires at least a partial solution of a linear system at each SSM iteration. Additional research ([14], [15]) has yielded alternative choices for the SSM acceleration vector, but these choices are similarly costly.

The SSM algorithm has several desirable features that can justify its cost. First, the algorithm does not require any assumptions on the eigenvalues of $H$. Second, if the subspace $S_k$ includes a good approximation of the eigenvector $\xi_1$, the SSM iteration
does not have to identify the hard case or take different action when the hard case is in effect. The LSTRS algorithm requires a special modification in the hard case, while the GLTR algorithm disregards the hard case as rare. (We will see later that the hard case can actually be quite common in certain algorithms for solving the inequality-constrained trust-region subproblem.) Finally, unlike the GLTR algorithm, the storage requirement for the SSM algorithm is fixed. As we have seen, however, the SSM algorithm is designed to find boundary solutions; to use this algorithm in general, it must be adapted to locate an interior solution to (2.1) when one exists.

**SSM with Annulus-based Acceleration**

The approach proposed by Griffin et al. [20] interweaves the SSM algorithm with a CG iteration. The method is called SSM with annulus-based acceleration; for convenience, we will call it the “annulus algorithm” in this paper. The annulus algorithm preserves the desirable features of the CG and SSM methods: fast convergence, suitability for large-scale and indefinite problems, effortless handling of the hard case, and fixed storage. Moreover, it identifies an interior solution when one exists.

The most computationally expensive elements of the SSM algorithm are the nearly exact eigenvector $\xi_1$ and the acceleration vector. The annulus algorithm efficiently approximates an eigenvector $\xi_1$ for the smallest eigenvalue of $H$ over successive SSM iterations, which avoids the excessive cost of computing a very accurate initial approximation of $\xi_1$. The annulus algorithm uses the CG steps $\{x_k\}$ and their residuals as the acceleration vectors for the SSM method and restarts the CG step based on feedback from the SSM iteration. This approach inexpensively accelerates the SSM iteration and maintains the conjugacy of the CG step directions.

The annulus algorithm generates two sequences of approximate solutions to the trust-region subproblem (2.1) and an approximate multiplier $\sigma_k$ for the trust-region
constraint. The first sequence of approximate solutions, \( \{\hat{x}_k\} \), arises from a CG iteration (called a “shifted CG iteration”) on the linear system

\[
(H + \hat{\sigma} I)\hat{x} = -g
\]

The second sequence, \( \{x_k\} \), arises from an SSM iteration on the subspace

\[
S_k = \{\hat{x}_k, r_{k-1}, r_k, x_{k-1}, v_{k-1}\}
\]

where \( \hat{x}_k \) is the previous shifted CG iterate, \( r_{k-1} \) and \( r_k \) are the previous and current CG residuals, \( x_{k-1} \) is the previous SSM solution, and \( v_{k-1} \) is the previous approximation of \( \xi_1 \). The algorithm monitors the norms of \( \hat{x}_k \) and \( x_k \); it restarts the shifted CG algorithm from the current SSM iterate whenever the absolute value of the difference between these two norms distance exceeds a specified tolerance. (This behavior gives the algorithm its name - as long as both solution streams remain inside the annular region depicted in Figure 3.4, the shifted CG iteration continues without restarting.) In this framework, the shifted CG iteration provides an inexpensive acceleration step for the SSM iteration. In turn, the SSM iteration provides increasingly accurate approximations of the trust-region multiplier \( \sigma \), so the shifted CG iteration comes closer to solving the first-order optimality condition \( (\nabla L(x, \sigma) = 0) \).

The annulus algorithm is detailed in Algorithm 3. The algorithm computes one \( n \)-by-\( n \) matrix-vector product per iteration; it uses this single product to update both solution streams. The parameter \( \tau_1 \) controls the annulus half-width for restarting CG iterations and the parameter \( \tau_2 \) controls the tolerance for terminating CG iterations. The annulus algorithm starts from the center of the trust region with a multiplier estimate \( \hat{\sigma}_0 = 0 \). In the first iteration, the algorithm takes a CG step to solve the linear system \( Hx = -g \). Before updating the CG iterate, the algorithm adds the
Figure 3.4: Annulus algorithm: restart indicated
Algorithm 3 SSM with annulus-based acceleration (the annulus algorithm)

Require: $H$, $g$, and $\Delta$, $\tau_1, \tau_2 \in (0,1)$.

1: Initialize: $v_0 = 0$, $x_0 = 0$
2: Initialize: $r_0 = -Hx_0 - g = 0$, $\gamma_0 = 0$, $p_0 = 0$
3: $j = 0$, $\hat{\sigma}_j = 0$, $\hat{x}_0 = 0$
4: for $k = 1, \ldots, \text{maxiter}$ do
5: Generate $r_k, \alpha_k, \beta_k, \gamma_k, p_k$ by performing CG update on $(H + \hat{\sigma}_j I, r_k, \gamma_{k-1}, p_{k-1})$
6: Form subspace $S_k = \{\hat{x}_{k-1}, r_{k-1}, r_k, x_{k-1}, v_{k-1}\}$
7: Set $W = \text{basis}(S_k)$
8: Generate $x_k, \sigma_k, v_k, \lambda_k$ by performing an SSM update on $(H, g, W, \Delta)$
9: if $\alpha_k \leq 0$ or $\left|\left|\left| x_k \right|\right| - \left|\left| \hat{x}_k \right|\right| \right| \leq \tau_1 \Delta$ then \{Restart CG\}
10: \hspace{0.5cm} $j = j + 1$
11: \hspace{0.5cm} $\hat{\sigma}_j = \sigma_k$
12: \hspace{0.5cm} $\hat{x}_k = x_k$
13: \hspace{0.5cm} $r_k = -(H + \hat{\sigma}_j I)\hat{x}_k - g$
14: \hspace{0.5cm} $\gamma_{k+1} = 0$
15: else
16: \hspace{0.5cm} $\hat{x}_k = \hat{x}_{k-1} + \alpha_k p_k$
17: \hspace{0.5cm} end if
18: if $\|r_k\| \leq \tau_2 \|g\|$ then
19: \hspace{0.5cm} break
20: end if
21: end for
newly computed CG residual $r_1$ to the subspace $S_1$ and takes an SSM step. The SSM step provides a new SSM iterate $x_1$, an approximate trust-region multiplier $\sigma_1$, and an approximate minimum eigenpair $(\lambda_1, v_1)$.

After the CG and SSM steps are complete, the annulus algorithm checks the restarting conditions for the CG iteration. If the algorithm detects a direction of negative curvature ($\alpha_1 < 0$) or a CG or SSM iterate exits the annular region \[\left( \|x_1\| - \|\hat{x}_1\| > \tau_1 \Delta \right),\]
the algorithm restarts the CG iteration from the SSM iterate $x_1$ by setting $\hat{\sigma}_1 = \sigma_1$ and $\hat{x}_k = x_k$ and resetting the values of $r_1$ and $\gamma_1$ as indicated in Algorithm 3. Otherwise, the algorithm updates the CG iterate $\hat{x}_1$ by using the step length $\alpha_1$ and direction $p_1$ computed at the beginning of the iteration.

The algorithm alternates between CG and SSM steps in this fashion until either the CG residual is small enough in norm ($\|r_k\| \leq \tau_2 \|g\|$) or the specified maximum number of iterations have been reached. It is important to note that the SSM iterations in the annulus algorithm can be disabled until the CG iterates reach the Steihaug-Toint point. Thus the annulus algorithm can find interior solutions to (2.1) and provides an objective value at least as low as the Steihaug-Toint point. Griffin [20] includes a convergence proof for the annulus algorithm.
Chapter 4

Algorithms for

Inequality-constrained Trust-region
Subproblems

4.1 Convex Case: Kearsley

The algorithm of Kearsley [24] finds an approximate solution to the quadratic trust-region model

$$\min_x \frac{1}{2} x^T H x + g^T x$$

s.t. \quad Ax \leq b \quad (4.1)$$

\quad x^T x \leq \Delta^2 $$

with a positive definite $H$ to determine a step for solving the inequality-constrained NLP. Kearsley’s algorithm incorporates the LSTRS algorithm into an interior-point method to handle the inequality constraints. When implementing Kearsley’s algorithm, any of the aforementioned methods for solving large scale trust region problems can be substituted for LSTRS. We investigate this possibility in Chapter 5.
Kearsley’s approach to solving (4.1) is comprised of major and minor iterations. In the major iterations, a classical log-barrier approach moves the inequality constraints (except the trust-region constraint) into the objective function:

\[
\min_x \quad F_\tau(x) = \frac{1}{2} x^T H x + g^T x - \tau \sum_{j=1}^m \ln(b_j - a_j^T x)
\]

s.t. \( x^T x \leq \Delta^2 \) \hspace{1cm} (4.2)

The function \( F_\tau(x) \) is defined for \( \tau > 0 \) and takes the value \( \infty \) for all points outside the set of feasible points

\[
\mathcal{F} = \{ x \in \mathbb{R}^n | Ax < b, x^T x \leq \Delta^2 \}.
\]

The major iterations are generated by a path-following algorithm as follows. Given a sequence \( \{\tau_k\} \) decreasing to 0, each major iterate \( x^k \) is an approximate solution to the actual solution \( x(\tau_k) \), of (4.2) for \( \tau = \tau_k \). As \( \tau_k \to 0 \), the sequence \( \{x(\tau_k)\} \) converges to the unique global minimizer of (4.1). It is obviously not possible to minimize \( F_{\tau_k}(x) \) exactly, but this is not necessary to achieve convergence of the major iterates if the \( x^k \) are adequate approximations of the \( x(\tau_k) \) (see [41]).

Algorithm 4 gives a general representation of the major iterations algorithm.

**Algorithm 4 Kearsley: Major algorithm**

**Require:** \( H, g, A, b, \Delta, x^0 \in \mathcal{F}, \tau_0 \)

1: Parameter: \( M \in (0, 1) \)
2: Initialize: \( k = 0 \)
3: for \( k = 1, 2, \ldots \), until convergence do
4: \( x^{k+1} = \arg \min_{x^T x \leq \Delta^2} F_{\tau_k}(x) \)
5: \( \tau_{k+1} = M \tau_k \)
6: \( k = k + 1 \)
7: end for

The minor iterations are defined as follows. Given an approximate solution \( x^{k-1} \) to
(4.2) for \( \tau = \tau_{k-1} \), a sequence of points \( \{x^{k,j}\} \) is generated by choosing \( x^{k,0} = x^{k-1} \) and

\[
x^{k,j+1} = x^{k,j} + t_{k,j} \delta^{k,j}
\]

where \( \delta^{k,j} \) is an approximate solution to the trust-region problem

\[
\min_x \quad \frac{1}{2} \delta^T \nabla^2 F_{\tau_k}(x^{k,j}) \delta + \nabla F_{\tau_k}(x^{k,j})^T \delta \\
\text{s.t.} \quad \| (x^{k,j} + \delta) \|^2 \leq \Delta^2
\]

and \( t_{k,j} \) is a line search parameter. The vectors \( \{\delta^{k,j}\} \) are obtained by using a large-scale algorithm for the trust-region problem. The choice of \( t_{k,j} \) is made to ensure an appropriate decrease in \( F_{\tau_k} \) and to maintain feasibility. The details are provided in Section 4.1.1.

Algorithm 5 gives a general description of the minor iterations.

**Algorithm 5** Kearsley: Minor algorithm

**Require:** \( x^{k,0} \in \mathcal{F}, \tau_0 \)

1. Parameters: \( \alpha \in (0, 1/2), \beta \in (0, 1), \nu \in (0, 1) \)
2. Initialize: \( j = 0 \)
3. for \( j = 1, 2, \ldots, \) until convergence do
4. Compute search direction \( \delta^{k,j} \) from the solution of (4.3)
5. If \( \delta^{k,j} = 0 \), STOP
6. Determine maximum step length \( t_{\max} \) so that \( A(x^{k,j} + t_{\max} \delta^{k,j}) < b \)
7. if \( t_{\max} < 1 \) then \{Inequality constraint prevents a full step\}
8. Prevent a step that lands directly on the boundary of an inequality constraint:
\[
t_{\max} = \nu t_{\max}
\]
9. end if
10. Determine \( t_{k,j} \) as the first number \( t \) in the sequence \( t_{\max}, \beta t_{\max}, \beta^2 t_{\max}, \ldots \) satisfying the Armijo condition
\[
F_{\tau_k}(x^{k,j} + t \delta^{k,j}) \leq F_{\tau_k}(x^{k,j}) + \alpha t \nabla F_{\tau_k}(x^{k,j})^T \delta^{k,j}
\]
11. \( x^{k,j+1} = x^{k,j} + t_{k,j} \delta^{k,j} \)
12. \( j = j + 1 \)
13. end for
The model (4.3) is approximately solved by applying a large-scale algorithm such as one of the algorithms described in Chapter 3. Kearsley opts to use the LSTRS algorithm for his numerical tests.

With the strong assumption that the matrix $H$ is positive definite, it is possible to prove that the minor iterates $x^{k,j}$ converge to the unique solution $x_k(\tau_k)$ of (4.2). Because one of the proofs in Kearsley’s paper contains an error, we present an alternative analysis of the problem in the following section.

### 4.1.1 Convergence of Minor Iterations

In this section we specify the choice of $t_{k,j}$ for the minor iterations and describe the properties of the resulting minor iterates. We assume that we obtain the $\delta^{k,j}$ by solving the trust region problem (4.3) for a fixed value of $\tau$.

First we note that if the $t_{k,j} \leq 1$, then the iterates $x^{k,j}$ are feasible by virtue of the trust region constraint in (4.3). In order to maintain feasibility with respect to the inequality constraints, we define

$$
\nu_{k,j} = .95 \times \begin{cases} 
1, & \text{if } A\delta^{k,j} \leq 0, \\
\min \left\{ 1, \frac{b_i - a_i^T x^{k,j}}{a_i^T \delta^{k,j}} : \text{s.t. } a_i^T \delta^{k,j} > 0 \right\}, & \text{otherwise.}
\end{cases}
$$

(4.4)

and require $t_{k,j}$ to satisfy

$$
t_{k,j} \|\delta^{k,j}\| \leq \nu_{k,j}
$$

(4.5)

Now we observe that $\delta = 0$ is a feasible solution to (4.3), so the optimal objective value
in (4.3) is nonpositive. Hence
\[
\nabla F_{\tau_k}(x^{k,j})^T \delta^{k,j} = -\frac{1}{2} \left( \delta^{k,j} \right)^T \nabla^2 F_{\tau_k}(x^{k,j}) \delta^{k,j} \leq 0
\]
because \(\nabla^2 F_{\tau_k}\) is positive definite. Thus \(\delta^{k,j}\) is a descent direction for \(F_{\tau_k}\) and we can choose \(t_{k,j}\) to satisfy the Armijo condition
\[
F_{\tau_k}(x^{k,j} + t_{k,j} \delta^{k,j}) \leq F_{\tau_k}(x^{k,j}) + \alpha t_{k,j} \nabla F_{\tau_k}(x^{k,j})^T \delta^{k,j} \tag{4.6}
\]
where \(\alpha\) is a fixed number satisfying \(0 < \alpha < 1/2\).

The convergence analysis for the minor iterations requires the definition of two sets. Define the compact set \(\mathcal{X}_0\) as the set of all points in the trust region.

\[
\mathcal{X}_0 = \{ x \mid \| x \| \leq \Delta \}
\]

Define the open set \(\mathcal{Y}_0\) as the set of points that are strictly feasible with respect to the linear inequality constraints \(Ax \leq b\).

\[
\mathcal{Y}_0 = \{ x \mid Ax < b \}
\]

We assume that the set \(\mathcal{X}_0 \cap \mathcal{Y}_0\) is nonempty. For the remainder of this analysis, we delete the dependence on the superscripts \(k\) and \(j\). That is, \(x\) represents an arbitrary point in \(\mathcal{X}_0 \cap \mathcal{Y}_0\), \(\delta_x\) represents the solution of (4.3) for the point \(x\), and \(t_x\) and \(\nu_x\) are parameter values satisfying (4.4) – (4.6).

With this notation we observe the following properties:

- **Property 1.** For \(x^* \in \mathcal{X}_0 \cap \mathcal{Y}_0\) and \(\varepsilon = \frac{1}{16} \nu_{x^*} > 0\), define \(B_\varepsilon(x^*)\) as a closed ball
of radius $\epsilon$ centered at $x^*$ such that $B_\epsilon(x^*) \cap X_0 \subset Y_0$. Then the set

$$C^* = \bigcap_{x \in B_\epsilon(x^*) \cap X_0} \{ z : z = x + \eta t_x \delta_x, \eta \in [0, 1] \}$$

is a compact subset of $X_0 \cap Y_0$. This is clear because $C^*$ is the intersection of compact subsets of $X_0 \cap Y_0$.

- **Property 2.** There exists a value $\nu_*$ such that $\nu_x \geq \nu_* > 0$ for all $x \in B_\epsilon(x^*) \cap X_0$. This holds because the distance from $x$ to the boundary $\overline{Y_0 \setminus Y_0}$ is a continuous function on the compact set $B_\epsilon(x^*) \cap X_0$.

- **Property 3.** Given $x^*$, there exist positive constants $m_*$ and $M_*$ such that

$$m_* \|\delta\|^2 \leq \delta^T \nabla^2 F_\tau(x + \eta t_x \delta_x) \delta \leq M_* \|\delta\|^2$$

for all $x \in B_\epsilon(x^*) \cap X_0$, all $\delta \in \mathbb{R}^n$, and all $\eta \in [0, 1]$. This statement is true because the set $C^*$ is compact and we have assumed that the function $F_\tau(x)$ is convex and twice continuously differentiable.

- **Property 4.** For $x \in X_0 \cap Y_0$ and $\delta_x \neq 0$, $\nabla F_\tau(x)^T \delta_x < 0$. This holds because $\nabla^2 F_\tau(x)$ is positive definite and $\delta = 0$ is a feasible solution to (4.3). Additionally, by Property 3, we have

$$\nabla F_\tau(x)^T \delta_x \leq -\frac{m_*}{2} \|\delta\|^2$$

for all $x \in B_\epsilon(x^*) \cap X_0$.

For $x^0 \in X_0 \cap Y_0$ (an initial feasible point), fixed $\beta \in (0, 1)$, and fixed $\alpha \in (0, 1/2)$, define a sequence of iterates $\{x^j\}$ as follows:

1. $\delta^j$ is the solution of (4.3).
2. \( t_j = \beta^s \) where \( s \) is the smallest nonnegative integer for which \( t_j \) assures that (4.5) and (4.6) are satisfied.

3. \( x^{j+1} = x^j + t_j \delta^j \).

**Lemma 4.1.1** The sequence of iterates \( \{x^j\} \) is well-defined and the sequence \( \{F_\tau(x^j)\} \) is decreasing. Moreover, any limit point of \( \{x^j\} \) is in \( X_0 \cap Y_0 \).

**Proof:** The definition of (4.3) and the requirement (4.5) on \( t_j \) imply that the sequence of iterates \( \{x^j\} \) remains in \( X_0 \cap Y_0 \), so the sequence is well-defined. The Armijo condition (4.6) guarantees that the sequence \( \{F_\tau(x^j)\} \) is decreasing.

Since the iterates are contained in the closed set \( X_0 \), any limit point \( x^* \) must be contained in \( X_0 \). Now, suppose the iterates converge to a point \( x^* \in \overline{Y_0} \setminus Y_0 \). Then, by the definition of \( F_\tau(x) \), \( \lim_{x^j \to x^*} F_\tau(x) = +\infty \). This contradicts the fact that the sequence \( \{F_\tau(x_j)\} \) is decreasing at each step, so the limit point \( x^* \) must be contained in \( Y_0 \).

**Lemma 4.1.2** The vector \( \delta^j = 0 \) if and only if \( x^j = x^* \), where \( x^* \) is a KKT point for (4.2).

**Proof:** A point \( x^* \in \mathbb{R}^n \) is a KKT point for (4.2) if there exists a multiplier \( \sigma^* \in \mathbb{R} \) satisfying

\[
Hx^* + g + \tau \sum_{l=1} b_l - a_l^T x^* a_l + 2\sigma^* x^* = 0 \tag{4.7}
\]

\[
(x^*)^T x^* \leq \Delta^2
\]

\[
\sigma^* ((x^*)^T x^* - \Delta^2) = 0
\]

\[
\sigma^* \geq 0
\]
A point \( \delta^j \in \mathbb{R}^n \) is a KKT point for (4.3) if there exists \( \sigma_j \in \mathbb{R} \) such that

\[
\nabla^2 F_\tau(x) \delta^j + \nabla F_\tau(x) + 2\sigma_j (\delta^j + x) = 0 \tag{4.8}
\]

\[(\delta^j + x)^T (\delta^j + x) \leq \Delta^2 \]

\[\sigma_j \left( (\delta^j + x)^T (\delta^j + x) - \Delta^2 \right) = 0 \]

\[\sigma_j \geq 0 \]

Now, suppose the vector \( \delta^j = 0 \). Substituting this value into (4.8), letting \( \sigma_j \) be the KKT multiplier, and expanding \( \nabla F_\tau(x^j) \) shows that \( x^j \) satisfies (4.7) with multiplier \( \sigma_j \). The converse follows in a similar manner.

**Lemma 4.1.3** If \( x^* \) is a limit point of \( \{x^j\} \) with convergent subsequence \( j' \in J \), then there exists a positive \( t_* \) independent of \( J \) such that for \( j' \in J \) sufficiently large, (4.5) and (4.6) are satisfied for \( t_{j'} \leq t_* \). Moreover, the backtracking procedure guarantees that \( t_{j'} \) is bounded below.

**Proof:** Define \( J_\epsilon \) as a positive integer such that \( j' > J_\epsilon \) implies that \( x^j' \in B_\epsilon(x^*) \cap X_0 \). Then for \( j' > J_\epsilon \), Taylor’s theorem implies that

\[
F_\tau(x^{j'+1}) - F_\tau(x^j') = t_{j'} \nabla F_\tau(x^j')^T \delta^j' + \left( t_{j'}^2 \right) \left( \frac{\delta^j'}{2} \right)^T \nabla^2 F_\tau(x^j' + t_{j'} \delta^j') \delta^j'.
\]

for some \( \eta \in [0, 1] \). Rewriting the right-hand side yields

\[
F_\tau(x^{j'+1}) - F_\tau(x^j') = \alpha t_{j'} \nabla F_\tau(x^j')^T \delta^j' + (1 - \alpha)t_{j'} \nabla F_\tau(x^j')^T \delta^j' + \left( \frac{t_{j'}^2}{2} \right) \left( \delta^j' \right)^T \nabla^2 F_\tau(x^j' + \eta t_{j'} \delta^j') \delta^j'.
\]

Combining this fact with Properties 3 and 4 and the fact that \( (1 - \alpha) > 0 \), we have that

\[
F_\tau(x^{j'+1}) - F_\tau(x^j') \leq \alpha t_{j'} \nabla F_\tau(x^j')^T \delta^j' - \frac{t_{j'}^2}{2} \{(1 - \alpha)m_* - t_{j'} M_*\}\|\delta^j'\|^2. \tag{4.9}
\]
Define the quantity $t_u$ as follows:

$$t_u = \frac{(1 - \alpha)m_*}{M_*} < 1$$

For $t_{j'} < t_u$, the second term on the right-hand side in (4.9) is negative and can be removed without changing the inequality. Thus all choices of $t_j$ smaller than $t_u$ satisfy the Armijo condition.

Consider the quantity $\nu_*$ defined in Property 2. Because $\nu_* \leq \nu_{j'}$ and $\|\delta^{j'}\| \leq 2\Delta$, we have

$$\frac{\nu_*}{2\Delta} \leq \frac{\nu_{j'}}{\|\delta^{j'}\|}$$

for all $j' > J_\epsilon$. Thus any choice of $t_{j'}$ smaller than $\frac{\nu_*}{2\Delta}$ satisfies (4.5).

Define $t_* = \min\{t_u, \frac{\nu_*}{2\Delta}\}$. Then for $t_{j'} \leq t_*$, (4.5) and (4.6) are satisfied. Let $\beta$ be the factor used for backtracking. If the backtracking line search $t_{j'} = \beta t_*$ reaches the interval $[\beta t_*, t_*]$, (4.5) and (4.6) are satisfied, so the line search will terminate.

Lemma 4.1.4 If the conditions of Lemma 4.1.3 are satisfied then $\delta^{j'} \xrightarrow{j' \in J} 0$.

**Proof:** By Lemma 4.1.3, for all $j' > J_\epsilon$, we have from Property 4 that

$$F_\tau(x^{j'+1}) - F_\tau(x^{j'}) \leq -\alpha \frac{t_{j'}}{2} m_* \|\delta^{j'}\|^2$$

$$\leq -\alpha \beta \frac{t_*}{2} m_* \|\delta^{j'}\|^2$$

Now, assume $\|\delta^{j'}\| > \tilde{\delta} > 0$ for all $j' \in J$. Then

$$F_\tau(x^{j'+1}) - F(x^{j'}) \leq -\alpha \beta \frac{t_*}{2} m_* \tilde{\delta}^2$$

for all $j' > J_\epsilon$. This implies that the sequence $\{F_\tau(x^{j'})\}$ decreases without bound, which contradicts the fact that the iterates $x^{j'}$ are contained within a closed, bounded set. Thus it must be the case that $\delta^{j'} \xrightarrow{j' \in J} 0$.  

57
Lemma 4.1.5 If \( \delta^{j'} \xrightarrow{j' \in \mathcal{J}} 0 \) then the accumulation point \( x^* \) of the sequence \( \{x^{j'}\} \) is the solution to (4.2).

Proof: Denote the multiplier satisfying (4.8) at the solution of (4.3) by \( \sigma_{j'} \). Suppose that the sequence \( \{\sigma_{j'}\} \) is unbounded, and consider the first equation in (4.8). In order to satisfy this equation, we must have \( x^{j'} \xrightarrow{j' \in \mathcal{J}} 0 \). Otherwise, one component of the term \( 2\sigma_{j'}x^{j'} \) would be unbounded while the same component of the other terms would stay constant or approach zero as \( \delta^{j'} \to 0 \). But for \( j' \) sufficiently large, this would contradict the complementary slackness condition of (4.8); thus the sequence \( \{\sigma_{j'}\} \) is bounded. Now, let \( \sigma_* \) be an accumulation point of the sequence \( \{\sigma_{j'}\} \). Taking limits in (4.8) shows that \( x^* \) is a KKT point for (4.2) with multiplier \( \sigma_* \).

Theorem 4.1.6 The sequence \( \{x^{j}\} \) converges to the unique solution \( x^* \).

Proof: This result follows directly from Lemmas 4.1.4 and 4.1.5 and the fact that the solution to (4.2) is unique.

4.1.2 Potential Hard Case

Recall that a potential hard case occurs when \( g \perp S_1 \), where \( S_1 \) is the eigenspace corresponding to the smallest eigenvalue \( \lambda_1 \) of the matrix \( H \). (The hard case occurs when \( g \perp S_1 \) and the value of the parameterized solution curve \( x(\lambda) \) at \( \lambda = \lambda_1 \) is less than \( \Delta \) in norm.) Kearsley presents a non-rigorous explanation why subproblems (4.3) in the minor iterations can tend towards the potential hard case.

The minor iterates \( x^{k,j} \) come from solutions of the quadratic model (4.3) and are kept strictly feasible \( (Ax < b) \) by the effects of the barrier term. Still, as the major iterate index \( k \) increases, the barrier parameter \( \tau_k \) decreases to 0 and iterates can draw close to satisfying some of the inequality constraints at equality. For \( k \) large and fixed, let \( \mathcal{I}_k \subseteq \{1, 2, \ldots, m\} \) be the set of linear constraints that are satisfied at equality by the
minor iterates \( x^{k,j} \) in limit \( (j \to \infty) \). The quadratic and linear terms of the objective in the quadratic model (4.3) can be expressed as follows, where \( a_i^T \) is the \( i \)th row of the matrix \( A \):

\[
\tilde{H} = H + \tau_k \sum_{i=1}^{m} \frac{1}{(b_i - a_i^T x^{k,j})^2} a_i a_i^T \\
\tilde{g} = g + \tau_k \sum_{i=1}^{m} \frac{b_i - 2a_i^T x^{k,j}}{(b_i - a_i^T x^{k,j})^2} a_i
\]

As \( j \) gets large, the penalty terms for \( i \in I_k \) dominate the other terms, so \( \tilde{H} \) approaches

\[
\tilde{H}_k = \sum_{i \in I_k} w_i a_i a_i^T \succeq 0
\]

and \( \tilde{g} \) approaches

\[
\tilde{g}_k = \sum_{i \in I_k} v_i a_i
\]

for some \( v_i, w_i \in \mathbb{R} \). Because \( \text{rank}(\tilde{H}_k) \leq m < n \), 0 is the smallest eigenvalue of \( \tilde{H}_k \). The eigenspace for the positive eigenvalues is \( S = \text{span}(a_i : i \in I_k) \). Because \( \tilde{H}_k \) is symmetric, the eigenvectors for the zero eigenvalue are spanned by the orthogonal complement \( S^\perp \) – that is, \( S_1 = S^\perp \). The vector \( \tilde{g}_k \) is clearly orthogonal to \( S_1 \), which is exactly the condition that defines a potential hard case.

The potential hard case can occur many times in the course of one run of Kearsley’s algorithm. As we will see, the potential hard case can also occur frequently in our proposed algorithm for indefinite \( H \). This fact suggests that a good algorithm for solving the quadratic model (4.3) should handle the hard case without major modification or additional computation.
4.1.3 Observations

While Kearsley’s algorithm has been implemented and tested on a variety of problems, he does not give a proof that it will converge to a global solution to (4.1) when the minor iterates are not computed in exact arithmetic. Moreover, the assumption of strict convexity of the objective function limits the algorithm’s applicability.

Kearsley’s algorithm is not suitable for the nonconvex case, in which the Hessian $H$ of the original quadratic function $\psi(x)$ is indefinite. Kearsley’s convergence proof requires $\nabla F_\tau(x)^T \delta < 0$, where $x + \delta$ minimizes the quadratic approximation to $F_\tau(x)$ on the original trust region. In the indefinite case, regardless of how accurately $\delta$ is computed, there is no guarantee that $\nabla F_\tau(x)^T \delta < 0$. When Kearsley’s algorithm is applied to (2.1) with indefinite $H$, the algorithm frequently generates minor iterates very near the boundary of the original trust region, which can result in very slow progress. Iterates near the trust-region boundary tend to generate short $\delta$ steps; these short steps can cause subsequent iterates to creep along the boundary.

4.2 Nonconvex Case

4.2.1 Proposed Algorithm

We propose a new algorithm to solve the inequality-constrained trust-region subproblem (4.1). The algorithm converts (4.1) into an unconstrained minimization problem by incorporating the set of linear inequality constraints $Ax \leq b$ and the trust-region constraint $x^T x \leq \Delta$ into a log-barrier function. The algorithm utilizes a “local” trust-region constraint with radius $\Delta_{local}$ to perform the minimization in an interior-point SQP-TR framework; it solves trust-region subproblems by calling the annulus algorithm.

The proposed algorithm begins by attempting to solve the inequality-constrained
trust-region problem without considering the inequality constraints at all. The annulus algorithm is applied to the following problem, in which the variable $s$ is used to distinguish the initial iterates from the iterates $x$ in the penalty phase:

\[
\begin{align*}
\text{minimize} \quad & \psi(s) = \frac{1}{2} s^T H s + g^T s \\
\text{subject to:} \quad & s^T s \leq \Delta^2
\end{align*}
\] (4.11)

Starting from a feasible point $s_0$, each iteration of the annulus algorithm generates a CG step $\hat{s}_l$ and an SSM step $s_l$. The annulus algorithm terminates when one of two conditions is met. The first condition is $\|r_l\| \leq \tau_2 \|g\|$ (the CG residual $r_l = H\hat{s}_l + g$ is within a user-specified tolerance), in which case the algorithm terminates and returns the most recent CG iterate $\hat{s}_l$ as the approximate solution to (4.1). This outcome can occur when all or most of the trust region is feasible with respect to the linear inequality constraints. The second (and more common) condition is $As_l \not\leq b$ (the SSM iterate is not feasible with respect to the linear inequality constraints), in which case the algorithm moves into the penalty phase. The penalty phase can start from the original feasible point $s_0$, the most recent feasible SSM iterate $s_{l-1}$, or some other heuristic choice.

It is important to note that the CG steps in the initial run of the annulus algorithm are allowed to be infeasible with respect to all of the constraints (linear equality constraints and trust-region constraint). The CG and SSM steps are the same until the CG steps hit the boundary of the trust region or a direction of negative curvature is detected. After that point, the CG steps might be infeasible, but the SSM steps pull them back to the trust-region boundary and shift the CG iteration by the approximate Lagrange multiplier if necessary. As long as no SSM steps violate the $Ax \leq b$ constraints, the initial annulus iteration continues. Once an SSM step violates the inequality constraints, there’s no way to restore feasibility, so the proposed algorithm
exits the initial annulus phase and begins the penalty phase.

In the penalty phase, the algorithm converts the original inequality-constrained problem (4.1) into an unconstrained minimization problem by forming the log-barrier function $F_\tau(x)$ as follows:

$$
\min F_\tau(x) = \psi(x) - \tau \sum_{i=1}^{m} \log (b_i - a_i^T x) - \tau \log (\Delta^2 - x^T x)
$$

(4.12)

Unlike Kearsley’s log-barrier function, which incorporates only the inequality constraints, (4.12) moves all of the constraints into the objective. To minimize (4.12) for a fixed value of $\tau$, the algorithm uses an SQP approach with a “local” trust region centered at the current iterate $x^{k,j}$ and defined by the radius $\Delta_{local} < \Delta$. At the current iterate $x^{k,j}$, a descent step $\delta^{k,j}$ can be obtained by minimizing the following quadratic model of (4.12) with an explicit trust-region constraint of radius $\Delta_{local}$:

$$
\text{minimize } m(\delta) = \frac{1}{2} \delta^T \nabla^2 F_\tau(x^{k,j}) \delta + \nabla F_\tau(x^{k,j})^T \delta \\
\text{subject to: } \delta^T \delta \leq \Delta_{local}^2
$$

(4.13)

Figure 4.1 depicts the local trust region centered at $x^k$. Once the subproblem (4.13) has been formed, it is approximately solved for $\delta^{k,j}$ by applying the annulus algorithm. The algorithm determines the maximum step length $t_{feas}$ for which $x^{k,j} + t_{feas} \delta$ is feasible with respect to both the inequality constraints and the trust-region constraint. Then it assigns

$$
\delta^{k,j} = \nu t_{feas} \delta^{k,j}
$$

where $\nu$ is a user-specified parameter that prevents numerical difficulties by maintaining strictly feasible iterates. (See Section 4.1.1 for more information about this parameter.)
$t_{k,j} \in \{1, \beta, \beta^2, \ldots\}$ that satisfies the Armijo condition

$$F_{\tau_k}(x^{k,j} + t_{k,j} \delta^{k,j}) \leq F_{\tau_k}(x^{k,j}) + \alpha t_{k,j} \nabla F_{\tau_k}(x^{k,j})^T \delta^{k,j} \quad (4.14)$$

If the line search reaches a user-specified maximum number of backtracking steps without satisfying the Armijo condition, the algorithm does not take a step. Instead, it reduces the trust region $\Delta_{local}$ and re-solves (4.13) for a new step direction $\delta^{k,j}$.

Once the algorithm has computed $t_{k,j}$ so that the proposed step $x^{k,j+1} = x^{k,j} + t_{k,j} \delta^{k,j}$ is feasible and satisfies the Armijo condition, the algorithm computes the ratio $\rho_j$ between the actual reduction $ared_j$ and the predicted reduction $pred_j$ for the proposed step by using the following equation:

$$\rho_j = \frac{ared_j}{pred_j} = \frac{F_{\tau_k}(x^{k,j}) - F_{\tau_k}(x^{k,j} + t_{k,j} \delta^{k,j})}{m_j(0) - m_j(t_{k,j} \delta^{k,j})} \quad (4.15)$$
The algorithm evaluates $\rho_j$ in a standard trust region framework [31] to update the trust region radius $\Delta_{local}$. In our numerical tests, we chose to accept the proposed step whenever $\rho_j > 0$. For $\rho_j \in (0, .25)$, the algorithm performs the update $\Delta_{local} = .25\Delta_{local}$. For $\rho_j > .75$, the algorithm performs the update $\Delta_{local} = 2\Delta_{local}$. For $\rho_j \in [.25, .75]$, the algorithm does not change the value of $\Delta_{local}$.

Finally, the algorithm evaluates criteria to determine whether the new iterate $x^{k,j+1}$ adequately approximates the solution of (4.13) at the current barrier parameter value $\tau_k$. If $x^{k,j+1}$ is a satisfactory approximate solution, the algorithm performs the update $\tau_{k+1} = M\tau_k$, computes $\nabla^2 F_{\tau_{k+1}}(x^{k,j+1})$ and $\nabla F_{\tau_{k+1}}(x^{k,j+1})$, and repeats the process of solving (4.13) for the new barrier parameter value $\tau_{k+1}$. When an iterate satisfies the termination criteria for a solution of the original problem (4.1), the algorithm returns this iterate as the solution.

Algorithm 6 shows a step-by-step conceptual representation of the proposed algorithm; Figure 4.2 depicts the proposed algorithm in flowchart format.

**Implementation Details**

We have described the conceptual version of the proposed algorithm, but it is also necessary to review some aspects of the implementation.

The final value of $\Delta_{local}$ for a fixed value $\tau_k$ can be very small, especially if high accuracy is desired for the solution of (4.12). When the algorithm updates the barrier parameter to $\tau_{k+1}$, having a very small local trust-region radius can prevent the algorithm from taking a significant step. Therefore, the local trust region radius $\Delta_{local}$ is initialized not only at the beginning of the penalty phase but also after each decrease of the barrier parameter $\tau_k$. The proposed algorithm initializes $\Delta_{local}$ as follows.

$$\Delta_{local} = 1.01 \min \left( \Delta^2 - \|x^{k,0}\|^2, \min_i (b_i - a_i^T x^{k,j}) \right)$$

(4.16)
Algorithm 6 Proposed algorithm

**Require:** $H, g, A, b, \Delta, x^0 \in \mathcal{F}$

1: Parameters: $\alpha \in (0, 1/2); \beta, \nu, C, M, \tau_1, \tau_2 \in (0, 1)$
2: Initialize: $l = 0$
3: for $l = 1, 2, \ldots$, until $\|r_l\| \leq \tau_2\|g\|$ or $As_l \notin b$ do
4: Take one step of annulus algorithm on (4.11) to compute shifted CG step $\hat{s}_l$ and SSM step $s_l$
5: end for
6: if Annulus terminated because $\|r_l\| \leq \tau_2\|g\|$ then \{CG converged\}
7: Return $x = \hat{s}_l$ and STOP
8: else
9: Return $x = s_{l-1}$ (or other heuristic choice) and enter penalty phase
10: end if

11: **Penalty phase:** Minimize barrier function (4.12)
12: Initialize: $\tau_0, x^{0,0} = x$
13: Compute $\nabla^2 F_{\tau_0}(x^{0,0}), \nabla F_{\tau_0}(x^{0,0})$
14: for $k = 1, 2, \ldots$, until convergence do
15: Initialize: $\Delta_{\text{local}}, x^{k,1} = x^{k-1,j_{\text{last}}}$
16: for $j = 1, 2, \ldots$, until convergence do
17: Call annulus algorithm to generate step direction $\delta^{k,j}$
18: if $\|\delta^{k,j}\| \leq C\tau_k$ then
19: Break
20: end if
21: Compute maximum feasible step $t_{\text{feas}}$ in direction $\delta^{k,j}$ and set $\delta^{k,j} = t_{\text{feas}}\delta^{k,j}$
22: Determine $t_j$ that satisfies Armijo condition (4.14)
23: Update $\Delta_{\text{local}}$ according to standard trust-region framework in [31]
24: if Line search fails then
25: Break
26: else
27: $x^{k,j+1} = x^{k,j} + t_j\delta^{k,j}$
28: Compute $\nabla^2 F_{\tau_k}(x^{k,j+1}), \nabla F_{\tau_k}(x^{k,j+1})$
29: end if
30: if Termination criteria for minor iterations are met then
31: Break
32: else
33: $j = j + 1$
34: end if
35: end for
36: if Termination criteria for major iterations are met then
37: Break
38: else
39: $\tau_{k+1} = M\tau_k$
40: $k = k + 1$
41: end if
42: end for
This formula provides additional space over which to minimize the quadratic model without allowing steps that are very infeasible or excessively large (in which case the backtracking line search can fail). Although the proposed algorithm can still generate infeasible steps $\delta^{k,j}$, the step-length computations guarantee that the accepted step is strictly feasible with respect to all constraints.

When the annulus algorithm is applied to the quadratic problem (4.13), it generally achieves good reduction in the quadratic objective in the early iterations, but the algorithm often reaches a point at which it will spend many more matrix-vector products without achieving significant additional reduction. This behavior is not desirable, particularly for low values of the major iteration counter $k$ (small $\tau$) and the minor iteration counter $j$; these iterations are merely finding improved starting points for future iterations, so high accuracy is unnecessary. To prevent the annulus algorithm from “stalling” in this manner, we implement an additional termination heuristic for the annulus algorithm. If the difference in objective value between the current and previous annulus iterates is less than $10^{-3}$ in absolute value, the algorithm terminates and returns the current CG iterate. This heuristic preserves significant decrease while preventing excessive matrix-vector products. In numerical tests, we did not observe that this heuristic caused violations of the $\nabla F_{\tau_k}(x^{k,j})^T\delta^{k,j} < 0$ condition, which is required for continuing the minor iterations.

Several different termination criteria govern the major (decreasing $\tau$) and minor (changing $\Delta_{local}$) iterations of the proposed algorithm. In the minor iterations, termination occurs when one of the following conditions is met.

- $\|\delta^{k,j}\| < C \tau_k$: The algorithm cannot take a step of significant size, so the current iterate is considered a reasonable approximation to the subproblem (4.13). The parameter $C \in (0, 1)$ prevents the minor iterations from terminating too early for small values of $\tau_k$. 


• \( \nabla F_{\tau_k}(x^{k,j})^T \delta^{k,j} > \min(-10^{-3}, -\tau) \): The proposed step does not offer significant improvement in the objective function.

• \( \|\nabla F_{\tau_k}(x^{k,j+1})\| < \tau_k \|\nabla F_{\tau_k}(x^{k,0})\| \): The algorithm has sufficiently decreased \( \|\nabla F_{\tau_k}(x)\| \) from its value at the first minor iterate (for the current value of \( \tau_k \)).

• \( \max \left( 10^{-5}, \|\!\|_{x^{k,j+1} - x^{k,j}}\!\| \right) \leq \min \left( \tau_k, 10^{-5}\|x^{k,j}\| \right) \): The relative change in solution size is sufficiently small to suggest that the algorithm cannot make further progress for the current value of \( \tau_k \).

• \( j = \max \text{iter}_j \): The algorithm has reached the maximum number of minor iterations.

If the algorithm encounters one of these conditions, it reduces the barrier parameter, reinitializes \( \Delta_{\text{local}} \), recomputes the Hessian and gradient of the barrier function at \( x^{k+1,0} = x^{k,j+1} \), and begins minor iterations again.

In the major iterations, termination occurs when one of the following conditions is met.

• \( \|\nabla F_{\tau_k}(x^{k,j+1})\| \leq 10^{-5} \) and \( \tau_k \leq 10^{-5} \): The optimality conditions for the unconstrained minimization problem (4.12) are met for a sufficiently small value of the barrier parameter \( \tau \).

• \( \|x^{k,j+1} - x^{k,j}\|/\|x^{k,j}\| < 10^{-5} \): The relative change in solution size is sufficiently small to suggest that the algorithm cannot make further progress.

• \( \|Hx^{k,j+1} + g + A^T \omega + \lambda x^{k,j+1}\| + (m + 1) \tau_k < 10^{-5} \): This formula approximates the sum of the norms of the KKT residuals for the original problem (4.1). It uses approximate multipliers \( \omega_i = \tau_k/(b_i - a_i^T x) \) for \( i = 1, \ldots, m \) and \( \lambda = \tau_k/(\Delta^2 - (x^{k,j+1})^T x^{k,j+1}) \). When this sum is small enough, the current iterate is a reasonable approximation of the solution to (4.1).
• $k = \text{maxiter}_k$: The algorithm has reached the maximum number of major iterations.

If the algorithm encounters one of these conditions, it returns the current iterate $x^{k,j+1}$ as the solution. The algorithm can also return the approximate multipliers that it uses to compute the KKT residuals.

Finally, if the ratio $\rho_j$ defined in (4.15) is greater than one, then the descent direction is excellent. In this case, we attempt to take a longer step with the goal of attaining greater decrease in the objective function. We allow the backtracking line search to start with a step length $t = \rho_j$. Backtracking continues until the algorithm finds a step length that satisfies the criteria of feasibility and sufficient decrease.

**Observations**

The application of the annulus algorithm to the trust-region subproblem (4.11) prior to the penalty phase provides two sources of potential computational savings. The greatest savings can occur when the global minimum of the objective function $\psi(x)$ is strictly feasible with respect to the linear inequality constraints. In this case, the annulus algorithm might solve only one trust-region subproblem (rather than many, which is the case in the penalty phase). An additional source of potential savings is the fact that the initial algorithm takes steps to decrease the objective function $\psi(x)$ before starting the penalty phase. However, if the last feasible SSM iterate $s_{l-1}$ is near a constraint boundary and the penalty phase starts with a high value of $\tau$, the algorithm might take steps away from optimality, then turn around and come back as $\tau$ decreases. This suggests the implementation of some heuristic to determine which feasible SSM iterate (or combination of iterates) to return and how to initialize $\tau$ for the penalty phase. Chapter 6 discusses some possible choices.

The proposed algorithm offers several additional advantages. It has a fixed storage
requirement and handles the hard case without having to detect it or perform special computations. The local trust-region constraint keeps the algorithm from spending a lot of effort on backtracking – in practice, when the minor iterates near the solution of (4.12) for a fixed value of \( \tau \), the algorithm takes full steps for \( j \) large enough. The nearly-exact solution quality offered by the annulus algorithm makes the \( \nabla F_{\tau}(x)^T \delta < 0 \) condition hold more often, which means that computation is not wasted by decreasing \( \Delta_{\text{local}} \) and re-solving (4.13). When the original problem with linear inequality constraints (4.1) is only one of a sequence of trust-region subproblems, the parameters of the proposed algorithm can be relaxed to provide a coarser and less computationally expensive solution. The algorithm can be adapted easily to accommodate preconditioning or non-spherical (ellipsoid) trust-region constraints.

There are some disadvantages associated with the proposed algorithm. Because the annulus algorithm can solve subproblems so accurately, termination criteria (the “stall” criterion and the \( \nabla F^T \delta \) criterion) must be added to prevent over-solving subproblems in the minor iterations. Without these criteria, the algorithm can be quite expensive in terms of matrix-vector products. The optimal settings for various parameters (\( M, C, \nu, \alpha, \beta \), and so on) are not obvious and might be problem-dependent.
Apply annulus algorithm to
\[ \min \psi(s) = \frac{1}{2}s^THs + g^Ts \]
s.t. \[ s^T s \leq \Delta^2 \]
(CG step \( \hat{s}_t \), SSM step \( s_t \))

Yes

CG converged?

No

As \( \leq b \)?

Yes

Return last
CG step
\((x = \hat{s}_t)\),
terminate

Return \( x_{k+1} \),
terminate

Check sign of \( \nabla F_\tau(x_k)^T \delta^* \),
check feasibility of \( x_k + \delta^* \)
w.r.t. \( Ax \leq b \) and \( x^T x \leq \Delta \)

Backtrack for sufficient
decrease (Armijo) \( \rightarrow t \)

Add barrier terms to get
\[ \min F_\tau(x) = \psi(x) - \tau \sum_i (b_i - a_i^T x) - \tau (\Delta^2 - x^T x) \]
initialize \( \tau, \Delta_{\text{local}} \), and \( x_0 \)

No

Compute \( H F_\tau(x_k) \) and \( \nabla F_\tau(x_k) \)

No

Apply annulus algorithm to
\[ \min \frac{1}{2} \delta^T H F_\tau(x_k) \delta + \nabla F_\tau(x_k)^T \delta \]
s.t. \[ \delta^T \delta \leq \Delta_{\text{local}} \rightarrow \text{solution } \delta^* \]

Check term.
criteria

Met

Yes

Update \( \tau = M \tau \),
reset \( \Delta_{\text{local}} \)

\( \| \delta^* \| \leq C \tau ? \)

No

Backtrack succeeded?

Using \( x_k + t \delta^* \), update \( \Delta_{\text{local}} \) via Nocedal

Yes

\( x_{k+1} = x_k + t \delta^* \)
Chapter 5

Numerical Results

We tested the proposed algorithm against Kearsley’s algorithm with various choices of subproblem solver (Steihaug’s method, GLTR, and the annulus algorithm) on randomly generated positive definite and indefinite problems. We varied the dimension of the problems ($n$), the number of constraints ($m$), the sparsity of the constraint matrix $A$, and the trust-region radius $\Delta$; for each combination of values (dimension, constraints, sparsity, $\Delta$), we generated 10 problems by specifying different seeds for random number streams. The parameter settings were $\alpha = .001$, $\beta = .75$, $\nu = .95$, $C = .001$, $M = .5$, $\tau_0 = 1$, and $t_{\text{max}} = 20$ (maximum number of backtracking steps). Intermediate iterations (fixed $\tau$) were limited to 25 and major iterations were limited to 25. (This resulted in a final $\tau$ value of approximately $2.98 \times 10^{-8}$.) Each algorithm was allowed a total of 20 internal iterations in which to solve a subproblem. An algorithm was terminated if the relative difference between successive major iterations fell below $10^{-5}$.

All algorithms were implemented in MATLAB 7.12. Each problem was adjusted to have a nonempty interior and to admit the origin as a feasible starting point. (Infeasibility detection and finding a feasible starting point are outside the scope of this thesis.) For convenience, we refer to Kearsley’s method with the Steihaug subproblem
solver as K-Steihaug. We also refer to K-GLTR and K-annulus. The penalty phase of the proposed algorithm started at \( x = 0 \) in all cases.

We examined three metrics for comparing the algorithms: the total number of Hessian-vector products, the total number of matrix-vector products with the constraint matrix \( A \), and the relative distance from the lowest objective value found. To compute the relative distance for a specified set of parameter values, we first determined the lowest objective value \( \psi_{\text{low}} \) of the four computed solutions. For each algorithm, we subtracted \( \psi_{\text{low}} \) from the algorithm’s final objective value, then divided the result by \( |\psi_{\text{low}}| \). (In this metric, the algorithm that finds the lowest objective value receives a value of 0.) The graphs in this chapter display the mean and standard deviation of each metric for each algorithm over the 10 random instances we solved for each set of parameter values. Rather than review an exhaustive set of results for all parameter combinations, we discuss results for a smaller representative selection. The full numerical results are presented in Appendix B.

5.1 Positive Definite Problems

Our positive definite tests followed the tests in [24]. We took \( H \) to be the discrete, second-order approximation to the two-dimensional Laplacian operator on the unit square. We chose problem sizes \( (n) \) of 36, 64, 128, 256, 512, 1024, and 2048. The number of constraints \( (m) \) varied over \( 0.5n, n, \) and \( 2n \). The elements of the sparse matrix \( A \) were uniformly distributed over \([−10, 10] \); the elements of the vectors \( g \) and \( b \) were uniformly distributed over \([−1, 1] \). The density (number of nonzero elements) of \( A \) took values of 0.01, 0.05, and 0.15.
Figure 5.1: Positive definite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.05$

5.1.1 Observations

Figure 5.1 depicts a typical set of results for the $\Delta = 0.1$ case. For each problem size and each algorithm, the graph at the top displays the number of matrix-vector products that use the constraint matrix $A$. The graph in the lower left corner displays the number of Hessian-vector products. The graph in the lower right corner displays the relative distance from the lowest objective value. We expected the Kearsley-based algorithms to perform reasonably well in the positive definite case, particularly for small trust regions. Small trust regions are less frequently intersected by constraints. This lessens the impact of the log barrier terms, which can introduce roundoff error. The K-Steihaug algorithm provided the coarsest subproblem solutions; this fact was reflected in the large number of Hessian-vector multiplications required to obtain competitive objective values. The K-annulus algorithm provided the lowest objective value, but its high Hessian-vector cost suggested that the algorithm was oversolving subproblems.
The proposed algorithm found the poorest objective values but had the lowest cost in matrix-vector products, which suggested that it could provide a better value if some of the restrictions (like the stall criterion) were relaxed. K-GLTR had the best overall performance.

Figure 5.2 depicts a typical set of results for the $\Delta = 1$ case. In this case, as problem size increased, it was clear that K-Steihaug and K-GLTR were reaching their iteration limits without significantly improving the objective value. K-annulus, which is capable of solving subproblems much more exactly, took many more intermediate iterations (and hence many more Hessian-vector products) without any significant improvement in objective value. The large difference in objective value between the Kearsley algorithms and the new algorithm most probably occurred because the Kearsley algorithms tend to jump immediately to boundary solutions, which limits their ability to take large steps. The Kearsley algorithms must creep along the boundary, even as the barrier parameter
Figure 5.3: Positive definite $H$, $\Delta = 10$, $m/n = 2$, density of $A = 0.05$

$\tau$ decreases. The proposed algorithm, on the other hand, remains in the interior of the trust region and can take larger steps as $\tau$ decreases.

Figure 5.3 depicts a typical set of results for the $\Delta = 10$ case. Again, the Kearsley algorithms became mired on the boundary. The proposed algorithm performed more matrix-vector products than it did for smaller trust regions, but the results were still superior to those obtained by the Kearsley algorithms.

In the positive definite case, the proposed algorithm generally fared better than the Kearsley algorithms as the problem size, trust region radius, number of constraints, and constraint matrix density increased.
5.2 Indefinite Problems

For our indefinite tests, the elements of the sparse matrix $H$ were uniformly distributed over $[-1, 1]$. The sparsity of $H$ was specified to equal the sparsity of a Laplacian matrix of the same dimension. We chose problem sizes ($n$) of 36, 64, 128, 256, 521, and 1024. The number of constraints ($m$) varied over $.5n$, $n$, and $2n$. The elements of the sparse matrix $A$ were uniformly distributed over $[-10, 10]$; the elements of the vectors $g$ and $b$ were uniformly distributed over $[-1, 1]$. The density (number of nonzero elements) of $A$ took values of 0.01, 0.05, and 0.15.

5.2.1 Observations

Figure 5.4 depicts a typical set of results for the $\Delta = 0.1$ case. We did not expect the Kearsley algorithms to perform well in the indefinite case; however, small trust

Figure 5.4: Indefinite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.05$
Figure 5.5: Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.05$

regions likely contain few local minima and thus resemble positive definite problems. Figure 5.4 shows that K-Steihaug and K-annulus performed the most Hessian-vector products (the former because it solves subproblems too coarsely, so it must perform many major iterations to converge; the latter because it solves subproblems too exactly, so it must perform many minor iterations to converge), but the extra work yielded low objective values. The new algorithm was again somewhat hampered by its restrictions against oversolving, but it produced reasonably low objective values for a low cost in Hessian-vector products. As in the positive definite case, K-GLTR performed well for the smallest-sized trust region, but the proposed algorithm outperformed K-GLTR for $\Delta = 0.1$ as the problem size, number of constraints, and constraint matrix density increased.

Figure 5.5 depicts a typical set of results for the $\Delta = 1$ case. K-Steihaug yielded objective values that were competitive with the proposed algorithm, but at a much larger
cost in Hessian-vector multiplications. K-GLTR and K-annulus produced significantly worse objective values for a much higher cost.

Figure 5.6 depicts a typical set of results for the $\Delta = 10$ case. In large trust regions, the proposed algorithm clearly provided the best objective values for the cost in matrix-vector products. This result continued to hold true as the problem size, number of constraints, and constraint matrix density increased.
Chapter 6

Conclusions and Future Work

We have presented an algorithm that minimizes large-scale, nonconvex quadratic functions on a sphere subject to linear inequality constraints. The algorithm incorporates the new annulus algorithm into a framework with standard elements such as log-barrier functions, sequential quadratic programming, and trust regions. By tuning the parameters of the algorithm, it is possible to achieve a very exact solution (or a very coarse solution, if desired). We have presented a new convergence proof for Kearsley’s algorithm in the positive definite case and investigated alternatives to LSTRS (Steihaug-Toint, GLTR, and annulus) for generating the minor iterates. We have compared the performance of the proposed algorithm with Kearsley’s algorithm in positive definite and indefinite scenarios; we contend that the added expense of the proposed algorithm is entirely justified by its robustness in the convex case and the ability to control the quality of the solution.

6.1 Future Work

There are a number of features and heuristics that could improve the performance of the proposed algorithm. For example, as previously mentioned, preconditioning the Hessian matrices to cluster eigenvalues could speed the convergence of the CG steps.
We could also adapt the algorithm to address problem scaling issues by optimizing over a general ellipsoid rather than a sphere.

For simplicity, we designed the numerical tests so that they always started at a feasible point. The proposed algorithm requires a strictly feasible initial point; locating such a point is usually a nontrivial matter. We could apply the “big-M” method for linear programming to make the starting point feasible with respect to the linear inequality constraints (or determine that the feasible region for those constraints is empty). That still leaves the problem of making the initial point feasible with respect to the trust-region constraint with radius $\Delta$. We might resolve this issue by substituting the vector 1-norm for the 2-norm in the trust-region constraint and then adding the constraint to the big-M linear program.

For the numerical tests in this thesis, we generated the constraint matrix at random. It would be interesting to investigate the case in which the linear inequality constraints are all simple bound constraints ($l_i \leq x_i \leq u_i$). In this case, there might be a more efficient way than the log-barrier function to handle the constraints.

The practical implementation of the proposed algorithm involves a number of heuristics. For example, if the initial annulus solve (before the penalty phase) encounters an infeasible SSM step, it resets the iterate to the starting point and terminates. By monitoring the value of the penalty function for different values of $\tau$ at each SSM step in the initial annulus solve, it might be possible to determine a better starting point for the penalty phase as well as a better initial value of $\tau$. We could implement such a heuristic with no additional cost in matrix-vector products. As another example, it might be useful to “warm-start” the annulus algorithm with the last minor iterate $(x^{k,j}, \sigma_{k,j})$ for $\tau_{k+1}$. Finally, it is possible that selecting different penalty parameters for the inequality constraints and the trust-region constraints (or decreasing them at different rates) might improve the performance of the proposed algorithm.
Appendix A

Details of Basic Algorithms

A.1 Conjugate Gradient Algorithm

Algorithm 7 describes the steps of the conjugate gradient (CG) algorithm. This algorithm eventually breaks down if $H$ is not positive definite.

\begin{algorithm}
\caption{Conjugate gradient iteration}
\begin{algorithmic}[1]
\Require $x_0$
\State $r_0 = -Hx_0 - g$, $p_0 = r_0$, $k = 0$
\For{$k = 0, 1, 2, \ldots$, until convergence}
\State $\alpha_k = \frac{r_k^T r_k}{p_k^T H p_k}$
\State $x_{k+1} = x_k + \alpha_k p_k$
\State $r_{k+1} = r_k + \alpha_k H p_k$
\State $\beta_{k+1} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$
\State $p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$
\EndFor
\end{algorithmic}
\end{algorithm}

In practical implementations, the CG algorithm converges when the norm of the gradient $r_k$ falls below a small user-specified tolerance. It is worth noting that the CG iterates lose $H$-conjugacy at the same rate as the Lanczos iterates lose orthonormality.

A.2 Lanczos Algorithm

Algorithm 8 describes the steps of the Lanczos algorithm. This algorithm generates a sequence of orthonormal vectors $\{q_k\}$ that span the Krylov subspace $\mathcal{K}_k$. Although in practice the Lanczos vectors are not based on CG iterations, the Lanczos vectors can
be computed from the CG residuals (provided the CG iteration has not broken down) by the equation

\[ q_k = r_k / \| r_k \| \]

The Lanczos iteration terminates if a solution is obtained or if the Krylov subspace \( K_k \) spanned by \( \{q_0, \ldots, q_k\} \) is an invariant subspace for \( H \).

**Algorithm 8** Lanczos iteration

1: Initialize: \( t_0 = g, w_{-1} = 0, k = 0 \)
2: for \( k = 0, 1, 2, \ldots \), until convergence do
3: \( \gamma_k = \| t_k \| \)
4: \( q_k = t_k / \gamma_k \)
5: \( \delta_k = q_k^T H q_k \)
6: \( t_{k+1} = H q_k - \delta_k w_k - \gamma_k w_{k-1} \)
7: end for

Like the CG algorithm, the Lanczos algorithm is terminated when the norm of the gradient \( r_{k+1} \) falls below a small user-specified tolerance. This norm can be computed by using the equation

\[ \| r_{k+1} \| = \gamma_{k+1} | e_{k+1}^T h_k | \]

where the vector \( h_k \) solves the linear system

\[ T_k h_k + \gamma_0 e_1 = 0 \]

for the tridiagonal matrix \( T_k \) defined as follows.

\[
T_k = \begin{pmatrix}
\| q_0 \| & \gamma_1 & & \\
\gamma_1 & \| q_1 \| & \ddots & \\
& \ddots & \ddots & \ddots \\
& & \| q_{k-1} \| & \gamma_k \\
\gamma_k & & \| q_k \| & \\
\end{pmatrix}
\]
The matrix $T_k$ can also be written in terms of the CG scalars $\alpha_k$ and $\beta_k$ as follows.

$$T_k = \begin{pmatrix}
\frac{1}{\alpha_0} & -\sqrt{\beta_0} \\
-\sqrt{\beta_0} & \frac{1}{\alpha_1} + \frac{\beta_0}{\alpha_0} & -\sqrt{\beta_1} \\
& \ddots & \ddots & \ddots \\
& & & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-2}}{\alpha_{k-2}} & -\sqrt{\beta_{k-1}} \\
& & & -\sqrt{\beta_{k-1}} & \frac{1}{\alpha_k} + \frac{\beta_{k-1}}{\alpha_{k-1}}
\end{pmatrix}$$
Appendix B

Full Results of Numerical Tests

This chapter presents the results of our numerical tests for all four algorithms (K-Steiaug, K-GLTR, K-annulus, proposed algorithm), averaged over 10 different instances. The problem descriptions, lists of parameter values, and algorithm settings appears in Chapter 5.

B.1 Positive Definite Case

![Graph showing the performance of different algorithms for positive definite case.]

Figure B.1: Positive definite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.01$
Figure B.2: Positive definite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.05$

Figure B.3: Positive definite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.15$
Figure B.4: Positive definite $H$, $\Delta = 0.1$, $m/n = 1$, density of $A = 0.01$

Figure B.5: Positive definite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.05$
Figure B.6: Positive definite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.15$

Figure B.7: Positive definite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.01$
Problem size
Constraint matrix/vector multiplications

Positive definite case: $\Delta = 0.1$, $m/n = 2$, dens$(A) = 0.05$

Problem size
Hessian matrix/vector multiplications

Relative objective value

$K^{-GLTR}$
$K^{-Annulus}$
$K^{-Steihaug}$
New alg

Figure B.8: Positive definite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.05$

Problem size
Constraint matrix/vector multiplications

Positive definite case: $\Delta = 0.1$, $m/n = 2$, dens$(A) = 0.15$

Problem size
Hessian matrix/vector multiplications

Relative objective value

$K^{-GLTR}$
$K^{-Annulus}$
$K^{-Steihaug}$
New alg

Figure B.9: Positive definite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.15$
Figure B.10: Positive definite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.01$

Figure B.11: Positive definite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.05$
Figure B.12: Positive definite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.15$

Figure B.13: Positive definite $H$, $\Delta = 1$, $m = n$, density of $A = 0.01$
Positive definite case: $\Delta = 1$, $m/n = 1$, dens($A$) = 0.05

Figure B.14: Positive definite $H$, $\Delta = 1$, $m = n$, density of $A = 0.05$

Positive definite case: $\Delta = 1$, $m/n = 1$, dens($A$) = 0.15

Figure B.15: Positive definite $H$, $\Delta = 1$, $m = n$, density of $A = 0.15
Figure B.16: Positive definite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.01$

Figure B.17: Positive definite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.05$
Figure B.18: Positive definite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.15$

Figure B.19: Positive definite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.01$
Figure B.20: Positive definite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.05$

Figure B.21: Positive definite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.15$
Figure B.22: Positive definite $H$, $\Delta = 10$, $m = n$, density of $A = 0.01$

Figure B.23: Positive definite $H$, $\Delta = 10$, $m = n$, density of $A = 0.05$
Figure B.24: Positive definite \( H, \Delta = 10, m = n \), density of \( A = 0.15 \)

Figure B.25: Positive definite \( H, \Delta = 10, m = 2n \), density of \( A = 0.01 \)
Figure B.26: Positive definite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.05$

Figure B.27: Positive definite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.15$
B.2 Indefinite Case

![Graphs showing matrix multiplications and relative objective values for indefinite case with problem size and density of matrix A.]  

Figure B.28: Indefinite $H$, $\Delta = 0.1$, $m/n = 0.5$, density of $A = 0.01$
Figure B.29: Indefinite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.05$

Figure B.30: Indefinite $H$, $\Delta = 0.1$, $m = 0.5n$, density of $A = 0.15$
Figure B.31: Indefinite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.01$

Figure B.32: Indefinite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.05$
Figure B.33: Indefinite $H$, $\Delta = 0.1$, $m = n$, density of $A = 0.15$

Figure B.34: Indefinite $H$, $\Delta = 0.1$, $m = 2n$, density of $A = 0.01$
Figure B.35: Indefinite $H, \Delta = 0.1, m = 2n$, density of $A = 0.05$

Figure B.36: Indefinite $H, \Delta = 0.1, m = 2n$, density of $A = 0.15$
Figure B.37: Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.01$

Figure B.38: Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.05$
Figure B.39: Indefinite $H$, $\Delta = 1$, $m = 0.5n$, density of $A = 0.15$

Figure B.40: Indefinite $H$, $\Delta = 1$, $m = n$, density of $A = 0.01$
Figure B.41: Indefinite $H$, $\Delta = 1$, $m = n$, density of $A = 0.05$

Figure B.42: Indefinite $H$, $\Delta = 1$, $m = n$, density of $A = 0.15$
Figure B.43: Indefinite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.01$

Figure B.44: Indefinite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.05$
Figure B.45: Indefinite $H$, $\Delta = 1$, $m = 2n$, density of $A = 0.15$

Figure B.46: Indefinite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.01$
Indefinite case: $\Delta = 10$, $m/n = 0.5$, $\text{dens}(A) = 0.05$

Figure B.47: Indefinite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.05$

Indefinite case: $\Delta = 10$, $m/n = 0.5$, $\text{dens}(A) = 0.15$

Figure B.48: Indefinite $H$, $\Delta = 10$, $m = 0.5n$, density of $A = 0.15$
Figure B.49: Indefinite $H$, $\Delta = 10$, $m = n$, density of $A = 0.01$

Figure B.50: Indefinite $H$, $\Delta = 10$, $m = n$, density of $A = 0.05$
Figure B.51: Indefinite $H$, $\Delta = 10$, $m = n$, density of $A = 0.15$

Figure B.52: Indefinite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.01$
Figure B.53: Indefinite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.05$

Figure B.54: Indefinite $H$, $\Delta = 10$, $m = 2n$, density of $A = 0.15


