NORTHWESTERN UNIVERSITY

# LARGE-SCALE NONLINEAR CONSTRAINED OPTIMIZATION USING TRUST REGIONS

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the Degree

DOCTOR OF PHILOSOPHY

Field of Electrical Engineering

By

Todd Dennis Plantenga

EVANSTON, ILLINOIS

December 1994

## ABSTRACT

Large-scale Nonlinear Constrained Optimization Using Trust Regions

Todd Dennis Plantenga

Ph.D. advisor: Jorge Nocedal

This dissertation describes the theory, software implementation and numerical testing of an algorithm for solving smooth nonlinear constrained optimization problems. The algorithm is designed to efficiently solve large sparse problems of unspecified structure. The use of trust region methods provides greater robustness and a different type of behavior than algorithms that instead use line search methods.

The first chapter concentrates on problems with nonlinear equality constraints. An appropriate form of Newton's method is derived, and its local convergence properties proved. Various trust region techniques for globalizing Newton's method are surveyed, and the method of Byrd and Omojokun chosen as an algorithm likely to be efficient for solving large equality constrained problems. A comprehensive study of techniques for solving unconstrained trust region subproblems is made, and the results applied to subproblems that occur in the Byrd-Omojokun algorithm. The final software implementation includes direct and iterative solvers for subproblems, and a novel "deconditioned" form of Steihaug's method appropriate for ellipsoidal trust regions. Results of numerical experiments on difficult test problems are presented and analyzed. Finally, global convergence of an abstract version of the algorithm is proved, then compared to the actual implementation in order to understand tradeoffs made for the sake of improving algorithm performance.

The second chapter formulates a new algorithm for solving nonlinear problems subject to equality and inequality constraints. The method solves a sequence of subproblems containing only equality constraints, using the software developed in the first chapter. It is derived as a generalization of primal interior point methods for linear programming problems. Preliminary test results for large nonlinear nonconvex problems are presented.

ACKNOWLEDGEMENT

I had never heard of numerical optimization when I began studying for my PhD three years ago at Northwestern. I am grateful to Jorge Nocedal for not only introducing me to the subject, but also undertaking the task of being my head advisor. I thank him for his generous instruction to such a novice, timely advice and assistance towards my post-doctoral career, and his continued tolerance for an impatient student more accustomed to a corporate than academic environment.

I am grateful to the other members of my committee, Alvin Bayliss and Eric Schwabe, for their assistance, and I thank Richard Byrd for freely discussing with me many aspects of this work.

I also give my love and thanks to Eileen, my wife of three years, who for my sake faithfully endured the scattered schedule and strained lifestyle of a graduate student. I promise to pay you back for all the weekends I borrowed to write this dissertation.

This work was supported in part by National Science Foundation grants CCR-9101359 and ASC-9213149, and by Department of Energy grant DE-FG02-87ER25047.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Equality Constraints

## 1.1 Introduction

This dissertation concerns the solution of a class of algebraic optimization problems using computer algorithms. The mathematical problem is defined more precisely below, but may be broadly described as the minimization of a smooth scalar function of many unknowns subject to a number of smooth constraint equations. Optimization problems of this nature arise in many important applications; for example, parameter estimation and data fitting, design of mechanical structures, chemical process control, electrical power grid management, inverse problems in meteorology and geoscience, and the study of physical systems formulated in terms of a variational principle. The most challenging practical problems are often very large and require powerful computing resources to solve. Frequently, an algebraic optimization problem is generated by discretizing systems of ordinary or partial differential equations; hence, its size may be limited only by the capacity of the machine available to solve it. The aim of this work is to devise an improved general-purpose algorithm for solving large-scale optimization problems.

¿From a practical point of view, a problem is large-scale if the matrices involved are too big to be treated as dense quantities. An algorithm must store the matrices in sparse format and solve corresponding linear systems by efficient techniques that exploit the sparsity. Operations such as QR factorizations that can destroy sparsity must be avoided. These considerations are constantly kept in mind in this study.

The first chapter is concerned with solving the general nonlinear equality constrained optimization problem

$$\min_{x \in \mathbf{R}^n} \ f(x) \quad \text{subject to} \quad c(x) = \emptyset \ . \tag{EP}$$

The objective function $f : \mathbf{R}^n \to \mathbf{R}$ and equality constraints $c : \mathbf{R}^n \to \mathbf{R}^m$ are each assumed to be at least twice continuously differentiable. The number of unknowns $n$ is typically large, and the number of constraints $m$ may be any value not exceeding $n$.

In the second chapter simple bound constraints of the form

$$u^i \geq x^i \geq l^i, \ \text{ for } \ i = 1, \ldots, n$$

are added to problem (EP). It turns out that more general inequality constraints can be put in this form, and that bound constrained problems can be solved efficiently by reducing them to a sequence of (EP) subproblems; therefore, the bulk of this study concerns equality constrained problems.

Years of analysis and numerical experience by researchers in the field of optimization have shown that the most efficient and robust algorithms use some version of Newton's method for solving systems of equations embedded within a globalization scheme. This same framework is employed here, using the *trust region* technique for globalizing Newton's method. An algorithm of this form applied to large-scale problems must attempt to retain the favorable properties of Newton's method while approximating the Newton solution in far less than $O(n^3)$ operations. Few algorithms, and fewer software codes, address this issue directly with any success; consequently, this dissertation makes a significant step forward in providing a practical and theoretically sound method for solving large-scale constrained optimization problems.

### 1.1.1 Scholarly Contributions of this Dissertation

- A robust, efficient, thoroughly tested software implementation of a trust region-based algorithm for solving large-scale equality constrained optimization problems is the main result of this work. The completed software is called ETR, an acronym for *E*quality constrained optimization using *T*rust *R*egions. This code is not limited to problems of a certain size or structure. It achieves generality by providing multiple subproblem solution options to a user, fully exploiting second derivative information or efficiently approximating it, and interfacing easily to a user's software via reverse communications. Heuristics motivated by theory and numerical experience are included in the algorithm, improving its performance without compromising robustness. Only one other general-purpose code is known that solves large-scale problems using trust region methods, and ETR is shown to differ significantly in both philosophy and performance.

- A comprehensive logical development of the algorithm is given. Important theoretical properties are established first, then held intact as efficient methods of approximation are introduced. Local and global convergence properties are stated and proved. Most of the advanced principles utilized in developing the algorithm are not new, but this dissertation links them together into a unified whole that clarifies the role of each.

- The full use of second derivative information requires solving subproblems containing indefinite and singular Hessians. This is accomplished with an iterative solver that obtains greater efficiency using a novel "deconditioner" of the reduced Hessian matrix. The theoretical properties of this solver are carefully examined.

- The flexibility and maturity of the software makes it ideal for use in developing optimization algorithms for solving other classes of problems, most notably problems with inequality constraints. A strong start towards this goal is made by applying interior point methods from linear programming, with promising experimental results. Many ideas for algorithms of this form come to mind, and the work of this dissertation provides an excellent foundation for exploring them.

### 1.1.2 Notation and Overview

Scalars and column vectors are denoted by lowercase italicized letters, with the precise meaning made clear by the context. Sometimes a column vector is written as the transpose of a row vector for convenience; for example, $[a \;\; b]^T$ is a two-dimensional column vector. Matrices are referred to using uppercase italized letters. Iterative algorithms generate sequences of points, steps, residuals, etc., and subscripts are used to identify particular members of these sequences; for instance, $\{x_1, \ldots, x_k\}$ denotes a set of $k$ vectors. Superscripts are used to indicate components of a vector or matrix; thus, $x_k^j$ refers to the $j$th component of the vector $x_k$. Subscripts are also frequently used as a shorthand for quantities evaluated at a particular point; for example, the equality constraint vector $c : \mathbf{R}^n \to \mathbf{R}^m$ evaluated at the point $x_k$ is $c(x_k)$, or more briefly, $c_k$. The context should make all instances of this practice clear.

The symbols listed below maintain a constant definition throughout this paper.

| | |
|---|---|
| $\emptyset$ | vector or matrix with all components equal to zero |
| $x$ | vector of $n$ unknowns |
| $f(x)$ | scalar objective function to be minimized |
| $g(x)$ | gradient of $f(x)$ |
| $c(x)$ | vector of $m$ equality constraints |
| $A(x)$ | $n \times m$ Jacobian of $c(x)$: $[\nabla c^1 \cdots \nabla c^m]$ |
| $Z(x)$ | full rank $n \times (n-m)$ matrix satisfying $A^T Z = \emptyset$ (see §1.2.1) |
| $\mathcal{L}(x, \lambda)$ | Lagrangian function, such as $f(x) - \lambda^T c(x)$ (see §1.2.2) |
| $W(x, \lambda)$ | Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}(x, \lambda)$, or an approximation (see §1.2.1) |
| a_red | actual reduction of a merit function (see §1.5.6) |
| p_red | predicted reduction based on a model merit function (see §1.5.6) |

Chapter 1 analyses nonlinear equality constrained problems of the form (EP) and describes the development of the ETR algorithm to solve them. Section 1.2 starts by deriving conditions that identify solutions of (EP), and adapts Newton's method as an iterative technique to compute them. Local convergence properties of this method are stated and proved. Section 1.3 discusses trust region techniques that make Newton's method globally convergent. A survey of proposed trust region algorithms is made and the most promising idea with respect to solving large-scale problems is selected. In section 1.4 the subject of solving large unconstrained trust region subproblems is covered. These subproblems feature prominently in the implementation of ETR, which is presented in detail in section 1.5. Numerical test results are examined in section 1.6, including a comparison of ETR to an alternative trust region algorithm available as a commercial product. Section 1.7 proves global convergence for an abstract form of the algorithm. The actual implemented version is compared and its convergence properties discussed. A final summary and list of possible future enhancements concludes the chapter.

Chapter 2 considers how to solve optimization problems with nonlinear inequality constraints using the results of chapter 1. These problems can be reduced to subproblems of the form (EP) by various techniques, one of which is pursued here. Section 2.2 outlines the primal barrier interior point method of linear programming, then shows one way of generalizing this idea to solve nonlinear nonconvex problems. In section 2.3 an algorithm is devised that makes maximum use of the ETR software of chapter 1. Some test results are provided in section 2.4. This chapter represents new ideas still under investigation, and is less comprehensive than chapter 1.

## 1.2 Local Methods

The goal of this section is to develop a locally convergent iterative algorithm for solving problem (EP). The algorithm needs to be computationally efficient when there are large numbers of variables and constraints, yet converge rapidly near a solution. If there are no constraints in (EP), then an excellent well-known local algorithm is to apply Newton's method for solving the system of equations $\nabla f(x) = \emptyset$. These equations describe a necessary condition for a point to be a local minimum of the unconstrained problem. Newton's method requires second derivative information, but enjoys a Q-quadratic convergence rate and is guaranteed to converge if started close enough to a solution.

In this section a variation of Newton's method is derived for problems with nonlinear equality constraints. This algorithm is proved to be locally convergent with a Q-quadratic rate, and is designed to be computationally efficient for solving large-scale problems. The material in this section is not new (most of it can be found in such standard references as [34] and [39]), but a consistent and comprehensive presentation helps provide a strong foundation for the rest of this dissertation.

Like Newton's method for unconstrained optimization, the algorithm developed in this section solves a system of equations that describe a necessary condition for local solutions to problem (EP). Therefore, this section begins by deriving the conditions that are necessary and sufficient for a point $x_*$ to be a solution of (EP).

### 1.2.1 Local Solutions

A local solution is a set of points defined with respect to some connected region $D$ of $\mathbf{R}^n$. It consists of those feasible points in $D$ for which the objective attains its minimum. If $D$ equals $\mathbf{R}^n$ or is big enough to encompass all feasible points in $\mathbf{R}^n$, then the local solution is also a global solution. In general, a local solution does not have to be a connected or bounded set, but it is more convenient to assume that it has these properties. Since $f(x)$ and every $c^i(x)$ are assumed continuous, a local solution set can always be made connected by defining a sufficiently small region $D$. Solution points at infinity need to be excluded because a numerical algorithm cannot converge to a point at infinity. A common way of eliminating solutions at infinity is to assume all level sets of $f(x)$ are bounded, but for this section let us just exclude such points from the definition.

**Definition 1.2.1** *A* **local solution** *to (EP) is a nonempty bounded connected set of points $S$ for some associated domain $D$ over which $f(x)$ and all $c^i(x)$ are defined. The set $S \subset D$, and for all $x \in S$ both $c(x) = \emptyset$ and $f(x) = argmin\ \{f(y) : y \in D \text{ and } c(y) = \emptyset\}$ are true. If $S$ contains only one point, then that point is called an* **isolated local solution**.

The general nonlinear problem (EP) may have many local solutions, with $S$ ranging from an isolated point to almost any imaginable closed set. The problem may also have no local solutions. This dissertation is not concerned with determining the existence of a solution, so it is assumed at least one local solution exists.

Every local solution to (EP) has certain well-known properties, most importantly, the first- and second-order KKT conditions. These conditions state that a point $x_*$ is a member of a local solution set only if there are no feasible directions from $x_*$ along which $f(x)$ decreases. To be mathematically rigorous, the meaning of "feasible direction" must be made precise in terms of sequences and limits.

**Definition 1.2.2** *Let $x_*$ be a point where $c(x_*) = \emptyset$. Let $\{d_1, d_2, \ldots\}$ be a sequence of unit vectors whose limit is the unit vector $d$. Let $\{\alpha_1, \alpha_2, \ldots\}$ be a corresponding sequence of nonnegative scalars with $\alpha_1 > 0$ whose limit is $0$. Now if $c(x + \alpha_k d_k) = \emptyset$ holds for all $k \geq 1$, then $d$ is a (unit)* **feasible direction** *at $x_*$.*

It can be proved (for example, [34, pp. 203-204]) that if all constraints are continuously differentiable and the $n \times m$ constraint Jacobian $A(x_*)$ has full rank $m$, then $d$ is a feasible direction at $x_*$ if and only if $d^T A(x_*) = \emptyset$; that is, $d$ must be in the tangent subspace of every constraint at $x_*$. However, when $A(x_*)$ is rank deficient this no longer holds and there is no clear way to characterize feasible directions. For instance, if a problem has two constraints, each a hypersphere in $\mathbf{R}^n$, with the hyperspheres touching one another tangentially at a single point $x_*$, then the rank of $A(x_*)$ is one and there are no feasible directions at $x_*$. Throughout this paper the fundamental *regularity assumption* (or *constraint qualification*) is made that $A(x_*)$ is full rank.

**Theorem 1.2.3** *Assume $f(x)$ and all $c^i(x)$ are continuously differentiable, and that $A(x_*)$ is full rank. The point $x_*$ is a local solution of (EP) only if*

$$c(x_*) = \emptyset \quad \textit{and} \quad \nabla f(x_*) = A(x_*)\,\lambda_* \tag{1.2.1}$$

*both hold, where $\lambda_*$ is a unique $m$-vector. These are called the* **first-order necessary KKT conditions**.

*Proof.* The regularity assumption implies that every feasible unit direction $d$ satisfies $d^T A(x_*) = \emptyset$. A Taylor series expansion of $f(x_* + \epsilon d)$ about the point $x_*$ shows that for sufficiently small $\epsilon > 0$ the objective can decrease only if $d^T \nabla f(x_*) < 0$. Therefore, a necessary condition for $x_*$ to be a local solution is that the set $\{d : d^T A(x_*) = \emptyset$ and $d^T \nabla f(x_*) < 0\}$ be empty. Farkas' Lemma proves that this statement is equivalent to (1.2.1) [34, p. 205]. The regularity assumption also ensures that $A(x_*)^T A(x_*)$ is nonsingular; hence, $\lambda_*$ is uniquely determined by the formula $\lambda_* = [A(x_*)^T A(x_*)]^{-1} A(x_*)^T \nabla f(x_*)$. □

When $A(x_*)$ is full rank, then there exists an $(n-m)$-dimensional tangent linear manifold of feasible directions passing through the point $x_*$. Let the matrix $Z_*$ be an orthogonal complement of $A(x_*)$, so that $A(x_*)^T Z_* = \emptyset$ and $Z_*$ is of rank $n-m$ (orthogonal complements are defined in A.1.2). Then every feasible direction at $x_*$ is in the column space of $Z_*$, and an alternate form of the first-order KKT conditions is obtained by multiplying (1.2.1) by $Z_*^T$:

$$c(x_*) = \emptyset \quad \text{and} \quad Z_*^T \nabla f(x_*) = \emptyset. \tag{1.2.2}$$

Equations (1.2.2) and (1.2.1) are equivalent if they define the same set of points $x_*$. Obviously, an $x_*$ satisfying (1.2.1) also satisfies (1.2.2). Conversely, if (1.2.2) holds at $x_*$ and the gradient is written in terms of its orthogonal projections $g^a$ and $g^z$ (using Theorem A.1.4) as

$$\nabla f(x_*) = A(x_*)g^a + Z_* g^z,$$

then (1.2.2) implies that $g^z$ is zero, and (1.2.1) implies that $g^a = \lambda_*$. Thus, (1.2.1) and (1.2.2) are equivalent.

Extending the Taylor series expansion of $f$ and $c^i$ one more term leads to the second-order condition, which together with (1.2.1) provides necessary and sufficient conditions for a local solution. It is useful at this point to introduce the matrix

$$W(x,\lambda) \equiv \nabla^2 f(x) - \sum_{i=1}^{m} \lambda^i \, \nabla^2 c^i(x) \tag{1.2.3}$$

and to denote $W(x_*, \lambda_*)$ by $W_*$.

**Theorem 1.2.4** *Assume $f(x)$ and all $c^i(x)$ are twice continuously differentiable, and that $A(x_*)$ is full rank. The point $x_*$ is an isolated local solution of (EP) if and only if the first-order equations (1.2.1) hold and*

$$d^T W_* d > 0 \quad \textit{for all feasible directions } d. \tag{1.2.4}$$

*Equation (1.2.4) is called the (strict)* **second-order KKT condition**.

*Proof.* Consider the quantity $f(x_* + \epsilon d) - \lambda_*^T c(x_* + \epsilon d)$, where $\epsilon > 0$ and $d$ is a unit vector. Substitute a two-term Taylor series expansion for both $f$ and $c$ about the point $x_*$, giving

$$\begin{aligned} &f(x_* + \epsilon d) - \lambda_*^T c(x_* + \epsilon d) = \\ &\qquad f(x_*) + \epsilon d^T \nabla f(x_*) - \lambda_*^T c(x_*) - \epsilon d^T A(x_*)\lambda_* + \frac{\epsilon^2}{2} d^T W_* d + O(\epsilon^3). \end{aligned}$$

Applying (1.2.1) leaves only

$$f(x_* + \epsilon d) - \lambda_*^T c(x_* + \epsilon d) = f(x_*) + \frac{\epsilon^2}{2} d^T W_* d + O(\epsilon^3).$$

For sufficiently small $\epsilon$ it is clear that $f(x_*)$ is a local minimum if and only if the term $d^T W_* d$ is positive. Furthermore, the condition only needs to hold for feasible directions $d$, since small displacements along infeasible directions always move to infeasible points (this is a consequence of Farkas' Lemma). □

If the second-order condition is not a strict inequality and $d^T W_* d$ can equal zero for some feasible directions, then the condition is no longer sufficient for a local solution, although it is still necessary. In this case, the local solution can be a connected set of points instead of just an isolated point. Although this situation occurs in real problems, the derivation of our local algorithm generally assumes a strict inequality in (1.2.4).

### 1.2.2 Lagrange Multipliers

The KKT conditions are expressed more compactly using the Lagrange multiplier formulation of equality constrained optimization. Let $\lambda$ now be a vector of $m$ independent variables, and define

$$\mathcal{L}(x, \lambda) \equiv f(x) - \lambda^T c(x). \tag{1.2.5}$$

Then the first-order KKT conditions (1.2.1) are equivalent to

$$\nabla_x \mathcal{L}(x_*, \lambda_*) = \emptyset \quad \text{and} \quad \nabla_\lambda \mathcal{L}(x_*, \lambda_*) = \emptyset, \tag{1.2.6}$$

where the notation means that the gradients are evaluated at the point $[x_* \;\; \lambda_*]^T$. Note that $\nabla_x^2 \mathcal{L}(x, \lambda) = W(x, \lambda)$, and so the second-order KKT condition (1.2.4) can be written as

$$d^T \nabla_x^2 \mathcal{L}(x_*, \lambda_*)\, d > 0 \quad \text{for all feasible directions } d.$$

The method of Lagrange multipliers solves problem (EP) by introducing the artificial variables $\lambda$. As noted in Theorem 1.2.3, $\lambda_*$ is uniquely determined by $x_*$ if $A(x_*)$ is full rank, so the artificial variables become redundant at a local solution. The main motivation for introducing Lagrange multipliers is to express the first-order KKT conditions (1.2.1) as a system of $n + m$ equations in $n + m$ unknowns, an idea that will be developed in the following section. But the multipliers also have a useful physical interpretation as measures of sensitivity.

Suppose for a unit vector $d$ that $x_* + \epsilon d$ is a point displaced from the local solution $x_*$ in such a way that the constraint $c^j$ is violated, but all other constraints are still satisfied. For small $\epsilon$,

$$\begin{aligned} f(x_* + \epsilon d) &= f(x_*) + \epsilon d^T \nabla f(x_*) + O(\epsilon^2), \quad \text{and} \\ c^i(x_* + \epsilon d) &= c^i(x_*) + \epsilon d^T \nabla c^i(x_*) + O(\epsilon^2), \quad \text{for} \quad i = 1, \ldots, m. \end{aligned}$$

The second equation shows that the term $d^T \nabla c^i(x_*)$ is of $O(\epsilon)$ for every constraint not being violated. This fact and the first-order KKT condition (1.2.1) shows that

$$\begin{aligned} d^T \nabla f(x_*) = d^T A(x_*) \lambda_* &= \sum_{i=1}^{m} \lambda_*^i d^T \nabla c^i(x_*) \\ &= \lambda_*^j d^T \nabla c^j(x_*) + O(\epsilon). \end{aligned}$$

Substituting this back into the expansion for $f(x_* + \epsilon d)$ gives

$$\frac{f(x_* + \epsilon d) - f(x_*)}{\epsilon} = \lambda_*^j d^T \nabla c^j(x_*) + O(\epsilon),$$

and the left-hand side becomes a directional derivative as $\epsilon \to 0$. Therefore, the sensitivity of $f$ to a slight perturbation of $x_*$ in a direction that violates one constraint is proportional to the Lagrange multiplier associated with that constraint. Note that this result holds only at a point where the first-order KKT condition is satisfied.

### 1.2.3 Newton's Method for the KKT Equations

Following up the idea introduced in the last section, let us treat the Lagrange multipliers $\lambda$ as independent variables and consider the first-order KKT conditions (1.2.1) or (1.2.6) as a system of $n+m$ equations in the $n+m$ unknowns $x$ and $\lambda$. This system (sometimes called the augmented system) can be written in the form

$$F(x,\lambda) \equiv \begin{bmatrix} \nabla_x \mathcal{L}(x,\lambda) \\ \nabla_\lambda \mathcal{L}(x,\lambda) \end{bmatrix} = \begin{bmatrix} g(x) - A(x)\lambda \\ -c(x) \end{bmatrix} = \emptyset, \qquad (1.2.7)$$

where $g(x)$ is used to denote the gradient $\nabla f(x)$. By Theorem 1.2.3, every local solution to (EP) must satisfy this system and hence be a stationary point of the Lagrangian function. Although the augmented system only contains information about the first-order KKT conditions, it is still an excellent starting point for a general algorithm. Therefore, let us apply Newton's method to iteratively solve the system of nonlinear equations $F(x,\lambda) = \emptyset$, an idea first suggested by Wilson [79].

The Newton step for this system requires the Jacobian of $F$, also known as the Lagrangian matrix $K$. Using equation (1.2.3) this matrix can be expressed in terms of $W(x,\lambda)$ as

$$K(x,\lambda) \equiv \begin{bmatrix} \nabla^2_{xx}\mathcal{L}(x,\lambda) & \nabla^2_{x\lambda}\mathcal{L}(x,\lambda) \\ \nabla^2_{\lambda x}\mathcal{L}(x,\lambda) & \nabla^2_{\lambda\lambda}\mathcal{L}(x,\lambda) \end{bmatrix} = \begin{bmatrix} W(x,\lambda) & -A(x) \\ -A(x)^T & \emptyset \end{bmatrix}, \qquad (1.2.8)$$

and the Newton step from the point $[x_k \;\; \lambda_k]^T$ is

$$\begin{bmatrix} x_{k+1} \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ \lambda_k \end{bmatrix} - K_k^{-1} \begin{bmatrix} g_k - A_k\lambda_k \\ -c_k \end{bmatrix}. \qquad (1.2.9)$$

The subscript $k$ is used to indicate that a quantity is evaluated at the point $[x_k \;\; \lambda_k]^T$; for instance, $A_k = A(x_k)$ and $K_k = K(x_k, \lambda_k)$.

The full iterative Newton method for finding a point $[x_* \;\; \lambda_*]^T$ at which $F(x_*, \lambda_*) = \emptyset$ is given below in Algorithm 1.2.5.

**Algorithm 1.2.5** *Newton's method for solving the first-order KKT equations*

Start point $[x_0 \;\; \lambda_0]^T$ and stop tolerance $\epsilon > 0$ are given
LOOP, starting with $k = 0$
    Compute $c_k$, $g_k$ and $A_k$
    IF $\|g_k - A_k\lambda_k\| < \epsilon$ AND $\|c_k\| < \epsilon$ THEN STOP
    Compute $W_k = \nabla^2_x \mathcal{L}_k$
    Compute the Newton step given by equation (1.2.9)
CONTINUE, after incrementing $k$

To compute Newton steps using equation (1.2.9) the Lagrangian matrix $K$ must be nonsingular at every iteration. It turns out that $K$ is nonsingular at $x_*$ if the point satisfies the second-order KKT condition and if the regularity assumption on $A_*$ holds. By continuity, the nonsingularity of the matrix $K$ can be extended over a local neighborhood of $x_*$; hence, Algorithm 1.2.5 is well-defined when iterates are sufficiently close to a solution point $x_*$. These statements are made more precise in Lemmas 1.2.6 and 1.2.7.

**Lemma 1.2.6** *If matrices $A$ and $Z$ are orthogonal complements (which implies that $A$ and $Z$ are both full rank and that $A^TZ = \emptyset$ — see Definition A.1.2 in the Appendix), $W$ is any $n \times n$ symmetric matrix and $Z^TWZ$ is positive definite, then the matrix*

$$\begin{bmatrix} W & -A \\ -A^T & \emptyset \end{bmatrix}$$

*is nonsingular.*

*Proof.* The proof is by contradiction. Assume the matrix is singular, so that for vectors $d \in \mathbf{R}^n$ and $\pi \in \mathbf{R}^m$

$$\begin{bmatrix} W & -A \\ -A^T & \emptyset \end{bmatrix} \begin{bmatrix} d \\ \pi \end{bmatrix} = \emptyset, \tag{1.2.10}$$

and either $d$ or $\pi$ (or both) is nonzero. Since $A$ and $Z$ are orthogonal complements, every vector $d$ has unique projections $d^a \in \mathbf{R}^m$ and $d^z \in \mathbf{R}^{n-m}$, so that the vector may be expressed in the form $d = Ad^a + Zd^z$ (see Theorem A.1.4 in the Appendix). The lower row of (1.2.10) becomes

$$-A^Td = -A^TAd^a - A^TZd^z = -A^TAd^a = \emptyset.$$

When $A$ is full rank, $A^TA$ is nonsingular; therefore, $d^a = \emptyset$. Now the upper row of (1.2.10) is

$$WZd^z - A\pi = \emptyset.$$

Multiplying by $Z^T$ reduces this to $Z^TWZd^z = \emptyset$, and since $Z^TWZ$ is positive definite, $d^z = \emptyset$. This leaves us with $A\pi = \emptyset$, and again the full rank of $A$ implies that $\pi = \emptyset$. Therefore, both $d$ and $\pi$ are zero, contradicting the hypothesis of singularity. □

**Lemma 1.2.7** *Assume $f(x)$ and all $c^i(x)$ are twice continuously differentiable, that $A_*$ is full rank, and that $x_*$ is an isolated local solution satisfying the necessary and sufficient KKT conditions (1.2.1) and (1.2.4). Then there exists some $\delta_1 > 0$ such that*

$$K(x,\lambda) = \begin{bmatrix} W(x,\lambda) & -A(x) \\ -A(x)^T & \emptyset \end{bmatrix}$$

*is nonsingular at all points $[x \;\; \lambda]^T$ satisfying*

$$\left\| \begin{bmatrix} x \\ \lambda \end{bmatrix} - \begin{bmatrix} x_* \\ \lambda_* \end{bmatrix} \right\| < \delta_1.$$

*Proof.* Since $A_*$ is full rank, the regularity assumption holds and all feasible directions $d$ at $x_*$ satisfy $d^TA_* = \emptyset$. Let $Z_*$ be an orthogonal complement of $A_*$. Then every $u \in \mathbf{R}^{n-m}$ defines a feasible direction $d = Z_*u$ at $x_*$. Now by (1.2.4), $u^TZ_*^TW_*Z_*u > 0$ for all $u$, and therefore $Z_*^TW_*Z_*$ is positive definite. Applying Lemma 1.2.6, we deduce that

$$\begin{bmatrix} W_* & -A_* \\ -A_*^T & \emptyset \end{bmatrix}$$

is nonsingular.

The assumptions imply that all elements of $A(x)$ and $W(x,\lambda)$ are continuous everywhere. Recall that the eigenvalues of a continuous matrix are themselves continuous functions [62, p. 35]. Now at $x_*$ we know there are $m$ linearly independent rows of $A_*$, so these form an $m \times m$ nonsingular basis matrix $B_*$. The continuity of the eigenvalues of $B(x)$ implies the existence of a finite neighborhood $\mathcal{N} = \{x : \|x - x_*\| < \epsilon_1\}$ in which $B(x)$ remains nonsingular and, hence, $A(x)$ remains full rank. In this neighborhood $B(x)^{-1}$ exists and is continuous [62, p. 46], so $Z(x)$ can always be chosen by direct elimination to be continuous and orthogonal to $A(x)$. Therefore, the $m \times m$ matrix $Z(x)^T W(x,\lambda) Z(x)$ is a continuous function for $x \in \mathcal{N}$ and every $\lambda \in \mathbf{R}^m$. Continuity of the eigenvalues further implies that $Z(x)^T W(x,\lambda) Z(x)$ remains positive definite in some finite neighborhood of $[x_* \;\; \lambda_*]^T$, which may be characterized as $\{[x \;\; \lambda]^T : \|x - x_*\| < \epsilon_2 \text{ and } \|\lambda - \lambda_*\| < \epsilon_3\}$. Now choose $0 < \delta_1 \leq \min\{\epsilon_1, \epsilon_2\} + \epsilon_3$, and consider the set of points $[x \;\; \lambda]^T$ satisfying

$$\left\| \begin{bmatrix} x \\ \lambda \end{bmatrix} - \begin{bmatrix} x_* \\ \lambda_* \end{bmatrix} \right\| < \delta_1.$$

We have established that $A(x)$ is full rank and $Z(x)^T W(x,\lambda) Z(x)$ is positive definite for every such point; therefore, by Lemma 1.2.6, the Lagrangian matrix $K$ is nonsingular. $\square$

Lemmas 1.2.6 and 1.2.7 show that Algorithm 1.2.5 is well-defined for points close to a solution. After one more technical lemma, we will be able to prove that the algorithm is convergent at a quadratic rate, the hallmark property of Newton's method. The proof in Theorem 1.2.9 is a straightforward adaptation of the classical proof for Newton's method on systems of equations (see [62] or [20]). But first, a Lipschitz condition is established for the Lagrangian matrix $K$.

**Lemma 1.2.8** *Let $D_x$ be a closed bounded region in $\mathbf{R}^n$, and $D_\lambda$ a closed bounded region in $\mathbf{R}^m$. Assume that $f(x)$ and all $c^i(x)$ are continuously differentiable and that their Hessians satisfy Lipschitz conditions for every $x \in D_x$. Then for all points $[x_1 \;\; \lambda_1]^T$ and $[x_2 \;\; \lambda_2]^T$ for which $x_1, x_2 \in D_x$ and $\lambda_1, \lambda_2 \in D_\lambda$, there exists a finite constant $\gamma_K$ such that*

$$||K_1 - K_2|| \leq \gamma_K \left\| \begin{bmatrix} x_1 - x_2 \\ \lambda_1 - \lambda_2 \end{bmatrix} \right\|.$$

*Proof.* First, observe that

$$\begin{aligned} K_1 - K_2 &= \begin{bmatrix} W_1 & -A_1 \\ -A_1^T & \emptyset \end{bmatrix} - \begin{bmatrix} W_2 & -A_2 \\ -A_2^T & \emptyset \end{bmatrix} \\ &= \begin{bmatrix} (\nabla^2 f_1 - \nabla^2 f_2 - \sum_{i=1}^{m} [\lambda_1^i \nabla^2 c_1^i - \lambda_2^i \nabla^2 c_2^i]) & (-A_1 + A_2) \\ -A_1^T + A_2^T & \emptyset \end{bmatrix}. \end{aligned}$$

Taking the $\ell_1$ norm of both sides, we obtain

$$
\begin{aligned}
||K_1 - K_2||_1 \;\; &\leq \;\; \|\nabla^2 f_1 - \nabla^2 f_2\|_1 \; + \; \|\sum_{i=1}^{m}(\lambda_1^i - \lambda_2^i)\nabla^2 c_1^i\|_1 \\
&\quad + \|\sum_{i=1}^{m}\lambda_2^i(\nabla^2 c_1^i - \nabla^2 c_2^i)\|_1 \; + \; \|A_1 - A_2\|_1.
\end{aligned}
$$

The hypotheses of this lemma imply the existence of finite constants satisfying

$$
\begin{array}{ccll}
\|\nabla^2 f(x_1) - \nabla^2 f(x_2)\|_1 & \leq & \gamma_f \|x_1 - x_2\|_1 & \\
\|\nabla^2 c^i(x_1) - \nabla^2 c^i(x_2)\|_1 & \leq & \gamma_c \|x_1 - x_2\|_1, & \text{for} \quad i = 1, \ldots, m \\
\|\nabla^2 c^i(x)\|_1 & \leq & \beta_c, & \text{for} \quad i = 1, \ldots, m \\
\|\lambda\|_1 & \leq & \beta_\lambda & \\
\|A(x_1) - A(x_2)\|_1 & \leq & \gamma_A \|x_1 - x_2\|_1. &
\end{array}
$$

The third and fifth inequalities above are not explicitly stated in the lemma, but are implied by the lemma hypotheses. The Hessians of the constraints are bounded because they satisfy a Lipschitz condition over a bounded region, and the Lipschitz condition on $A(x)$ follows from the constraint Hessians being bounded and the constraint gradients being continuous. In terms of these constants, we have

$$
\begin{aligned}
||K_1 - K_2||_1 \;\; &\leq \;\; (\gamma_f + \gamma_A + m\beta_\lambda\gamma_c)\|x_1 - x_2\|_1 + \beta_c\|\lambda_1 - \lambda_2\|_1 \\
&\leq \;\; \max\{\gamma_f + \gamma_A + m\beta_\lambda\gamma_c, \; \beta_c\} \left\| \begin{bmatrix} x_1 - x_2 \\ \lambda_1 - \lambda_2 \end{bmatrix} \right\|_1 .
\end{aligned}
$$

Finally, let $c_v$ be the equivalence constant between the $\ell_1$ norm and any other norm. Then defining

$$
\gamma_K \equiv c_v \max\{\gamma_f + \gamma_A + m\beta_\lambda\gamma_c, \; \beta_c\}
$$

gives the desired result. □

**Theorem 1.2.9** *Assume that $f(x)$ and all $c^i(x)$ are twice continuously differentiable, that their Hessians satisfy Lipschitz conditions, and that all iterates $\lambda_k$ stay in a closed bounded region. Let $x_*$ be an isolated local solution of (EP) satisfying the necessary and sufficient KKT conditions (1.2.1) and (1.2.4), and assume that $A_*$ is full rank. Then if the start point $[x_0 \;\; \lambda_0]^T$ is sufficiently close to $[x_* \;\; \lambda_*]^T$, Algorithm 1.2.5 converges to $[x_* \;\; \lambda_*]^T$, and the rate of convergence is Q-quadratic.*

*Proof.* The continuity of $f$ and $c$ and their first derivatives allows us to write a Taylor series for the gradient of the Lagrangian at $[x_* \;\; \lambda_*]^T$ expanded about the point $[x_k \;\; \lambda_k]^T$. Using the integral form of the remainder (see (A.3.2) in the Appendix), this is

$$
\begin{bmatrix} \nabla_x \mathcal{L}_* \\ \nabla_\lambda \mathcal{L}_* \end{bmatrix} = \begin{bmatrix} \nabla_x \mathcal{L}_k \\ \nabla_\lambda \mathcal{L}_k \end{bmatrix} + \int_0^1 \begin{bmatrix} \nabla^2_{xx}\mathcal{L}(a(t), b(t)) & \nabla^2_{x\lambda}\mathcal{L}(a(t), b(t)) \\ \nabla^2_{\lambda x}\mathcal{L}(a(t), b(t)) & \nabla^2_{\lambda\lambda}\mathcal{L}(a(t), b(t)) \end{bmatrix} \begin{bmatrix} x_* - x_k \\ \lambda_* - \lambda_k \end{bmatrix} dt,
$$

where

$$\begin{aligned} a(t) &\equiv x_k + t(x_* - x_k) \\ b(t) &\equiv \lambda_k + t(\lambda_* - \lambda_k). \end{aligned}$$

By the first-order KKT conditions, the left-hand side is zero. Substituting from (1.2.7) and (1.2.8) then yields

$$-\begin{bmatrix} g_k - A_k\lambda_k \\ -c_k \end{bmatrix} = \int_0^1 K(a(t), b(t)) \begin{bmatrix} x_* - x_k \\ \lambda_* - \lambda_k \end{bmatrix} dt.$$

Let us rewrite the Newton step (1.2.9) in the form

$$-\begin{bmatrix} g_k - A_k\lambda_k \\ -c_k \end{bmatrix} = K_k \begin{bmatrix} x_{k+1} - x_* \\ \lambda_{k+1} - \lambda_* \end{bmatrix} + K_k \begin{bmatrix} x_* - x_k \\ \lambda_* - \lambda_k \end{bmatrix}.$$

Now equate the last two expressions and move the term that is independent of $t$ inside the integral, obtaining

$$K_k \begin{bmatrix} x_{k+1} - x_* \\ \lambda_{k+1} - \lambda_* \end{bmatrix} = \int_0^1 [K(a(t), b(t)) - K_k] \begin{bmatrix} x_* - x_k \\ \lambda_* - \lambda_k \end{bmatrix} dt. \tag{1.2.11}$$

According to Lemma 1.2.7, the Lagrangian matrix $K_k$ is invertible if $[x_k \;\; \lambda_k]^T$ is sufficiently close to $[x_* \;\; \lambda_*]^T$. This means the norm of $K_k^{-1}$ is bounded by some finite constant $\beta_K$ in the neighborhood of $[x_* \;\; \lambda_*]^T$, i.e.,

$$||K_k^{-1}|| \leq \beta_K, \quad \text{when} \quad \left|\left| \begin{bmatrix} x_k - x_* \\ \lambda_k - \lambda_* \end{bmatrix} \right|\right| < \delta_1.$$

Then multiplying both sides of (1.2.11) by the inverse of the Lagrangian matrix and taking norms of both sides, we see that

$$\left|\left| \begin{bmatrix} x_{k+1} - x_* \\ \lambda_{k+1} - \lambda_* \end{bmatrix} \right|\right| \leq \beta_K \left|\left| \begin{bmatrix} x_* - x_k \\ \lambda_* - \lambda_k \end{bmatrix} \right|\right| \int_0^1 ||K(a(t), b(t)) - K(x_k, \lambda_k)||\, dt.$$

Applying Lemma 1.2.8 to the term inside the integral reduces this to

$$\begin{aligned} \left|\left| \begin{bmatrix} x_{k+1} - x_* \\ \lambda_{k+1} - \lambda_* \end{bmatrix} \right|\right| &\leq \beta_K \left|\left| \begin{bmatrix} x_* - x_k \\ \lambda_* - \lambda_k \end{bmatrix} \right|\right| \int_0^1 \gamma_K \left|\left| \begin{bmatrix} t(x_* - x_k) \\ t(\lambda_* - \lambda_k) \end{bmatrix} \right|\right| dt \\ &\leq \beta_K \gamma_K \left|\left| \begin{bmatrix} x_k - x_* \\ \lambda_k - \lambda_* \end{bmatrix} \right|\right|^2, \end{aligned} \tag{1.2.12}$$

where $\gamma_K$ is the constant of Lemma 1.2.8. Remember that relation (1.2.12) is true only if $[x_k \;\; \lambda_k]^T$ is within $\delta_1$ of $[x_* \;\; \lambda_*]^T$.

Next, consider Algorithm 1.2.5 and assume that

$$\left\| \begin{bmatrix} x_0 - x_* \\ \lambda_0 - \lambda_* \end{bmatrix} \right\| \leq \delta_2, \quad \text{where} \quad \delta_2 \equiv \tau \min\{\delta_1, \frac{1}{\beta_K \gamma_K}\}, \ \ \tau \in (0,1). \tag{1.2.13}$$

We know from Lemmas 1.2.7 and 1.2.8 that $\beta_K$ and $\gamma_K$ are bounded, so $0 < \delta_2 < \delta_1$. The condition (1.2.13) establishes a closed bounded region of possible start points $[x_0 \ \ \lambda_0]^T$ for Algorithm 1.2.5. The Lagrangian matrix $K_0$ is nonsingular for such start points, and by (1.2.12) the first Newton step to $[x_1 \ \ \lambda_1]^T$ satisfies

$$\left\| \begin{bmatrix} x_1 - x_* \\ \lambda_1 - \lambda_* \end{bmatrix} \right\| \ \leq \ \delta_2 \beta_K \gamma_K \left\| \begin{bmatrix} x_0 - x_* \\ \lambda_0 - \lambda_* \end{bmatrix} \right\| \ \leq \ \tau \left\| \begin{bmatrix} x_0 - x_* \\ \lambda_0 - \lambda_* \end{bmatrix} \right\|.$$

Thus, the point $[x_1 \ \ \lambda_1]^T$ is closer to the solution than $[x_0 \ \ \lambda_0]^T$, meaning that $K_1$ is also nonsingular. A simple induction argument now proves that all subsequent iterates $[x_k \ \ \lambda_k]^T$ generated by Algorithm 1.2.5 are well-defined, each moving closer to the solution by at least a factor $\tau$. Therefore,

$$\left\| \begin{bmatrix} x_k - x_* \\ \lambda_k - \lambda_* \end{bmatrix} \right\| \leq \tau^k \left\| \begin{bmatrix} x_0 - x_* \\ \lambda_0 - \lambda_* \end{bmatrix} \right\|, \quad \text{and} \quad \lim_{k \to \infty} \left\| \begin{bmatrix} x_k - x_* \\ \lambda_k - \lambda_* \end{bmatrix} \right\| = 0.$$

This proves that Algorithm 1.2.5 converges, and relation (1.2.12) shows that the rate of convergence is quadratic. □

### 1.2.4 Solving for the Newton Step

Theorem 1.2.9 establishes that the repeated application of the Newton step (1.2.9) in Algorithm 1.2.5 leads to convergence at a quadratic rate in the neighborhood of a local solution. The theorem proves that a unique Newton step exists at every iteration, but does not really specify how to compute the step. In this section three classical methods of computing the Newton step (1.2.9) are examined, with special consideration given to large-scale problems. To this end, rewrite the problem of finding the Newton step in the form

$$K_k \begin{bmatrix} d_k \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} W_k & -A_k \\ -A_k^T & \emptyset \end{bmatrix} \begin{bmatrix} d_k \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -g_k \\ c_k \end{bmatrix}, \tag{1.2.14}$$

where $d_k = x_{k+1} - x_k$ and $\lambda_{k+1}$ are the unknowns, and the terms containing $\lambda_k$ have cancelled out.

The first method inverts the full $(n+m) \times (n+m)$ Lagrangian matrix $K_k$. This matrix is symmetric and indefinite, and may be factored directly using the algorithm of Bunch and Parlett [6] into the form $K_k = LDL^T$, where $D$ is a block diagonal matrix with blocks of size $1 \times 1$ and $2 \times 2$. This factorization is known to be as numerically stable as Gaussian elimination if full or partial pivoting is used [4], [5]. For large problems, $A_k$ and $W_k$ are sparse matrices and a factorization of $K_k$ needs to remain sparse. Duff and Reid et al. [26] addressed this by developing a sparse version of the Bunch-Parlett factorization, choosing pivots according to a generalization of the Markowitz criteria. Sparse symmetric indefinite matrix factorization is an ongoing area of research, particularly in trying to exploit the kind of block structure present in $K_k$ (see for example [23] and [35]). The inverse of a symmetric indefinite matrix can also be approximated by iterative methods, a number of which have been proposed in recent years, including BCG [31], MINRES/SYMMLQ [63], GMRES [70], CGS [72], and QMR [36]. All of these algorithms are based on the Lanczos method for computing eigenvalues, and all of them aspire to extend the conjugate gradient method to general linear systems (including nonsymmetric systems). Conjugate gradient, of course, is restricted to positive definite matrices, but possesses the desirable properties of minimizing over an expanding Krylov subspace using only matrix-vector products and requiring storage of just three vectors. Unfortunately, none of the more general iterative methods have the same set of properties.

The *range space* (or dual Newton) method is a second way of computing the Newton step that makes better use of the block structure of the Lagrangian matrix. Assuming $W_k$ is nonsingular, its inverse can be used to perform a Gaussian elimination operation on the matrix $K_k$, so that (1.2.14) becomes

$$\begin{bmatrix} A_k^T & -A_k^T W_k^{-1} A_k \\ -A_k^T & \emptyset \end{bmatrix} \begin{bmatrix} d_k \\ \lambda_{k+1} \end{bmatrix} = \begin{bmatrix} -A_k^T W_k^{-1} g_k \\ c_k \end{bmatrix}.$$

Then the Newton step is given by

$$\begin{aligned} \lambda_{k+1} &= (A_k^T W_k^{-1} A_k)^{-1} (A_k^T W_k^{-1} g_k - c_k) \\ d_k &= -W_k^{-1} (g_k - A_k \lambda_{k+1}). \end{aligned}$$

Notice that the "dual" variables $\lambda_{k+1}$ are calculated first, and that the $m \times m$ matrix $A_k^T W_k^{-1} A_k$ is the projection of $W_k^{-1}$ onto the range space of $A_k$. Gould [44] proved that $K_k$ is nonsingular if and only if the number of negative eigenvalues of $W_k^{-1}$ equals the number of negative eigenvalues of $A_k^T W_k^{-1} A_k$, provided that $W_k$ is nonsingular. The difficulties with the range space method are that $W_*$ may be singular at a solution $x_*$, and that $W_k^{-1}$ and $A_k^T W_k^{-1} A_k$ are in general indefinite. However, if the matrices happen to be positive definite, then a technique devised by Fletcher [34, pp. 237–239] can be used to simultaneously factor both $W_k$ and $A_k^T W_k^{-1} A_k$.

The third method of solving (1.2.14) also exploits the block structure of the Lagrangian matrix. It uses the full rank orthogonal complements $A_k$ and $Z_k$ and is therefore known as the *null space* (or primal Newton) method. To derive the Newton step by this method, start by expressing the step $d_k$ in terms of its unknown projections $d_k^a$ and $d_k^z$ (see Theorem A.1.4):

$$d_k = A d_k^a + Z d_k^z.$$

Substituting for $d_k$ in the bottom row of (1.2.14) and noting that $A_k^T Z_k = \emptyset$ shows that

$$d_k^a = -(A_k^T A_k)^{-1} c_k. \tag{1.2.15}$$

Substituting for $d_k$ and $d_k^a$ in the top row of (1.2.14) produces

$$-W_k A_k (A_k^T A_k)^{-1} c_k + W_k Z_k d_k^z - A_k \lambda_{k+1} = -g_k,$$

which is multiplied by $Z_k$ to obtain the solution for $d_k^z$:

$$d_k^z = (Z_k^T W_k Z_k)^{-1}[Z_k^T W_k A_k (A_k^T A_k)^{-1} c_k - Z_k^T g_k]. \tag{1.2.16}$$

Now $d_k$ is defined by (1.2.15) and (1.2.16), and $\lambda_{k+1}$ is defined in terms of $d_k$, so the null space form of the Newton step is

$$\begin{aligned} d_k &= -A_k(A_k^T A_k)^{-1} c_k \\ &\quad + Z_k (Z_k^T W_k Z_k)^{-1}[Z_k^T W_k A_k (A_k^T A_k)^{-1} c_k - Z_k^T g_k] && (1.2.17) \\ \lambda_{k+1} &= (A_k^T A_k)^{-1}[A_k^T W_k d_k + A_k^T g_k]. && (1.2.18) \end{aligned}$$

Notice that here the "primal" variables $d_k$ are calculated first, and that the $(n-m)\times(n-m)$ matrix $Z_k^T W_k Z_k$ is the projection of $W_k$ onto the null space of $A_k$. This method assumes that both $A_k^T A_k$ and $Z_k^T W_k Z_k$ are nonsingular, but we know from the KKT conditions that this is true at an isolated solution $x_*$. In fact, Lemma 1.2.7 proves that the nonsingularity of $A_k^T A_k$ and $Z_k^T W_k Z_k$ are exactly the conditions necessary to make $K_k$ nonsingular.

Each of these three methods for computing the Newton step has its advantages for particular optimization problems, but our goal is to devise a general purpose algorithm for large-scale problems. The null space method seems the most suitable option for these purposes. The first full space method relies on techniques for inverting the $(n+m)\times(n+m)$ indefinite matrix $K_k$. The range space and null space methods each exploit the structure of $K_k$ and instead invert two smaller matrices. Between these two methods, the null space method is preferred because the two smaller matrices it inverts are both positive definite at a solution, and robust large-scale computational techniques exist for solving such systems. The drawback to the null space method is that it must compute an orthogonal complement $Z_k$, but general purpose large-scale methods exist for this task.

### 1.2.5 Newton-type Method for the KKT Equations

The null space form of the Newton step in (1.2.17)-(1.2.18) computes $d$ and $\lambda_{k+1}$ sequentially, so that $d$ does not depend on $\lambda_{k+1}$. A clever idea for reducing the cost of this method is to approximate the Lagrange multipliers $\lambda_{k+1}$. This will not affect the step $d$ found by (1.2.17), but if some of the constraints are nonlinear then it will change $W_{k+1}$ and hence alter the next Newton step generated in Algorithm 1.2.5. The important property of Algorithm 1.2.5 is its local convergence at a quadratic rate, and it turns out that certain approximations for $\lambda_{k+1}$ preserve this property.

One appropriate approximation drops the term involving $d_k$ in equation (1.2.18), leaving

$$\hat{\lambda}_{k+1} = (A_k^T A_k)^{-1} A_k^T g_k. \tag{1.2.19}$$

As an intuitive justification for this idea, observe that $d_k \to \emptyset$ as Algorithm 1.2.5 converges to $x_*$, and thus $\hat{\lambda}_{k+1} \to \lambda_{k+1}$ as $k \to \infty$. This *least squares* approximation to the Lagrange multipliers seems to have first been tried in the context of a Newton method by Buys and Haarhoff [7]. It has the advantage of making the computations of $d$ and $\hat{\lambda}$ independent from each other. Combining (1.2.19) with the null space method for computing $d_k$ gives us a new algorithm for finding a first-order KKT point. Technically, this algorithm is no longer a Newton method and is instead called a Newton-type method.

**Algorithm 1.2.10** *Newton-type method for solving the first-order KKT equations*

> Start point $x_0$ and stop tolerance $\epsilon > 0$ are given
> LOOP, starting with $k = 0$
> > Compute $c_k$, $g_k$ and $A_k$
> > $\hat{\lambda}_k = (A_k^T A_k)^{-1} A_k^T g_k$
> > Compute $Z_k$ such that $A_k^T Z_k = \emptyset$
> > IF $\|Z_k^T g_k\| < \epsilon$ AND $\|c_k\| < \epsilon$ THEN STOP
> > Compute $W_k$ to be $W(x_k, \hat{\lambda}_k)$
> > Compute $x_{k+1} = x_k + d_k$ using (1.2.17)
>
> CONTINUE, after incrementing $k$

The different stop test in Algorithm 1.2.10 is equivalent to that of Algorithm 1.2.5, and serves to remove the influence of the approximate Lagrange multipliers. A more important difference between the two algorithms is the location of the multiplier computation. In Algorithm 1.2.10 it is at the top of the loop instead of the bottom. This allows $\hat{\lambda}_k$ and $W(x_k, \hat{\lambda}_k)$ to be computed from gradient and Hessian information at the same point $x_k$; otherwise, an estimate of $\hat{\lambda}$ made at the bottom of the loop would be out of step with $W$.

Equation (1.2.19) effectively makes $\hat{\lambda}$ a function of $x$, just like $d$, so Algorithm 1.2.10 is solving for only $n$ unknowns. A more fundamental motivation for a Newton-type algorithm is to go back to the reduced form of the first-order KKT equations and consider developing a Newton method for the system (1.2.2); i.e., solve

$$\begin{bmatrix} Z(x)^T g(x) \\ c(x) \end{bmatrix} = \emptyset \tag{1.2.20}$$

instead of (1.2.7). This system of $n$ nonlinear equations in $n$ unknowns corresponds to a direct elimination of variables using the equality constraints in (EP), and so avoids introducing Lagrange multiplier variables. Unfortunately, Goodman [43] shows that the correct Newton's method for the system (1.2.2) requires computing the Jacobian of the null matrix $Z(x)^T$, which is a tensor quantity containing $O(n^3)$ elements – too many for large-scale applications. However, Algorithm 1.2.10 may be thought of as a Newton-type method for solving (1.2.20) in the sense that near a solution its step becomes the same as the step computed by Newton's method (see [60] for a further discussion). The discrepancy between the two methods is small enough that the quadratic convergence rate of the correct Newton method is preserved in Algorithm 1.2.10, as will now be proved.

Let us write $\hat{W}_k$ for $W(x_k, \hat{\lambda}_k)$. Recognize that Algorithm 1.2.10 uses $\hat{W}_k$ instead of $W_k$, so that at the point $x_k$ it computes the quantities

$$\hat{\lambda}_k = (A_k^T A_k)^{-1} A_k^T g_k \tag{1.2.21}$$

$$\begin{aligned} d_k = {} & -A_k (A_k^T A_k)^{-1} c_k \\ & + Z_k (Z_k^T \hat{W}_k Z_k)^{-1} [Z_k^T \hat{W}_k A_k (A_k^T A_k)^{-1} c_k - Z_k^T g_k]. \end{aligned} \tag{1.2.22}$$

To follow this recipe, the matrices $A_k^T A_k$ and $Z_k^T \hat{W}_k Z_k$ must be nonsingular at every iteration. This will be true close to $x_*$ when $A_*$ is full rank, $\hat{\lambda}(x)$ is continuous, and $x_*$ satisfies the second-order KKT condition. Lemmas 1.2.11, 1.2.12 and 1.2.13 formalize these assertions.

**Lemma 1.2.11** *Assume that all $c^i(x)$ are continuously differentiable and that $A_*$ is full rank. Then there exists some $\delta_1 > 0$ defining a neighborhood $\mathcal{N}_1 = \{x : \|x - x_*\| \leq \delta_1\}$ such that for all $x \in \mathcal{N}_1$ the matrix $A(x)^T A(x)$ is nonsingular and*

$$\|[A(x)^T A(x)]^{-1}\| \leq \beta_{A^T A}.$$

*Proof.* As shown in Lemma 1.2.7, if $A_*$ is full rank then there exists a region of points $\{x : \|x - x_*\| < \epsilon_1\}$ in which $A(x)$ remains full rank; therefore, in this region $A(x)^T A(x)$ is nonsingular and positive definite. Since $A(x)^T A(x)$ is a continuous function, its eigenvalues are continuous functions of $x$ and thus there is a finite region around $x_*$ in which these eigenvalues remain positive. In the terminology of real analysis, for every $\beta_{A^T A}$ there is some $\epsilon_2$ such that $\|[A(x)^T A(x)]^{-1}\| \leq \beta_{A^T A}$ for $\{x : \|x - x_*\| < \epsilon_2\}$. To complete the proof, take $\delta_1 < \min\{\epsilon_1, \epsilon_2\}$. □

**Lemma 1.2.12** *Assume that $f(x)$ and all $c^i(x)$ are twice continuously differentiable, that $A_*$ is full rank, and that $x_*$ is an isolated local solution satisfying the necessary and sufficient KKT conditions (1.2.1) and (1.2.4). Then there exists some $\delta_2 > 0$ defining a neighborhood $\mathcal{N}_2 = \{x : \|x - x_*\| \leq \delta_2\}$ such that for all $x \in \mathcal{N}_2$ the matrix $Z(x)^T \hat{W}(x, \hat{\lambda}(x)) Z(x)$ is nonsingular and*

$$\|[Z(x)^T \hat{W}(x, \hat{\lambda}(x)) Z(x)]^{-1}\| \leq \beta_{Z^T W Z}.$$

*Proof.* As shown in Lemma 1.2.7, if $A_*$ is full rank then there exists a region of points $\{x : \|x - x_*\| < \epsilon_1\}$ in which $Z(x)$ can be chosen to be continuous and orthogonal to $A(x)$. Lemma 1.2.11 and the continuity of $g(x)$ show that $\hat{\lambda}(x)$ is a continuous function for $x \in \mathcal{N}_1$; hence, $\hat{W}(x, \hat{\lambda}(x))$ and $Z(x)$ are continuous in $\mathcal{N}_1$. Since $\hat{\lambda}_* = \lambda_*$, it follows that $\hat{W}_* = W_*$, and from the second-order KKT condition we know that $Z_*^T W_* Z_*$ is positive definite, so $Z_*^T \hat{W}_* Z_*$ is both positive definite and continuous. Now by the continuity of eigenvalues ([62, p. 35]) we may state that for every $\beta_{Z^T W Z}$ there is some $\epsilon_3$ such that $\|[Z(x)^T \hat{W}(x, \hat{\lambda}(x)) Z(x)]^{-1}\|$ is bounded by $\beta_{Z^T W Z}$ for $\{x : \|x - x_*\| < \epsilon_3\}$. Using the definition of $\delta_1$ given in Lemma 1.2.11, choose $\delta_2 < \min\{\delta_1, \epsilon_3\}$ to define $\mathcal{N}_2$. □

**Lemma 1.2.13** *Let $D$ be a closed bounded region in $\mathbf{R}^n$ containing $x_*$. Assume that for every $x \in D$ the functions $f(x)$ and $c^i(x)$ are twice continuously differentiable, that their Hessians satisfy Lipschitz conditions, and that $g(x)$ and $A(x)$ are bounded. Also assume that $A_*$ is full rank. Then for every $x_1, x_2 \in \{D \cap \mathcal{N}_1\}$,*

$$\|\hat{W}(x_1, \hat{\lambda}(x_1))\| \leq \hat{\beta}_W \quad \textit{and} \quad \|\hat{W}(x_1, \hat{\lambda}(x_1)) - \hat{W}(x_2, \hat{\lambda}(x_2))\| \leq \hat{\gamma}_W \|x_1 - x_2\|,$$

*where $\mathcal{N}_1$ is the neighborhood defined in Lemma 1.2.11.*

*Proof.* First, let us list all the bounds that follow directly from the hypotheses of this lemma. When $x_1$ and $x_2$ are in $\{D \cap \mathcal{N}_1\}$, there exist finite constants satisfying

$$\begin{array}{rcll}
\|\nabla^2 f(x_1) - \nabla^2 f(x_2)\| & \leq & \gamma_f \|x_1 - x_2\| & \\
\|\nabla^2 f(x)\| & \leq & \beta_f & \\
\|g(x_1) - g(x_2)\| & \leq & \gamma_g \|x_1 - x_2\| & \\
\|g(x)\| & \leq & \beta_g & \\
\|\nabla^2 c^i(x_1) - \nabla^2 c^i(x_2)\| & \leq & \gamma_c \|x_1 - x_2\|, & \text{for} \quad i = 1, \ldots, m \\
\|\nabla^2 c^i(x)\| & \leq & \beta_c, & \text{for} \quad i = 1, \ldots, m \\
\|A(x_1) - A(x_2)\| & \leq & \gamma_A \|x_1 - x_2\| & \\
\|A(x_1)^T - A(x_2)^T\| & \leq & \gamma_{A^T} \|x_1 - x_2\| & \\
\|A(x)\| & \leq & \beta_A & \\
\|A(x)^T\| & \leq & \beta_{A^T}. &
\end{array}$$

The bounds involving $\beta_f$ and $\beta_c$ are a result of the Hessians being continuous over a closed bounded region. The Lipschitz conditions on $g$ and $A$ then follow from the boundedness of the Hessians and the continuity of $g$ and $A$. Note that the pairs of constants $\gamma_A$, $\gamma_{A^T}$ and $\beta_A$, $\beta_{A^T}$ are each related by an appropriate vector norm conversion constant.

From these bounds and Lemma 1.2.11 we see immediately that

$$\begin{aligned}
\|\hat{\lambda}(x)\| &= \|[A(x)^T A(x)]^{-1} A(x)^T g(x)\| \\
&\leq \beta_{A^T A} \beta_{A^T} \beta_g. 
\end{aligned} \tag{1.2.23}$$

From the definition of $\hat{W}$ we obtain the bound

$$\begin{aligned}
\|\hat{W}(x, \hat{\lambda}(x))\| &= \|\nabla^2 f(x) - \sum_{i=1}^{m} \hat{\lambda}^i(x) \nabla^2 c^i(x)\| \\
&\leq \beta_f + \beta_c \sum_{i=1}^{m} |\hat{\lambda}^i(x)|.
\end{aligned}$$

The sum in the last term is simply $\|\hat{\lambda}(x)\|_1$. If $c_v$ denotes the equivalence constant between the $\ell_1$ norm and any other vector norm, then the first part of the lemma is proved by

$$\|\hat{W}(x, \hat{\lambda}(x))\| \leq \hat{\beta}_W, \quad \text{where} \quad \hat{\beta}_W = \beta_f + c_v \beta_c \beta_{A^T A} \beta_{A^T} \beta_g. \tag{1.2.24}$$

To prove the Lipschitz bound on $\hat{W}(x, \hat{\lambda}(x))$, we first need a Lipschitz bound on $\hat{\lambda}(x)$. To make the notation easier, let us use the subscripts 1 and 2 for quantities evaluated at $x_1$ and $x_2$. Start from the definition of $\hat{\lambda}$ given in (1.2.21) to obtain

$$\begin{aligned}
\hat{\lambda}_1 - \hat{\lambda}_2 &= (A_1^T A_1)^{-1} A_1^T g_1 - (A_2^T A_2)^{-1} A_2^T g_2 \\
&= (A_1^T A_1)^{-1} (A_1^T - A_2^T) g_1 \\
&\quad + (A_1^T A_1)^{-1} A_2^T (g_1 - g_2) + [(A_1^T A_1)^{-1} - (A_2^T A_2)^{-1}] A_2^T g_2.
\end{aligned}$$

Taking the norm of both sides and using Lemma 1.2.11 plus the inequalities listed at the beginning of this proof, we have that

$$\begin{aligned}\|\hat\lambda_1-\hat\lambda_2\| &\le \beta_{A^TA}(\beta_g\gamma_{A^T}+\beta_{A^T}\gamma_g)\|x_1-x_2\| \\ &\quad +\beta_{A^T}\beta_g\|(A_1^TA_1)^{-1}-(A_2^TA_2)^{-1}\|.\end{aligned}$$

Now rewrite the last term as

$$\begin{aligned}(A_1^TA_1)^{-1}-(A_2^TA_2)^{-1} &= (A_1^TA_1)^{-1}[A_2^TA_2-A_1^TA_1](A_2^TA_2)^{-1} \\ &= (A_1^TA_1)^{-1}[A_2^T(A_2-A_1)+(A_2-A_1)^TA_1](A_2^TA_2)^{-1}.\end{aligned}$$

The norm of this satisfies

$$\|(A_1^TA_1)^{-1}-(A_2^TA_2)^{-1}\| \le \beta_{A^TA}^2(\beta_{A^T}\gamma_A+\beta_A\gamma_{A^T})\|x_1-x_2\|.$$

Let us combine all the constants and define

$$\hat\gamma_\lambda=\beta_{A^TA}(\beta_g\gamma_{A^T}+\beta_{A^T}\gamma_g)+\beta_{A^T}\beta_g\beta_{A^TA}^2(\beta_{A^T}\gamma_A+\beta_A\gamma_{A^T}),$$

so that $\|\hat\lambda_1-\hat\lambda_2\|\le\hat\gamma_\lambda\|x_1-x_2\|$.

Now we are ready to prove the Lipschitz bound on $\hat W(x,\hat\lambda(x))$. Plug in the definition of $\hat W$, take the norm of both sides and apply our bounds to obtain

$$\begin{aligned}\|\hat W_1-\hat W_2\| &= \|(\nabla^2 f_1-\sum_{i=1}^m\hat\lambda_1^i\nabla^2c_1^i)-(\nabla^2 f_2-\sum_{i=1}^m\hat\lambda_2^i\nabla^2c_2^i)\| \\ &\le \|\nabla^2f_1-\nabla^2f_2\|+\|\sum_{i=1}^m(\hat\lambda_1^i-\hat\lambda_2^i)\nabla^2c_1^i\|+\|\sum_{i=1}^m\hat\lambda_2^i(\nabla^2c_1^i-\nabla^2c_2^i)\| \\ &\le \gamma_f\|x_1-x_2\|+\beta_c\sum_{i=1}^m|\hat\lambda_1^i-\hat\lambda_2^i|+\gamma_c\|x_1-x_2\|\sum_{i=1}^m|\hat\lambda_2^i|.\end{aligned}$$

As before, we may replace the terms containing a sum of components $|\hat\lambda^i|$ with $\|\hat\lambda\|_1=c_v\|\hat\lambda\|$. Then the Lipschitz condition on $\hat\lambda(x)$ and the absolute bound in (1.2.23) can be applied to yield

$$\begin{aligned}\|\hat W_1-\hat W_2\| &\le \gamma_f\|x_1-x_2\|+c_v\beta_c\|\hat\lambda_1-\hat\lambda_2\|+c_v\gamma_c\|\hat\lambda_2\|\,\|x_1-x_2\| \\ &\le (\gamma_f+c_v\beta_c\hat\gamma_\lambda+c_v\gamma_c\beta_{A^TA}\beta_{A^T}\beta_g)\|x_1-x_2\|.\end{aligned}$$

Therefore, the second part of the lemma is proved with

$$\hat\gamma_W=\gamma_f+c_v\beta_c\hat\gamma_\lambda+c_v\gamma_c\beta_{A^TA}\beta_{A^T}\beta_g.$$

□

With lemmas 1.2.11, 1.2.12 and 1.2.13 it is now possible to prove that the Newton-type method of Algorithm 1.2.10 is quadratically convergent. The assumptions needed for this proof are conceptually the same as those required to prove the quadratic convergence of Newton's method in Theorem 1.2.9 – the proof is again based on bounding Taylor series approximations of continuous functions.

**Theorem 1.2.14** *Assume that $f(x)$ and all $c^i(x)$ are twice continuously differentiable, that their Hessians satisfy Lipschitz conditions, and that $g(x)$ and $A(x)$ are bounded. Let $x_*$ be an isolated local solution to (EP) satisfying the necessary and sufficient KKT conditions (1.2.1) and (1.2.4), and assume that $A_*$ is full rank. Then if the start point $x_0$ is sufficiently close to $x_*$, Algorithm 1.2.10 converges to $x_*$, and the rate of convergence is Q-quadratic.*

*Proof.* Assume for now that an iterate $x_k$ is close enough to $x_*$ for Lemmas 1.2.11, 1.2.12 and 1.2.13 to apply, and consider the next iterate $x_{k+1} = x_k + d_k$ computed by Algorithm 1.2.10 using (1.2.22). Let us denote the errors in these two iterates by $e_k = x_k - x_*$ and $e_{k+1} = x_{k+1} - x_*$. This proof will show that $\|e_{k+1}\| < \|e_k\|$ by bounding the projections of $e_{k+1}$ onto the column spaces of orthogonal complements $A_k$ and $Z_k$. Start by using Theorem A.1.4 and write $e_{k+1}$ in terms of its unknown projections $e^a_{k+1}$ and $e^z_{k+1}$:

$$e_{k+1} = A_k e^a_{k+1} + Z_k e^z_{k+1}. \tag{1.2.25}$$

Now make a Taylor series expansion for $c(x)$ at $x_*$ in terms of values at $x_k$ (refer to (A.3.2) in the Appendix):

$$c_* = c_k + \int_0^1 A(x_k + t(x_* - x_k))^T (x_* - x_k)\, dt.$$

From equation (1.2.22) we see that $A_k^T d_k = A_k^T(x_{k+1} - x_k) = -c_k$, and since $c_* = \emptyset$ we have that

$$A_k^T(x_{k+1} - x_*) = \int_0^1 [A(x_k + t(x_* - x_k))^T - A_k^T](x_* - x_k)\, dt.$$

Substituting (1.2.25) for $x_{k+1} - x_*$ gives an expression for $e^a_{k+1}$, and then taking the norm of both sides and using the bounds of Lemmas 1.2.11 and 1.2.13 we obtain

$$\begin{aligned} e^a_{k+1} &= (A_k^T A_k)^{-1} \int_0^1 [A(x_k + t(x_* - x_k))^T - A_k^T](x_* - x_k)\, dt \\ \|e^a_{k+1}\| &\le \beta_{A^TA} \|x_* - x_k\| \int_0^1 \|A(x_k + t(x_* - x_k))^T - A_k^T\|\, dt \\ &\le \beta_{A^TA}\gamma_{A^T} \|x_k - x_*\|^2. \end{aligned} \tag{1.2.26}$$

Next, we seek a bound on $e^z_{k+1}$. As explained in Lemma 1.2.12, $\hat{\lambda}(x)$ is a continuous function in the neighborhood of $x_*$, which includes the points on a line between $x_*$ and our hypothetical iterate $x_k$. Let us then make a Taylor series expansion for the other first-order KKT residual at $x_*$ in terms of values at $x_k$:

$$g_* - A_*\hat{\lambda}_* = g_k - A_k\hat{\lambda}_k + \int_0^1 [\nabla^2 f(a(t)) - J\{A(a(t))\hat{\lambda}(a(t))\}]\,(x_* - x_k)\, dt,$$

where $a(t) = x_k + t(x_* - x_k)$ and $J\{A(x)\hat{\lambda}(x)\}$ stands for the Jacobian of $A(x)\hat{\lambda}(x)$ with respect to $x$. This Jacobian works out to be

$$J\{A(x)\hat{\lambda}(x)\} = A(x)J\{\hat{\lambda(x)}\} + \sum_{i=1}^{m} \hat{\lambda}^i(x)\nabla^2 c^i(x),$$

where $J\{\hat{\lambda}(x)\}$ is just the Jacobian of $\hat{\lambda}(x)$. Substituting in, the Taylor series becomes

$$-(g_k - A_k\hat{\lambda}_k) = \int_0^1 [\hat{W}(a(t), \hat{\lambda}(a(t))) - A(a(t))J\{\hat{\lambda}(a(t))\}](x_* - x_k)\, dt.$$

Multiply this by $Z_k^T$ to remove a term on the left, obtaining

$$-Z_k^T g_k = \int_0^1 Z_k^T[\hat{W}(a(t), \hat{\lambda}(a(t))) - A(a(t))J\{\hat{\lambda}(a(t))\}](x_* - x_k)\, dt.$$

Referring back to equation (1.2.22), observe that $Z_k^T\hat{W}_k d_k = -Z_k^T g_k$, or

$$-Z_k^T g_k = Z_k^T\hat{W}_k(x_{k+1} - x_*) + Z_k^T\hat{W}_k(x_* - x_k).$$

The two equations above then combine into

$$Z_k^T\hat{W}_k e_{k+1} = \int_0^1 Z_k^T[\hat{W}(a(t), \hat{\lambda}(a(t))) - \hat{W}_k - A(a(t))J\{\hat{\lambda}(a(t))\}](x_* - x_k)\, dt.$$

Since $Z_k^T A_k = \emptyset$, we may introduce an artificial term of this form into the integral to set things up for the application of a Lipschitz condition. Let us also substitute for $e_{k+1}$ from (1.2.25), obtaining

$$\begin{aligned} Z_k^T\hat{W}_k Z_k e_{k+1}^z &= -Z_k^T\hat{W}_k A_k e_{k+1}^a \\ &\quad + \int_0^1 Z_k^T[\hat{W}(a(t), \hat{\lambda}(a(t))) - \hat{W}_k](x_* - x_k)\, dt \\ &\quad - \int_0^1 Z_k^T[A(a(t)) - A_k]J\{\hat{\lambda}(a(t))\}(x_* - x_k)\, dt. \end{aligned}$$

By Lemma 1.2.12, $Z_k^T\hat{W}_k Z_k$ is nonsingular, so we can multiply both sides by its inverse and take norms to bound $e_{k+1}^z$:

$$\begin{aligned} \|e_{k+1}^z\| &\leq \beta_{Z^TWZ}\hat{\beta}_W\beta_A\|Z_k^T\|\,\|e_{k+1}^a\| \\ &\quad + \beta_{Z^TWZ}\|Z_k^T\|\,\|x_k - x_*\| \int_0^1 \|\hat{W}(a(t), \hat{\lambda}(a(t))) - \hat{W}_k\|\, dt \\ &\quad + \beta_{Z^TWZ}\|Z_k^T\|\,\|x_k - x_*\| \int_0^1 \|J\{\hat{\lambda}(a(t))\}\|\,\|A(a(t)) - A_k\|\, dt. \end{aligned}$$

Recall from Lemmas 1.2.12 and 1.2.7 that $Z_k$ can be defined from $A_k$ by the method of direct elimination, and therefore it is bounded so long as $\|A_k\|$ and $\|(A_k^T A_k)^{-1}\|$ are bounded ($Z_k$ can also be defined by other methods, such as QR-factorization, and similarly bounded). The Jacobian of $\hat{\lambda}(x)$ can be expressed (after some unenlightening algebraic manipulations) in terms of $[A(x)^T A(x)]^{-1}$ and the first and second derivatives of $f(x)$ and $c(x)$, all of which are bounded in the proof of Lemma 1.2.13. Hence, we may infer the existence of three more inequalities for points sufficiently close to $x_*$ (that is, close in the same sense as in Lemmas 1.2.11, 1.2.12 and 1.2.13):

$$\begin{aligned} \|Z(x)\| &\leq \beta_Z \\ \|Z(x)^T\| &\leq \beta_{Z^T} \\ \|J\{\hat{\lambda}(x)\}\| &\leq \hat{\beta}_J. \end{aligned}$$

Using these three relations, Lemma 1.2.13 and (1.2.26) we see that

$$\|e^z_{k+1}\| \le \beta_{Z^TWZ}\beta_{Z^T}(\hat{\beta}_W\beta_A\beta_{A^TA}\gamma_{A^T} + \hat{\gamma}_W + \hat{\beta}_J\gamma_A)\|x_k - x_*\|^2.$$

Returning to (1.2.25), we conclude that

$$\begin{aligned}\|e_{k+1}\| &\le \|A_k\|\,\|e^a_{k+1}\| + \|Z_k\|\,\|e^z_{k+1}\| \\ &\le \beta\,\|e_k\|^2,\end{aligned} \tag{1.2.27}$$

where

$$\beta \equiv \beta_A\beta_{A^TA}\gamma_{A^T} + \beta_Z\beta_{Z^TWZ}\beta_{Z^T}(\hat{\beta}_W\beta_A\beta_{A^TA}\gamma_{A^T} + \hat{\gamma}_W + \hat{\beta}_J\gamma_A).$$

Now turn to Algorithm 1.2.10 and assume that the start point $x_0$ satisfies

$$\|x_0 - x_*\| \le \delta_3, \quad \text{where} \quad \delta_3 \equiv \tau\min\{\frac{1}{\beta}, \delta_2\}, \quad \tau \in (0,1),$$

and $\delta_2$ defines $\mathcal{N}_2$ in Lemma 1.2.12. Then $x_0$ meets all the requirements for Lemmas 1.2.11, 1.2.12 and 1.2.13, the next iterate $x_1$ is well-defined by (1.2.22), and it satisfies

$$\|x_1 - x_*\| \;\le\; \beta\,\|x_0 - x_*\| \;\le\; \tau\|x_0 - x_*\|.$$

Thus, the iterate $x_1$ is closer to the solution $x_*$; furthermore, it is again sufficiently close for all pertinent lemmas and bounds to hold, and so the next step of Algorithm 1.2.10 is well-defined. By induction, all subsequent iterates $x_k$ generated by Algorithm 1.2.10 are well-defined, each moving closer to the solution by at least a factor $\tau$. Therefore,

$$\|x_k - x_*\| \le \tau^k\|x_0 - x_*\|, \quad \text{and} \quad \lim_{k\to\infty}\|x_k - x_*\| = 0.$$

This proves that Algorithm 1.2.10 converges, and the relation (1.2.27) shows that the rate of convergence is quadratic. □

Theorem 1.2.14 proves that the quadratic rate of convergence of Newton's method is not impeded when the Lagrange multipliers are approximated using equation (1.2.21). The proof of Theorem 1.2.14 is written in a general form that allows us to identify the properties of the approximation that are essential for retaining quadratic convergence. A careful examination shows that *any* Lagrange multiplier approximation $\tilde{\lambda}(x)$ used in Algorithm 1.2.10 will have a quadratic rate of convergence provided that $\tilde{\lambda}(x)$ has the following properties:

1. $\tilde{\lambda}(x)$ is continuous in the neighborhood of a solution $x_*$.

2. $\tilde{\lambda}_* = \lambda_*$.

3. The Jacobian of $\tilde{\lambda}(x)$ is bounded in the neighborhood of $x_*$.

4. The two bounds of Lemma 1.2.13 on $\tilde{W}(x, \tilde{\lambda}(x))$ hold.

Other Lagrange multiplier approximations will be investigated later in this dissertation, but the estimate (1.2.21) is the most useful one.

### 1.2.6 First-Order Methods Viewed as Optimizers

The two previous sections have discussed a Newton (Algorithm 1.2.5) and Newton-type (Algorithm 1.2.10) method for iteratively solving the system of first-order KKT conditions (1.2.7). The methods are both seen to have excellent convergence properties near an isolated solution point; however, a vital assumption is made that all iterates $x_k$ satisfy the second-order KKT condition (i.e., $Z_k^T W_k Z_k$ is presumed positive definite). That assumption is in general false far from a solution, and will be relaxed later in §1.3. In preparation for that improvement, a different view of the solution found by Algorithms 1.2.5 and 1.2.10 needs to be developed.

Consider the problem

$$\min_{d \in \mathbf{R}^n} d^T g_k + \frac{1}{2} d^T W_k d \qquad \text{subject to} \quad A_k^T d + c_k = \emptyset, \tag{1.2.28}$$

where quantities are defined with respect to some particular $x_k$ and $\lambda_k$ (or $\hat{\lambda}_k$). The condition of $Z_k^T W_k Z_k$ being positive definite is important for this problem also.

**Theorem 1.2.15** *If $A_k$ is full rank and $Z_k^T W_k Z_k$ is positive definite, then the problem (1.2.28) has a unique solution $d_*$.*

*Proof.* Because $A_k$ is full rank, the linear constraints are consistent and a linear manifold of feasible points exists. A set of $n - m$ basis vectors spanning this manifold can be arranged as the columns of a full rank $n \times (n - m)$ matrix $Z_k$, so that $A_k^T Z_k = \emptyset$. The matrices $A_k$ and $Z_k$ are orthogonal complements, allowing $d_*$ to be represented in terms of the unique projections $d_*^a$ and $d_*^z$:

$$d_* = A_k d_*^a + Z_k d_*^z.$$

Substituting into the constraint of (1.2.28) and noting that $A_k^T A_k$ is nonsingular, we have that

$$d_*^a = -(A_k^T A_k)^{-1} c_k.$$

Using $d = A_k d_*^a + Z_k d^z$ in (1.2.28) eliminates the constraints, leaving us with an unconstrained minimization problem:

$$\min_{d^z \in \mathbf{R}^{n-m}} [A_k d_*^a + Z_k d^z]^T g_k + \frac{1}{2}[A_k d_*^a + Z_k d^z]^T W_k [A_k d_*^a + Z_k d^z]. \tag{1.2.29}$$

The Hessian with respect to $d^z$ of this problem is simply $Z_k^T W_k Z_k$, which is positive definite by hypothesis; therefore, (1.2.29) has a unique solution $d_*^z$, and $d_*$ is unique. □

Let us finish solving the unconstrained subproblem of Theorem 1.2.15 to see what $d_*$ is. The unique minimizer $d_*^z$ of (1.2.29) is found by setting the gradient of that objective to zero. This gives

$$d_*^z = -(Z_k^T W_k Z_k)^{-1} [Z_k^T W_k A_k d_*^a + Z_k^T g_k].$$

Therefore, when (1.2.28) has a unique solution, it is

$$d_* = -A_k(A_k^T A_k)^{-1} c_k + Z_k(Z_k^T W_k Z_k)^{-1}[Z_k^T W_k A_k(A_k^T A_k)^{-1} c_k - Z_k^T g_k].$$

This should look familiar because it is exactly the same as the null space form of the Newton step in equation (1.2.17). In other words, the solution to (1.2.28) is the same as the step $d_k$ computed in Algorithms 1.2.5 and 1.2.10, assuming that $A_k$ is full rank and $Z_k^T W_k Z_k$ is positive definite. This means we can replace the computation of $d_k$ by (1.2.17) with the subproblem (1.2.28) and obtain a new algorithm. Let us make this substitution in our Newton-type algorithm.

**Algorithm 1.2.16** *Locally quadratic iterative method for solving (EP)*

Start point $x_0$ and stop tolerance $\epsilon > 0$ are given
LOOP, starting with $k = 0$
  Compute $c_k$, $g_k$ and $A_k$
  $\hat{\lambda}_k = (A_k^T A_k)^{-1} A_k^T g_k$
  Compute $Z_k$ such that $A_k^T Z_k = \emptyset$
  IF $\|Z_k^T g_k\| < \epsilon$ AND $\|c_k\| < \epsilon$ THEN STOP
  Compute $W_k$ to be $W(x_k, \hat{\lambda}_k)$
  Obtain $d_k$ by solving the subproblem
    $\min \;\; d^T g_k + \frac{1}{2} d^T W_k d \quad$ subject to $\quad A_k^T d + c_k = \emptyset$
  $x_{k+1} = x_k + d_k$
CONTINUE, after incrementing $k$

Near a solution, Theorem 1.2.15 proves that this algorithm is equivalent to Algorithm 1.2.10 and thus shares its quadratic rate of convergence. The advantage of Algorithm 1.2.16 is that it provides a more general and meaningful framework for computing iterates far from a solution, where equation (1.2.17) may be inappropriate. The subproblem (1.2.28) solved at each iteration can be interpreted as a local model at the iterate $x_k$ of the general nonlinear equality constrained problem (EP). The model contains all gradient and Hessian information of the nonlinear objective and constraints. Close to $x_*$ the model is accurate enough to allow a quadratic rate of convergence. Far from $x_*$ where $Z_k^T W_k Z_k$ is not positive definite, a meaningful solution to the model subproblem can be defined with respect to the real problem, as we shall see in §1.3. Also note that Algorithm 1.2.16 can be made independent of the matrix $Z_k$ by using the equivalent stop test $\|g_k - A_k \hat{\lambda}_k\| < \epsilon$.

Some authors prefer the model subproblem

$$\min_{d \in \mathbf{R}^n} \; d^T(g_k - A_k \lambda_k) + \frac{1}{2} d^T W_k d \qquad \text{subject to} \quad A_k^T d + c_k = \emptyset \tag{1.2.30}$$

to (1.2.28), where $\lambda_k$ may be the proper Lagrange multipliers of (1.2.7) or an estimate such as (1.2.21). Note that the objective in (1.2.30) can be expressed conveniently in terms of the Lagrangian: $d^T \nabla_x^2 \mathcal{L}_k + \frac{1}{2} d^T \nabla_{xx}^2 \mathcal{L}_k d$. It is easy to see that the proof of Theorem 1.2.15 works for (1.2.30), and that both $d_*^a$ and $d_*^z$ are unchanged (the new objective only alters

(1.2.29) by the constant term $-\lambda_k^T A_k^T A_k d_*^a$). Hence, the subproblem (1.2.30) can be used in place of (1.2.28) without altering the local convergence properties of Algorithm 1.2.16; however, when $x_k$ is far from a solution the two subproblems behave differently.

### 1.2.7 Summary of Section

In this section the local properties of solutions to problem (EP) were derived and used to construct locally convergent algorithms. Local solutions are characterized by the first- and second-order KKT conditions, which were shown to be both necessary and sufficient conditions for an isolated local solution. By defining artificial Lagrange multiplier variables and applying Newton's method to the nonlinear system of equations resulting from the first-order KKT conditions, Algorithm 1.2.5 was obtained. This algorithm was shown to be convergent at a quadratic rate when started close to an isolated local solution. Algorithm 1.2.10 was constructed after deciding on the null space method for computing the Newton step and devising a simplified formula for estimating the Lagrange multipliers. Finally, Algorithm 1.2.10 was recast into the form of Algorithm 1.2.16, which iteratively solves equality constrained model subproblems.

All the theorems of algorithm convergence assume that the local solution set is an isolated point. This need not be the case, as the second-order KKT condition (1.2.4) does not have to be a strict inequality at a solution. In most instances, Algorithms 1.2.5, 1.2.10 and 1.2.16 can still theoretically converge to a point in the local solution set, although the convergence rate may be slower than quadratic. In practice, the matrices $Z_k^T W_k Z_k$ become badly ill-conditioned and computational roundoff errors may prevent convergence to an exact solution. All of our theorems also make the regularity assumption that $A_*$ is full rank – this is more difficult to dispense with and still maintain a practical algorithm.

The primary result of this section is Algorithm 1.2.16. In the next section this algorithm will be extended to allow iterations from any point, not just points close to a solution. The model subproblem form anticipates this extension, but it is important to remember that an extended algorithm should reduce to the Newton-type method of Algorithm 1.2.10 close to a solution, so that it enjoys the benefits of a quadratic rate of convergence and low computation cost.

## 1.3 Globalization with Trust Regions

The purpose of this section is to extend Algorithm 1.2.16 so that it converges to a local solution $x_*$ of (EP) from *any* starting point $x_0$; i.e., *globalize* the convergence of Algorithm 1.2.16. There are two classical techniques for globalizing a locally convergent algorithm: use a line search or impose a trust region. This dissertation focuses on trust region methods, in part because of their excellent success in globalizing algorithms for solving unconstrained optimization problems and nonlinear systems of equations, and in part because much more research effort has already gone into the exploration of line search methods. The principle concern here is to devise a trust region method that executes efficiently for large problems. This section includes a review of the various trust region methods proposed for solving equality constrained problems, and develops one preferred algorithm in greater detail.

### 1.3.1 Trust Region Basics

The model subproblem for equality constrained optimization (1.2.28) is formulated to be a meaningful simplification of the actual nonlinear constrained problem (EP). Theorem 1.2.14 proves that the solution $d$ to the model subproblem is meaningful and appropriate when $x_k$ is sufficiently close to $x_*$; however, when the current point $x_k$ is far from $x_*$, then the solution $d$ does not always make sense. For example, if the reduced Hessian matrix $Z_k^T W_k Z_k$ in (1.2.28) is not positive definite, then there are feasible directions along which the model objective is unbounded from below; hence, $\|d\|$ is infinite. A trust region is simply a restriction on the size of $\|d\|$, whose purpose is to limit the subproblem to a region in which it accurately models the actual problem.

The method of trust regions has been used with great success to globalize Newton and quasi-Newton algorithms for solving unconstrained optimization problems (refer to [20] or any standard reference). The method guarantees convergence to some local minimum from any starting point and preserves the asymptotic convergence rate of the local algorithm. But there are certain difficulties in extending trust region concepts to nonlinear equality constrained problems. To understand these better, let us first review the fundamental principles of trust region methods in the context of unconstrained optimization.

If the unconstrained problem is to minimize a smooth objective $f(x)$, then a Taylor series expansion about the point $x_k$ gives the approximation

$$f(x_k + d) \cong f(x_k) + d^T \nabla f(x_k) + \frac{1}{2} d^T \nabla^2 f(x_k)\, d,$$

which provides a local quadratic model of $f(x)$. The first-order necessary condition for $x_*$ to be a minimizer of $f(x)$ is that $\nabla f(x_*) = \emptyset$. This is a system of equations that can be solved by a Newton method, which takes iterative steps of the form $-[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$. Now in the neighborhood of a minimizer the solution of the model quadratic subproblem

$$\min_{d \in \mathbf{R}^n} \; d^T \nabla f(x_k) + \frac{1}{2} d^T \nabla^2 f(x_k)\, d \tag{1.3.1}$$

is the same as the Newton step at $x_k$; hence, an algorithm that solves (1.3.1) and then takes the step $x_{k+1} = x_k + d_k$ will converge to $x_*$ at a quadratic rate, assuming the algorithm starts from a point sufficiently close to $x_*$. This locally convergent algorithm may not work if $x_k$ is far from a minimizer because the solution of (1.3.1) is not always reasonable (e.g., $\nabla^2 f(x_k)$ may be indefinite). To globalize the convergence of the algorithm, a trust region constraint is added to regulate $\|d\|$ in a specific manner. An example of such a method is shown in the algorithm below.

**Algorithm 1.3.1** *Trust region method for minimizing $f(x)$*

Constants $\epsilon > 0$ and $\eta, \gamma \in (0, 1)$ are given
Choose any $x_0$ and $\Delta_0 > 0$
LOOP, starting with $k = 0$
  Compute $g_k = \nabla f(x_k)$ and $H_k = \nabla^2 f(x_k)$
  IF $\|g_k\| < \epsilon$ THEN STOP
  Obtain $d_k$ by solving the subproblem

$$\min \; d^T g_k + \tfrac{1}{2} d^T H_k d \quad \text{subject to} \quad \|d\| \leq \Delta_k \tag{1.3.2}$$

  Compute a_red $= f(x_k) - f(x_k + d_k)$ and p_red $= -d_k^T g_k - \frac{1}{2} d_k^T H_k d_k$
  IF $\dfrac{\text{a\_red}}{\text{p\_red}} \geq \eta$
    THEN $x_{k+1} = x_k + d_k$, $\Delta_{k+1} \geq \Delta_k$
    ELSE $x_{k+1} = x_k$, $\Delta_{k+1} \leq \gamma \|d_k\|$
CONTINUE, after incrementing $k$

Algorithm 1.3.1 computes new iterates by solving the constrained quadratic subproblem (1.3.2). The quantity a_red is the actual reduction in $f(x)$ produced by the trial step $d_k$, which should be positive for a reasonable step. The quantity p_red is the reduction achieved by $d_k$ in the model objective, and it will be positive if $d_k$ is an accurate solution of the subproblem (1.3.2). Now p_red represents a prediction of a_red based on the quadratic model of $f(x)$, and the final IF statement in Algorithm 1.3.1 tests whether the actual reduction is sufficiently close to the predicted reduction. If it is, then the trial step $d_k$ is accepted and used to compute the next iterate $x_{k+1}$. If not, then it must be that (1.3.2) cannot be trusted as a good model of $f(x)$ to the distance $\|d_k\|$, and so the subproblem is reconsidered with a smaller trust region constraint. As $\Delta_k$ becomes smaller the quadratic model (1.3.1) becomes a better approximation of $f(x)$, and eventually some finite step $d_k$ will be accepted by Algorithm 1.3.1. This is the fundamental idea underlying the global convergence of trust region methods.

The global convergence property can be pushed to impressive extremes that are particularly attractive for solving large-scale problems. The exact solution of (1.3.2) requires inverting the large matrix $H_k$, but the strong convergence properties of the trust region method can be used to streamline this computationally expensive operation. For instance, it can be shown [71] that convergence to a first-order point is still obtained even when $d_k$ is only an approximate solution to (1.3.2), and even if the subproblem is altered by using some choice of $H_k$ different from $\nabla^2 f(x_k)$. Hence, (1.3.2) can be solved approximately by a fast iterative solver, and an efficiently stored, easily inverted matrix $H_k$ can be used in place of $\nabla^2 f(x_k)$. It turns out that similar strong convergence properties hold when trust region methods are applied to problems with nonlinear equality constraints, and the same approximation ideas can be exploited for solving large-scale problems.

However, there are a few complications in extending the trust region method to globalize Algorithm 1.2.16. The method requires two main modifications to a local method, both illustrated by Algorithm 1.3.1: the trust region constraint must be added to the model subproblem, and the step that solves the constrained model subproblem must be evaluated for how much progress it makes in solving the actual problem. When nonlinear equality constraints are thrown in, both of these modifications become more complicated than in the unconstrained case.

The second modification, evaluating the step, is more difficult because the step must try to reduce the actual objective $f(x)$ and satisfy the equality constraints $c(x)$; these are sometimes contradictory goals. The solution is to invent a single *merit function* for judging the value of a step. The function must account for both goals in a manner that guarantees global convergence of the trust region algorithm. Merit functions are a topic in themselves, and further discussion is deferred until §1.5.6.

Now consider the first modification, adding a trust region constraint to the model subproblem. If we simply add it onto the quadratic subproblem in Algorithm 1.2.16, then we obtain the iterative trust region subproblem:

$$\min_{d \in \mathbf{R}^n} \quad d^T g_k + \frac{1}{2} d^T W_k d \tag{1.3.3}$$

$$\text{subject to} \quad A_k^T d + c_k = \emptyset \tag{1.3.4}$$

$$\|d\| \leq \Delta_k. \tag{1.3.5}$$

Notice that if $x_k$ is far from satisfying one or more of the linearized constraints in (1.3.4) and $\Delta_k$ is small, then the set of constraints (1.3.4)-(1.3.5) may be inconsistent; i.e., there may be no feasible points in the subproblem. This is illustrated below in Figure 1.1, a problem in two unknowns with one linear constraint. Clearly, there is no step from $x_k$ that stays within the circular trust region constraint and satisfies the linear equality constraint. As noted earlier, the global convergence of the trust region method depends on being able

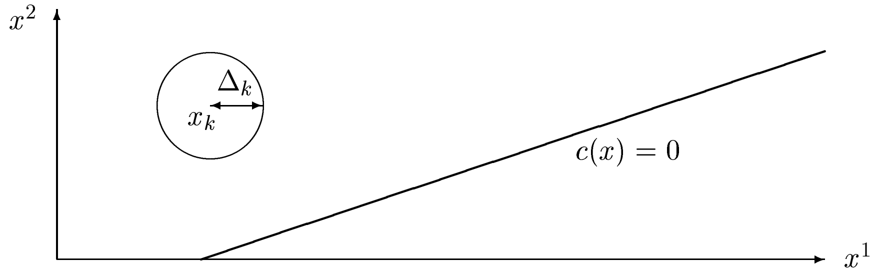


Figure 1.1: Example of inconsistent constraints

to reduce $\Delta_k$ until the model subproblem accurately represents the actual problem; hence, inconsistent constraints are unavoidable in the formulation (1.3.3)-(1.3.5).

### 1.3.2 Making the Constraints Consistent

Many different ways have been proposed for applying a trust region constraint in a manner that does not generate inconsistent subproblems. Let us briefly survey these ideas and evaluate them for their effectiveness in solving large problems.

One general approach is to put the linearized equality constraints (1.3.4) into the objective (1.3.3) in the form of a penalty term, leaving only the trust region (1.3.5) as a constraint. Fletcher proposed in [32] to use a quadratic model of the exact $\ell_1$ penalty function, thus replacing (1.3.3)-(1.3.5) with the subproblem

$$\begin{array}{ll} \min\limits_{d \in \mathbf{R}^n} & \nu_k(d^T g_k + \frac{1}{2} d^T W_k d) + \|A_k^T d + c_k\|_1 \\ \text{subject to} & \|d\|_\infty \leq \Delta_k. \end{array}$$

The scalar $\nu_k$ is a penalty parameter, chosen finite and positive to balance the two goals of simultaneously reducing the model objective function and satisfying the linearized equality constraints (the same two goals mentioned in connection with merit functions). Fletcher calls the resulting iterative algorithm S$\ell_1$QP (see [33] and [34, pp. 312–314]) because it solves successive subproblems that are quadratic models of the nonlinear $\ell_1$ penalty function $\nu f(x) + \|c(x)\|_1$. Not only does this formulation overcome the inconsistency of (1.3.4)-(1.3.5), it also allows relaxation of the regularity assumption, so that $A_k$ can be singular at some iterates. Notice that Fletcher chooses the $\ell_\infty$ norm for the trust region constraint (trust region theory holds for every vector norm), thus confining $d$ to a hypercube of "radius" $\Delta_k$. The $\ell_\infty$ norm has the advantage of easily incorporating inequality bound constraints into the subproblem (the hypercube becomes a general box), but the disadvantage of making the exact solution of the subproblem combinatorial (in fact, NP-hard) and therefore unattractive for large-scale problems. Finally, note that the $\ell_1$ term is not continuous at points of feasibility, so that the solution of the subproblem requires nonsmooth optimization techniques.

Conn, Gould and Toint follow the same approach but instead use the augmented Lagrangian function, solving subproblems of the form

$$\min_{x \in \mathbf{R}^n} \quad f(x) + \hat{\lambda}_k^T c(x) + \frac{1}{2\mu_k} c(x)^T S_k c(x) \tag{1.3.6}$$

$$\text{subject to} \quad \|x - x_k\|_\infty \leq \Delta_k. \tag{1.3.7}$$

They call the penalty parameter $\mu_k$, choose $S_k$ to be some diagonal scaling matrix, and employ a special estimate $\hat{\lambda}_k$ for the Lagrange multipliers. The objective function is now smooth and can be minimized by standard optimization techniques (in fact, they solve a sequence of quadratic models of (1.3.6), similar to the operation of Algorithm 1.3.1). The trust region constraint (1.3.7) still has the combinatorial $\ell_\infty$ norm, so it would seem to be as unsuitable for large problems as Fletcher's S$\ell_1$QP algorithm; however, Conn, Gould and Toint prove in [16] that an approximate solution of (1.3.6)-(1.3.7) is adequate for global convergence, and in [15] they present an ingenious fast method for finding such an approximation. Their algorithm is particularly efficient for large-scale problems when the augmented Lagrangian function is group partially separable. The commercial software product LANCELOT [17] is a successful implementation of this approach. The performance of the LANCELOT algorithm is discussed further in §1.6.6.

The other general approach for overcoming inconsistency in (1.3.4)-(1.3.5) is to relax the linearized equality constraints (1.3.4) when necessary to allow feasible subproblem solutions. This idea is easily illustrated in Figure 1.2 for the simple problem considered in Figure 1.1. If an appropriate vector $\pi_k$ is chosen for this problem and the linear equality constraint

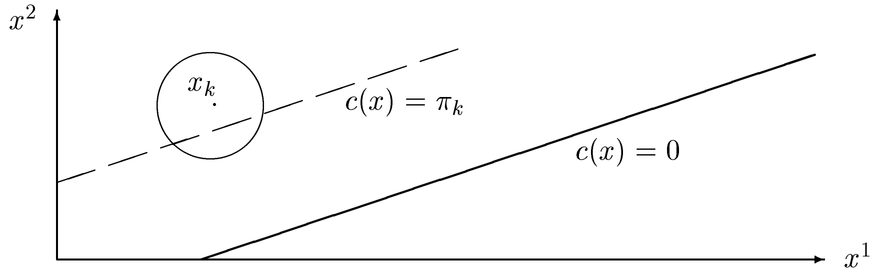


Figure 1.2: Example of relaxed constraints becoming consistent

is changed to $A_k^T d + c_k = \pi_k$ (shown as the dashed line), then the relaxed constraint falls inside the trust region constraint. Thus, the new subproblem

$$\begin{aligned} \min_{d \in \mathbf{R}^n} \quad & d^T g_k + \frac{1}{2} d^T W_k d && (1.3.8) \\ \text{subject to} \quad & A_k^T d + c_k = \pi_k && (1.3.9) \\ & \|d\|_2 \leq \Delta_k && (1.3.10) \end{aligned}$$

admits solutions $d$ that lie on the relaxed constraint (1.3.9). Alternatively, a nonnegative scalar $\theta_k$ can be chosen and a different subproblem (sometimes called the "two trust region" subproblem) defined as

$$\begin{aligned} \min_{d \in \mathbf{R}^n} \quad & d^T g_k + \frac{1}{2} d^T W_k d && (1.3.11) \\ \text{subject to} \quad & \|A_k^T d + c_k\|_2^2 \leq \theta_k && (1.3.12) \\ & \|d\|_2 \leq \Delta_k. && (1.3.13) \end{aligned}$$

This admits solutions $d$ in the region between the relaxed constraint (1.3.12) and the trust region (1.3.13). In either case, a value of $\pi_k = \emptyset$ or $\theta_k = 0$ yields the original subproblem (1.3.3)-(1.3.5). Although these formulations are consistent in every vector norm, the $\ell_2$ norm is preferred by all the researchers reviewed in the following paragraphs.

Both Vardi [77] and Byrd, Schnabel and Schultz [11] proposed algorithms with the subproblem formulation (1.3.8)-(1.3.10), and both selected $\pi_k = -\alpha_k c_k$ for some $\alpha_k \in [0, 1)$. Vardi limits his analysis to the case where some positive definite approximation of $W_k$ is used, thus sacrificing the quadratic convergence rate of second derivatives to permit the use of numerically stable Cholesky factorizations. His method for solving the consistent subproblem is motivated by the Moré and Sorensen algorithm GQTPAR [58] for solving quadratic trust region subproblems. Vardi requires a QR factorization of $A_k$ and finds an acceptable value for the relaxation parameter $\alpha_k$ in an inner loop, each inner iteration making a Cholesky factorization of size $n$. Although he employs clever updating techniques to minimize the computational work of these factorizations, the sparse matrices of large problems can always suffer from fill-in (especially in the QR factorization) and cause an excessive amount of work in this algorithm.

Byrd, Schnabel and Schultz [11] reduce the computational workload of solving (1.3.8)-(1.3.10) by decomposing the step $d$ into two parts that can be solved separately. They write $d = \alpha_k v_k + Z_k u$, where $v_k = -A_k(A_k^T A_k)^{-1} c_k$ and $u$ solves the reduced subproblem

$$\begin{array}{ll} \displaystyle\min_{u \in \mathbf{R}^{n-m}} & (g_k + \alpha_k W_k v_k)^T Z_k u + \frac{1}{2} u^T Z_k^T W_k Z_k u \\ \text{subject to} & \|Z_k u\|_2 \leq \sqrt{\Delta_k^2 - \alpha_k^2 \|v_k\|_2^2}, \end{array}$$

which results from substitution of their formula for $d$ into (1.3.8)-(1.3.10). Computing $v_k$ and $u$ requires solving smaller systems of sizes $m$ and $n-m$, respectively, and they prove that $u$ need only be found approximately to ensure the global convergence of the algorithm; furthermore, this formulation allows $W_k$ to be indefinite or singular. Notice that if $\alpha_k = 1$ and $x_k$ is close enough to a solution $x_*$ for the trust region constraint to be inactive, then the step $d$ is the same as that given by the null space Newton step (1.2.17). The main difficulty with this algorithm is choosing $\alpha_k$ intelligently. Though not crucial for obtaining global convergence, the choice of $\alpha_k$ strongly influences algorithm performance because it determines the relative sizes of $\|v_k\|$ and $\|Z_k u\|$, and hence controls whether $d$ tends more to satisfy constraint feasibility or to minimize the objective function.

The algorithm proposed by Zhang and Zhu [82] bears much similarity to that of Byrd, Schnabel and Schultz, although they derive it from a different point of view. They do not consider second derivatives and instead approximate the two-sided projected Hessian $Z_k^T W_k Z_k$ by a positive definite matrix $B_k$. The step $d$ is again decomposed into orthogonal parts $d = \alpha_k v_k + Z_k u$ with the same formula for $v_k$, but with $\alpha_k$ chosen to give the minimum of $\|\alpha_k A_k^T v_k + c_k\|_2^2$ subject to $\alpha_k \|v_k\|_2 \leq \Delta_k$. The reduced subproblem for $u$ differs in that the term containing the one-sided projection $Z_k^T W_k$ is dropped, and the size of the component is not tied to the size of $\alpha_k$:

$$\begin{array}{ll} \displaystyle\min_{u \in \mathbf{R}^{n-m}} & g_k^T Z_k u + \frac{1}{2} u^T B_k u \\ \text{subject to} & \|Z_k u\|_2 \leq \Delta_k. \end{array}$$

Despite the loss of information from the dropped term, they still prove superlinear convergence for suitable choice of $B_k$. The computation of $d$ is always well-defined (no inconsistent constraints), and the trust region constraint $\|d\|_2 \leq \sqrt{2}\,\Delta_k$ is always satisfied. Note that unlike Byrd, Schnabel and Schultz, this algorithm has no freedom in the choice of $\alpha_k$, and instead equal weight is given to the two components of each step $d$.

Celis, Dennis and Tapia proposed the two trust region subproblem (1.3.11)-(1.3.13) in [13]. They suggest a relaxation parameter of $\theta_k \leq \|A_k^T v_{cp} + c_k\|_2^2$, where $v_{cp}$ is the Cauchy point (see §1.4.2 for a definition of the Cauchy point) for the subproblem

$$\min_{v \in \mathbf{R}^n} \|A_k^T v + c_k\|_2^2 \quad \text{subject to} \quad \|v\|_2 \leq \Delta_k.$$

This choice forces the solution of (1.3.11)-(1.3.13) to make as much progress towards feasibility as the Cauchy point, and this turns out to be sufficient for proving global convergence [27]. However, more sophisticated choices of $\theta_k$ provide a powerful way of controlling the relative progress $d$ makes towards constraint feasibility or minimization of the objective. For example, Powell and Yuan [69] specify

$$\min_{\|d\|_2 \leq b_1 \Delta_k} \|A_k^T d + c_k\|_2^2 \leq \theta_k \leq \min_{\|d\|_2 \leq b_2 \Delta_k} \|A_k^T d + c_k\|_2^2, \quad \text{where} \quad 0 < b_2 \leq b_1 < 1.$$

This is a more stringent condition that forces greater progress towards satisfying feasibility, which they believe provides better algorithm performance. Unfortunately, the general subproblem (1.3.11)-(1.3.13), regardless of the value of $\theta_k$, is much more difficult to solve than problems with a single trust region constraint (a full analysis of this subproblem can be found in [81]). Iterative solution methods have been worked out for the special case of $W_k$ or $Z_k^T W_k Z_k$ always positive definite (see [83], for instance), but this restriction may preclude the use of second derivative information. A promising idea developed in [78] is to solve (1.3.11)-(1.3.13) approximately by restricting $d$ to a two-dimensional subspace. This reduced subproblem can be solved exactly (even for indefinite Hessians), and the two-dimensional subspace can be chosen at each iteration in a manner that should lead to global convergence. The algorithm in [78] relies on dense matrix calculations unsuitable for large-scale problems, but it may be possible to substitute cheaper methods and still maintain global convergence.

### 1.3.3 Byrd-Omojokun Algorithm

In this disquisition the formulation of Byrd and Omojokun [8] [61] is used to make the constraints (1.3.4)-(1.3.5) consistent. They first compute a step $v$ that satisfies a certain relaxation of the linearized equality constraints defined by the subproblem

$$\min_{v \in \mathbf{R}^n} \quad \|A_k^T v + c_k\|_2 \tag{1.3.14}$$

$$\text{subject to} \quad \|v\|_2 \leq \zeta \Delta_k, \quad \text{where} \quad \zeta \in (0,1). \tag{1.3.15}$$

Problem (1.3.14)-(1.3.15) is called the *vertical subproblem*, and its solution $v_k$ is called the *vertical step* at $x_k$. Now the original subproblem (1.3.3)-(1.3.5) is reformulated as

$$\min_{d \in \mathbf{R}^n} \quad d^T g_k + \frac{1}{2} d^T W_k d \tag{1.3.16}$$

$$\text{subject to} \quad A_k^T d = A_k^T v_k \tag{1.3.17}$$

$$\|d\|_2 \leq \Delta_k. \tag{1.3.18}$$

This new formulation is obviously consistent because the choice $d = v_k$ immediately satisfies both the constraints (1.3.17)-(1.3.18). It is the vertical subproblem that relaxes the original constraints enough to allow consistency, and the relation (1.3.17) has the effect of forcing the final step $d$ to make the same progress towards satisfying the linearized equality constraints as $v_k$ does. An interesting feature of this formulation is that when $m = n$, i.e., when the problem reduces to that of finding the root of a system of nonlinear equations, then only the vertical subproblem remains, and now the Byrd-Omojokun method coincides with the well-known Levenberg-Marquardt method.

As it stands, this formulation leaves a considerable amount of flexibility in the choice and computation of $v$ and $d$. In [8] [61] Byrd and Omojokun suggest a particular method that is well-suited to solving large-scale problems efficiently. Like the algorithm of Byrd, Schnabel and Schultz [11], they decompose $d$ into two orthogonal parts with respect to the matrix $A_k$. The vertical step $v$ is restricted to lie in the range space of $A_k$, and the other part of $d$ is confined to the null manifold spanned by the matrix $Z_k$; thus,

$$d = v + Z_k u, \quad \text{for some} \quad u \in \mathbf{R}^{n-m},$$

and $v^T Z_k u = 0$ (it is assumed as usual that $A_k$ is full rank). Having solved the vertical subproblem for an appropriate $v_k$, the substitution $d = v_k + Z_k u$ is made into (1.3.16)-(1.3.18) to obtain a subproblem with $u$ as the unknown. Note that the equation (1.3.17) is satisfied identically for every $u$, so it falls out of the subproblem. If terms in the objective independent of $u$ are dropped and the orthogonality relationship $v_k^T Z_k u = 0$ is used, then the result is simply

$$\min_{u \in \mathbf{R}^{n-m}} \quad (g_k + W_k v_k)^T Z_k u + \frac{1}{2} u^T Z_k^T W_k Z_k u \tag{1.3.19}$$

$$\text{subject to} \quad \|Z_k u\|_2 \leq \sqrt{\Delta_k^2 - \|v_k\|_2^2}. \tag{1.3.20}$$

This is called the *horizontal subproblem*, and the component $Z_k u_k$ found from it is called the *horizontal step* at $x_k$. Some researchers prefer to use the terms *normal* and *tangential* step instead of vertical and horizontal; they mean exactly the same thing, and this paper consistently uses only the latter terminology.

Figure 1.3 illustrates the vertical and horizontal steps for a simple problem with two unknowns and one nonlinear equality constraint. The dashed elliptical lines represent level curves of the objective $f(x)$, with the minimum in the lower right part of the picture. The solid line is the equality constraint, and the broken circle is the trust region constraint. The null manifold is shown as a dotted line through the point $x_k$ ($Z_k$ has just a single column in this example), and the vertical step runs perpendicularly to it. This same manifold translated to the point $x_k + v_k$ defines the possible set of horizontal steps, as shown by the lower dotted line. The final step $d$ in this case reaches all the way to the trust region constraint.

The Byrd-Omojokun method of vertical and horizontal steps is computationally efficient because it breaks the original equality constrained problem (1.3.3)-(1.3.5) into two smaller subproblems, each with only a single trust region constraint. In addition, the global convergence theory of trust regions allows each of these subproblems to be solved approximately, giving even greater savings. The method does depend on using the large matrix $Z_k$, but we will see that this can be done in a reasonably efficient manner. The key idea of computing the step $d$ via vertical and horizontal subproblems can be viewed as an extension of the null space method for solving the Newton subproblem derived in §1.2.4. The idea is also inherent in the algorithms proposed by Byrd, Schnabel and Schultz [11], Zhang and Zhu [82], and Dennis, El-Alem and Maciel [19].

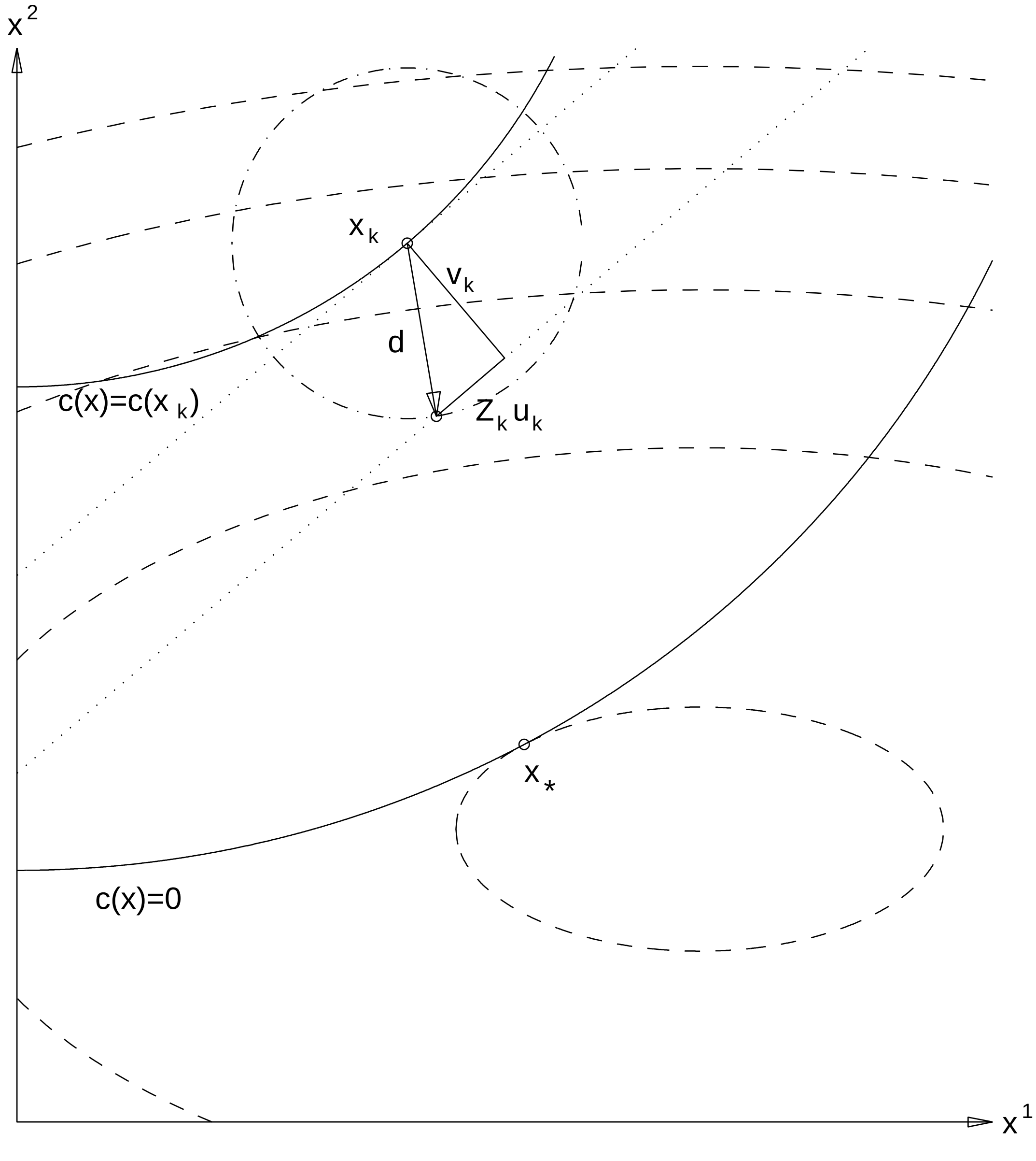


Figure 1.3: Byrd-Omojokun step $d = v_k + Z_k u_k$, with $\|d\|_2 \leq \Delta_k$

A general outline of the Byrd-Omojokun algorithm is given below. It follows the same format as Algorithm 1.3.1, our prototype for unconstrained minimization using trust regions. Algorithm 1.3.2 globalizes the convergence of Algorithm 1.2.16, the Newton-type method derived in §1.2.6 (this claim is substantiated later in §1.7). The form of Algorithm 1.3.2 is more general than that of Algorithm 1.2.16, but if it is specialized to compute $\hat{\lambda}_k$ and $W_k$ the same way as in Algorithm 1.2.16, then Algorithm 1.3.2 can possess the same local convergence properties as the Newton-type method.

**Algorithm 1.3.2** *General description of the Byrd-Omojokun algorithm*

Constants $\epsilon > 0$ and $\eta, \zeta, \gamma \in (0,1)$ are given
Choose any $x_0$ and $\Delta_0 > 0$
LOOP, starting with $k = 0$
  Compute $c_k$, $g_k$ and $A_k$
  Compute multiplier estimates $\hat{\lambda}_k$
  IF $\|g_k - A_k\hat{\lambda}_k\|_\infty < \epsilon$ AND $\|c_k\|_\infty < \epsilon$ THEN STOP
  Solve the vertical subproblem (1.3.14)-(1.3.15) for $v_k$
  Compute $Z_k$ such that $A_k^T Z_k = \emptyset$
  Compute $W_k$ to be $W(x_k, \hat{\lambda}_k)$ or update an approximation to it
  Solve the horizontal subproblem (1.3.19)-(1.3.20) for $u_k$
  $d_k = v_k + Z_k u_k$
  Compute the actual reduction in the merit function, a_red,
  and the predicted reduction p_red at $d_k$
  IF $\frac{\text{a\_red}}{\text{p\_red}} \geq \eta$
    THEN $x_{k+1} = x_k + d_k$, $\Delta_{k+1} \geq \Delta_k$
    ELSE $x_{k+1} = x_k$ $\Delta_{k+1} \leq \gamma \|d_k\|_2$
CONTINUE, after incrementing $k$

In this section the Byrd-Omojokun algorithm was derived as a consistent method for applying trust regions to the quadratic subproblem (1.2.28). Trust regions globalize the convergence of the algorithm, even when the steps $v_k$ and $u_k$ are computed approximately. The Byrd-Omojokun algorithm was chosen over other alternatives because it breaks the constrained quadratic subproblem (1.3.3)-(1.3.5) into two smaller, simpler subproblems, and because its performance does not hinge an determining values for extraneous algorithm parameters.

The next step in this dissertation is to show how Algorithm 1.3.2 can be implemented in a manner that solves large problems efficiently. This requires careful consideration of every computation made in the algorithm, most especially the solution of the vertical and horizontal subproblems. Fortunately, these are both of the same type as those that arise in unconstrained trust region methods, which have been studied extensively for years. The next section digresses briefly and presents efficient methods for solving these subproblems.

## 1.4 Large-Scale Trust Region Subproblems

This section is concerned with solving the problem

$$\min_{d \in \mathbf{R}^q} \quad \phi(d) = g^T d + \frac{1}{2} d^T B d \tag{1.4.1}$$

$$\text{subject to} \qquad \|Md\|_2 \le \Delta, \tag{1.4.2}$$

where $B$ is a $q \times q$ symmetric matrix and $M$ is a full rank $p \times q$ matrix with $p \ge q$. This general *trust region subproblem* was historically considered in simpler form by Levenberg [51] and Marquardt [56] for solving nonlinear least squares problems, and later by Goldfeld, Quandt and Trotter [40] for the unconstrained minimization of a nonlinear function. The standard methods for solving trust region subproblems are discussed in most modern textbooks (such as [34], [20], [39]), and in the excellent articles of Moré [57] and Sorensen [73].

Our reason for studying (1.4.1)-(1.4.2) is to find efficient ways of solving it when $p$ and $q$ are large and the matrices $B$ and $M$ are sparse. In the Byrd-Omojokun algorithm of the previous section both the vertical subproblem (1.3.14)-(1.3.15) and the horizontal subproblem (1.3.19)-(1.3.20) are of the form (1.4.1)-(1.4.2), and both subproblems are large and sparse. This section will examine the exact solution of (1.4.1)-(1.4.2), and a variety of approximate solutions whose accuracy is sufficient to preserve global convergence of the Byrd-Omojokun algorithm.

Since $M$ is full rank, the $q \times q$ matrix $M^T M$ is positive definite, and the trust region (1.4.2) is ellipsoidal in shape (with principle axes in the directions of the eigenvectors of $M^T M$ and semi-axis lengths that are inversely proportional to the square root of the eigenvalues of $M^T M$). A practical approach for obtaining methods to solve (1.4.1)-(1.4.2) is to transform it into an equivalent problem that has a spherical trust region constraint, develop methods for solving that problem, and then transform the results back. For this purpose we will use the transformation defined by

$$\tilde{d} = (M^T M)^{1/2} d \tag{1.4.3}$$

and assume that a symmetric square root is used. The inverse transformation is well-defined because $M^T M$ is nonsingular; it is simply

$$d = (M^T M)^{-1/2} \tilde{d}. \tag{1.4.4}$$

Now inserting (1.4.4) into (1.4.1)-(1.4.2) yields the equivalent problem

$$\min_{\tilde{d} \in \mathbf{R}^q} \quad \tilde{\phi}(\tilde{d}) = \tilde{g}^T \tilde{d} + \frac{1}{2} \tilde{d}^T \tilde{B} \tilde{d} \tag{1.4.5}$$

$$\text{subject to} \qquad \|\tilde{d}\|_2 \le \Delta, \tag{1.4.6}$$

where

$$\tilde{g} \equiv (M^T M)^{-1/2} g \quad \text{and} \quad \tilde{B} \equiv (M^T M)^{-1/2} B \, (M^T M)^{-1/2}. \tag{1.4.7}$$

The two objectives (1.4.1) and (1.4.5) have the same value when evaluated at points $\tilde{d}$ and $d$ that are images of one another; i.e., $\tilde{\phi}(\tilde{d}) = \phi((M^T M)^{1/2} d)$.

### 1.4.1 Exact Solution

Although our primary interest is to find approximate solutions to (1.4.5)-(1.4.6), it is worth examining the exact solution to obtain a better understanding of the trust region subproblem. The complete solution is specified in the theorem below (shorter but less edifying proofs can be found in [73] and [34]).

**Theorem 1.4.1** *The point $\tilde{d}_*$ is a solution to (1.4.5)-(1.4.6) if and only if $\|\tilde{d}_*\|_2 \le \Delta$ and there exists some $\tilde{\mu}_* \in \mathbf{R}$ such that the following conditions are satisfied:*

$$(\tilde{B} + \tilde{\mu}_* I)\, \tilde{d}_* = -\tilde{g}, \tag{1.4.8}$$

$$\tilde{\mu}_* \ge 0, \tag{1.4.9}$$

$$\tilde{\mu}_*(\Delta - \|\tilde{d}_*\|_2) = 0, \quad \textit{and} \tag{1.4.10}$$

$$\tilde{B} + \tilde{\mu}_* I \quad \textit{is a positive semidefinite matrix.} \tag{1.4.11}$$

*Proof.* All of the conditions but (1.4.11) are a consequence of the first-order KKT conditions for inequality constrained problems applied to (1.4.5)-(1.4.6), but it is instructive to make a more direct proof, invoking only the KKT conditions for equality constrained optimization derived in §1.2.1. The two directions of the if and only if statement must both be proved. They are: (forward direction) if $\tilde{d}_*$ is a solution of (1.4.5)-(1.4.6) then it implies that (1.4.8)-(1.4.11) are satisfied, and (reverse direction) if $\tilde{d}_*$ and $\tilde{\mu}_*$ satisfy (1.4.8)-(1.4.11) then it implies that $\tilde{d}_*$ is a solution. For each direction the trust region constraint (1.4.6) is either active or inactive, making a total of four cases to consider.

*Case i: Forward direction, $\|\tilde{d}_*\|_2 < \Delta$.* Here $\tilde{d}_*$ is assumed to be a solution of the unconstrained problem (1.4.5). The first- and second-order KKT conditions must be true at a solution, and these conditions for (1.4.5) are that

$$\nabla\tilde{\phi}(\tilde{d}_*) = \tilde{B}\tilde{d}_* + \tilde{g} = \emptyset, \quad \text{and} \quad \nabla^2\tilde{\phi}(\tilde{d}_*) = \tilde{B} \quad \text{is positive semidefinite.}$$

Therefore, (1.4.8)-(1.4.11) are satisfied by choosing $\tilde{\mu}_* = 0$.

*Case ii: Reverse direction, $\|\tilde{d}_*\|_2 < \Delta$.* Here (1.4.8)-(1.4.11) are assumed to hold for $\tilde{d}_*$. Condition (1.4.10) implies that $\tilde{\mu}_* = 0$, and then (1.4.8) and (1.4.11) reduce to

$$\tilde{B}\tilde{d}_* = -\tilde{g}, \quad \text{and } \tilde{B} \text{ is positive semidefinite.}$$

These are exactly the first- and second-order necessary conditions for $\tilde{d}_*$ to be a minimizer of (1.4.5); furthermore, since $\tilde{\phi}$ is quadratic (has no terms beyond second-order) these conditions are also sufficient.

*Case iii: Forward direction, $\|\tilde{d}_*\|_2 = \Delta$.* Here (1.4.5)-(1.4.6) is treated as an equality constrained problem and $\tilde{d}_*$ is assumed to solve it. The first- and second-order KKT conditions for such a problem, which must hold at a solution, were derived earlier in (1.2.1) and (1.2.4). Writing the constraint (1.4.6) as $\tilde{c}(\tilde{d}) = \frac{1}{2}(\Delta^2 - \|\tilde{d}\|_2^2)$ and calling its Lagrange multiplier $\lambda_*$, the KKT conditions for the equality constrained version of (1.4.5)-(1.4.6) takes the form

$$\tilde{c}(\tilde{d}_*) = 0, \quad \nabla\tilde{\phi}(\tilde{d}_*) = \lambda_*\nabla\tilde{c}(\tilde{d}_*), \quad \text{and}$$
$$s^T[\nabla^2\tilde{\phi}(\tilde{d}_*)]s \geq 0 \quad \text{for all } s \in \mathbf{R^q} \text{ such that} \quad s^T\nabla\tilde{c}(\tilde{d}_*) = 0.$$

(Notice a nonstrict inequality is used for the second-order KKT condition – this is more general than (1.2.4).) Substituting for $\tilde{\phi}$ and $\tilde{c}$ and evaluating the gradients and Hessians gives

$$\|\tilde{d}_*\|_2^2 = \Delta^2, \quad \tilde{B}\tilde{d}_* + \tilde{g} = -\lambda_*\tilde{d}_*, \quad \text{and}$$
$$s^T(\tilde{B} + \lambda_* I)s \geq 0 \quad \text{for all } s \in \mathbf{R^q} \text{ such that} \quad s^T\tilde{d}_* = 0. \tag{1.4.12}$$

If we choose $\tilde{\mu}_* = \lambda_*$, then the first line immediately implies (1.4.8) and (1.4.10). Now the point $-\tilde{d}_*$ is also on the trust region, and since $\tilde{\phi}$ is minimized at $\tilde{d}_*$ it must be that $\tilde{\phi}(\tilde{d}_*) \leq \tilde{\phi}(-\tilde{d}_*)$; therefore,

$$\tilde{\phi}(\tilde{d}_*) - \tilde{\phi}(-\tilde{d}_*) = 2\tilde{g}^T\tilde{d}_* \leq 0. \tag{1.4.13}$$

Every vector $s \in \mathbf{R^q}$ can be expressed as the sum of its projection along $\tilde{d}_*$ plus a vector orthogonal to $\tilde{d}_*$. The second-order KKT condition (1.4.12) shows $\tilde{B} + \tilde{\mu}_* I$ to be positive semidefinite with respect to vectors orthogonal to $\tilde{d}_*$, so to finish proving (1.4.11) we need to show that $\tilde{B} + \tilde{\mu}_* I$ is positive semidefinite also for vectors along $\tilde{d}_*$; that is, for $s = \alpha\tilde{d}_*$, $\alpha \in \mathbf{R}$. This is established directly by using (1.4.8) and (1.4.13):

$$(\alpha\tilde{d}_*)^T(\tilde{B} + \tilde{\mu}_* I)(\alpha\tilde{d}_*) = \alpha^2\tilde{d}_*^T[(\tilde{B} + \tilde{\mu}_* I)\tilde{d}_*] = -\alpha^2\tilde{d}_*^T\tilde{g} \geq 0.$$

Finally, to prove (1.4.9) recall the *case iii* assumption that the minimizer $\tilde{d}_*$ lies on the trust region constraint, which implies that $\tilde{\phi}$ cannot be reduced by moving to a point slightly inside the trust region. Let us write this statement as $\tilde{\phi}((1-\epsilon)\tilde{d}_*) \geq \tilde{\phi}(\tilde{d}_*)$, for $0 < \epsilon \ll 1$. Substituting into $\tilde{\phi}$ gives

$$\tilde{\phi}((1-\epsilon)\tilde{d}_*) = \tilde{\phi}(\tilde{d}_*) - \epsilon(\tilde{g}^T\tilde{d}_* + \tilde{d}_*^T\tilde{B}\tilde{d}_*) + \frac{1}{2}\epsilon^2\tilde{d}_*^T\tilde{B}\tilde{d}_* \geq \tilde{\phi}(\tilde{d}_*),$$

and so for sufficiently small $\epsilon$

$$-(\tilde{g}^T\tilde{d}_* + \tilde{d}_*^T\tilde{B}\tilde{d}_*) \geq 0.$$

Having already established (1.4.8) for this case, we can use it to prove (1.4.9):

$$-(\tilde{g}^T\tilde{d}_* + \tilde{d}_*^T\tilde{B}\tilde{d}_*) = \tilde{\mu}_*\tilde{d}_*^T\tilde{d}_* \geq 0.$$

*Case iv: Reverse direction,* $\|\tilde{d}_*\|_2 = \Delta$. Here (1.4.8)-(1.4.11) are assumed to hold for the point $\tilde{d}_*$. But (1.4.8) and (1.4.11) immediately imply the first- and second-order necessary conditions for $\tilde{d}_*$ to be the minimizer of (1.4.5)-(1.4.6) with (1.4.6) treated as an equality (see *case iii*). For quadratic objective $\tilde{\phi}$ these conditions are also sufficient. □

**Corollary 1.4.2** *The point $d_*$ is a solution to (1.4.1)-(1.4.2) if and only if* $\|Md_*\|_2 \leq \Delta$ *and there exists some $\mu_* \in \mathbf{R}$ such that the following conditions are satisfied:*

$$(B + \mu_* M^T M)\, d_* = -g, \tag{1.4.14}$$

$$\mu_* \geq 0,$$

$$\mu_*(\Delta - \|Md_*\|_2) = 0, \quad \textit{and}$$

$$B + \mu_* M^T M \quad \textit{is a positive semidefinite matrix.}$$

*Proof.* Make the substitutions (1.4.3) and (1.4.7) into conditions (1.4.8)-(1.4.11) of Theorem 1.4.1. □

A deeper understanding of the exact solution $\tilde{d}_*$ is obtained by examining a spectral expansion of the problem. Let $\{\tilde{\lambda}_1, \ldots, \tilde{\lambda}_q\}$ be the set of eigenvalues of $\tilde{B}$ indexed in the order $\tilde{\lambda}_1 \leq \cdots \leq \tilde{\lambda}_q$, and let $\{\tilde{\nu}_1, \ldots, \tilde{\nu}_q\}$ be a corresponding set of orthonormal eigenvectors. Define the matrices $\tilde{Q} = [\tilde{\nu}_1 \cdots \tilde{\nu}_q]$ and $\tilde{\Lambda} = diag\{\tilde{\lambda}_1, \ldots, \tilde{\lambda}_q\}$. Then the eigenvalue/eigenvector properties of $\tilde{B}$ are expressed compactly as $\tilde{B}\tilde{Q} = \tilde{Q}\tilde{\Lambda}$ and $\tilde{Q}^T\tilde{Q} = I$, and substitution of these into (1.4.8) yields

$$(\tilde{\Lambda} + \tilde{\mu}_* I)\, \tilde{Q}^T \tilde{d}_* = -\tilde{Q}^T \tilde{g}.$$

By writing out each of the rows of this equation we obtain

$$\begin{bmatrix} (\tilde{\lambda}_1 + \tilde{\mu}_*)\, \tilde{\nu}_1^T \tilde{d}_* \\ \vdots \\ (\tilde{\lambda}_q + \tilde{\mu}_*)\, \tilde{\nu}_q^T \tilde{d}_* \end{bmatrix} = \begin{bmatrix} -\tilde{\nu}_1^T \tilde{g} \\ \vdots \\ -\tilde{\nu}_q^T \tilde{g} \end{bmatrix}. \tag{1.4.15}$$

The $q$ equalities in (1.4.15) allow solutions of (1.4.5)-(1.4.6) to be constructed in terms of the eigenvalues and eigenvectors of $\tilde{B}$, and the projection of $\tilde{g}$ on each eigenvector. From this point of view, solutions may be classified into one of the following categories:

- $\tilde{B}$ is positive definite and $\|\tilde{B}^{-1}\tilde{g}\|_2 \leq \Delta$. In this case $\tilde{\mu}_* = 0$ and $\tilde{d}_* = -\tilde{B}^{-1}\tilde{g}$.

- $\tilde{B}$ is positive definite and $\|\tilde{B}^{-1}\tilde{g}\|_2 > \Delta$. Now $\tilde{\mu}_*$ must be positive to make $\tilde{d}_*$ different from $-\tilde{B}^{-1}\tilde{g}$. Because the eigenvectors are of unit size, it follows from (1.4.15) that

$$\|\tilde{d}_*\|_2^2 = \sum_{i=1}^{q} \left( \frac{\tilde{\nu}_i^T \tilde{g}}{\tilde{\lambda}_i + \tilde{\mu}_*} \right)^2. \tag{1.4.16}$$

  If $\tilde{\mu} > 0$ is regarded as an independent variable in (1.4.16), then $\|\tilde{d}_*\|_2$ is strictly decreasing as $\tilde{\mu}$ increases; hence, there exists a unique value of $\tilde{\mu}_*$ just large enough to make $\|\tilde{d}_*\|_2 = \Delta$.

- $\tilde{\lambda}_1 = \cdots = \tilde{\lambda}_j$ with $\tilde{\lambda}_1 \leq 0$, and at least one of the projections $\tilde{\nu}_1^T \tilde{g}, \ldots, \tilde{\nu}_j^T \tilde{g}$ is nonzero. The nonzero projection implies that $\tilde{\mu}_* \neq -\tilde{\lambda}_1$, and with condition (1.4.11) this means that $\tilde{\mu}_* > -\tilde{\lambda}_1$. Equation (1.4.16) still holds for this case and determines the exact value of $\tilde{\mu}_*$.

- $\tilde{\lambda}_1 = \cdots = \tilde{\lambda}_j$ with $\tilde{\lambda}_1 \le 0$, and $\tilde{\nu}_1^T \tilde{g} = \cdots = \tilde{\nu}_j^T \tilde{g} = 0$. Since all the projections are zero, it is possible to choose $\mu_* = -\tilde{\lambda}_1$, in which case the terms $\tilde{\nu}_1^T \tilde{d}_*, \ldots, \tilde{\nu}_j^T \tilde{d}_*$ are not determined by (1.4.15). Let $\bar{d}$ be the solution to (1.4.15) when $\mu_* = -\tilde{\lambda}_1$ and all the undetermined projections are set to zero; i.e., when $\tilde{\nu}_1^T \bar{d} = \cdots = \tilde{\nu}_j^T \bar{d} = 0$. Then (1.4.16) becomes
$$\|\bar{d}\|_2^2 = \sum_{i=j+1}^{q} \left( \frac{\tilde{\nu}_i^T \tilde{g}}{\tilde{\lambda}_i + \tilde{\mu}_*} \right)^2 .$$
Now there are three possibilities: if $\|\bar{d}\|_2 > \Delta$, then $\mu_* > -\tilde{\lambda}_1$ and equation (1.4.16) specifies the solution; if $\|\bar{d}\|_2 = \Delta$, then $\mu_* = -\tilde{\lambda}_1$ and $\tilde{d}_* = \bar{d}$; finally, if $\|\bar{d}\|_2 < \Delta$, then $\mu_* = -\tilde{\lambda}_1$ and the undetermined projections in (1.4.15) can take any value so long as $\|\tilde{d}_*\|_2 \le \Delta$. This very last possibility characterizes the only situation in which the minimizer of (1.4.5)-(1.4.6) is not unique.

Transforming (1.4.15) to obtain the spectral expansion of (1.4.14) does not reveal much that is useful. But as a side note, certain types of generalized eigenvalue problems can be transformed to the form $\tilde{B}\tilde{Q} = \tilde{Q}\tilde{\Lambda}$ (see [75, pp. 343-345]), and so to some extent the generalized eigenvalue problem can be related to problem (1.4.1)-(1.4.2).

A numerically stable algorithm that converges to the exact solution of (1.4.5)-(1.4.6) in all cases was developed by Moré and Sorensen [58]. Unfortunately, this algorithm requires Cholesky factorizations of $\tilde{B} + \tilde{\mu} I$, which is not an acceptable strategy for large-scale problems (the factorization could fill in and become dense, or an explicit form of the large matrix $\tilde{B}$ may not be available for factorization). But the exact solution $\tilde{d}_*$ is not necessary for global convergence of a trust region algorithm; in fact, the minimum degree of accuracy required for robust performance of the Byrd-Omojokun algorithm is supplied by any point $\tilde{d}$ that makes as much decrease in $\tilde{\phi}$ as the Cauchy point does (see §1.7.2 for details). This can be obtained fairly cheaply, as the next section shows.

### 1.4.2 Cauchy Point

The Cauchy point can be defined for problems even more general than (1.4.1)-(1.4.2).

**Definition 1.4.3** *The* **Cauchy point** *for the problem*
$$\min_{s \in \mathbf{R^q}} \ \phi(s) \qquad \textit{subject to} \quad \|Ms\|_2 \le \Delta,$$
*with $M$ full rank, is the vector $s_{cp}$ that solves*
$$\min_{s \in \mathbf{R^q}} \quad \phi(s)$$
$$\begin{array}{rrcl} \textit{subject to} & \|Ms\|_2 & \le & \Delta \\ & s & = & -\alpha (M^T M)^{-1} \nabla \phi(0) \\ & \alpha & \ge & 0. \end{array}$$

Definition 1.4.3 says that the Cauchy point is the minimizer along the direction of scaled steepest descent. If we start by linearizing $\phi$ around the point $s = \emptyset$, then the solution to the simpler problem

$$\min_{s \in \mathbf{R}^q} \ \phi(0) + s^T \nabla\phi(0) \qquad \text{subject to} \quad \|Ms\|_2 \le \Delta$$

is always in the direction of scaled steepest descent $-(M^T M)^{-1} \nabla\phi(0)$. The Cauchy point is the minimizer in this direction of the fully nonlinear objective $\phi$, still subject to its constraint.

Applying Definition 1.4.3 to (1.4.5)-(1.4.6), we see that the direction of scaled steepest descent is simply $-\tilde{g}$, and the Cauchy point is

$$\tilde{d}_{cp} = -\tilde{\alpha}\tilde{g}, \tag{1.4.17}$$

where $\tilde{\alpha} \ge 0$ is found by solving

$$\min_{\tilde{\alpha} \in \mathbf{R}} \ \tilde{g}^T(-\tilde{\alpha}\tilde{g}) + \frac{1}{2}(-\tilde{\alpha}\tilde{g})\tilde{B}(-\tilde{\alpha}\tilde{g}) \qquad \text{subject to} \quad \|(-\tilde{\alpha}\tilde{g})\|_2 \le \Delta.$$

The answer to this is easily computed to be

$$\tilde{\alpha} = \begin{cases} \dfrac{\tilde{g}^T\tilde{g}}{\tilde{g}^T\tilde{B}\tilde{g}}, & \text{if } \tilde{g}^T\tilde{B}\tilde{g} > 0 \quad \text{and} \quad \dfrac{(\tilde{g}^T\tilde{g})^{3/2}}{\tilde{g}^T\tilde{B}\tilde{g}} \le \Delta \\ \dfrac{\Delta}{\|\tilde{g}\|_2}, & \text{otherwise.} \end{cases} \tag{1.4.18}$$

The Cauchy point for (1.4.1)-(1.4.2) can be found directly from Definition 1.4.3, or by applying the transformations (1.4.3) and (1.4.7) to (1.4.17) and (1.4.18). In either case, the result is

$$d_{cp} = -\tilde{\alpha}(M^T M)^{-1} g, \tag{1.4.19}$$

showing that the ellipsoidal trust region constraint induces a scaling of the direction of steepest descent.

In unconstrained optimization (and also in the Byrd-Omojokun algorithm) the use of trust regions leads to global convergence if an approximate solution to each trust region subproblem makes at least as much reduction in the objective as a step to the Cauchy point of the subproblem does. This property is a consequence of the fact that the Cauchy point is the limiting solution to the trust region subproblem as $\Delta \to 0$. The radius $\Delta$ restricts the size of the subproblem solution, and when the size is much smaller than one then the nonlinear objective of the subproblem is accurately represented by a linear approximation. As we have seen, the Cauchy point is in precisely the direction that solves this linearized version of the subproblem; hence, the Cauchy point is guaranteed to make a sufficiently accurate solution to the trust region subproblem when $\Delta$ becomes sufficiently small.

### 1.4.3 Dogleg Method

If the Hessian matrix $\tilde{B}$ in (1.4.5) is positive definite, then a commonly used approximate solution to (1.4.5)-(1.4.6) is that given by Powell's dogleg method [64]. This method calculates the Cauchy point $\tilde{d}_{cp}$ and the Newton point $\tilde{d}_{nw} = -\tilde{B}^{-1}\tilde{g}$, temporarily ignoring the trust region constraint. The dogleg path is defined as the two line segments extending from $\tilde{d} = \emptyset$ to $\tilde{d} = \tilde{d}_{cp}$, and from $\tilde{d} = \tilde{d}_{cp}$ to $\tilde{d} = \tilde{d}_{nw}$. The dogleg method finds the minimizer of $\tilde{\phi}$ along this path subject to $\|\tilde{d}\|_2 \leq \Delta$. It will be seen shortly that the objective $\tilde{\phi}$ decreases monotonically along the dogleg path, so the dogleg solution is simply the intersection point of the path with the trust region boundary, or the Newton point if the path lies entirely inside the trust region [20, pp. 139–143].

Two essential properties of the dogleg method make it well-defined and straightforward to calculate: the Newton point is always farther away than the Cauchy point, and the quadratic objective $\tilde{\phi}$ decreases monotonically along the dogleg path. In particular, global convergence follows from these properties because they ensure that the dogleg solution reduces the objective at least as much as the Cauchy point does.

**Theorem 1.4.4** *If $\tilde{B}$ is positive definite in (1.4.5)-(1.4.6), then*

$$\|\tilde{d}_{cp}\|_2 \leq \|\tilde{d}_{nw}\|_2.$$

*Proof.* Examination of (1.4.18) shows that $\tilde{\alpha}$ is at most $\tilde{g}^T\tilde{g}/\tilde{g}^T\tilde{B}\tilde{g}$, so that

$$\|\tilde{d}_{cp}\|_2 \leq \frac{(\tilde{g}^T\tilde{g})^{3/2}}{\tilde{g}^T\tilde{B}\tilde{g}};$$

therefore,

$$\frac{\|\tilde{d}_{nw}\|_2}{\|\tilde{d}_{cp}\|_2} \geq \sqrt{\tilde{g}^T\tilde{B}^{-2}\tilde{g}}\,\frac{\tilde{g}^T\tilde{B}\tilde{g}}{(\tilde{g}^T\tilde{g})^{3/2}}.$$

Now it follows from the Cauchy-Schwartz inequality that

$$(\tilde{g}^T\tilde{B}^{-2}\tilde{g})(\tilde{g}^T\tilde{g}) \;=\; \|\tilde{B}^{-1}\tilde{g}\|_2^2\,\|\tilde{g}\|_2^2 \;\geq\; (\tilde{g}^T\tilde{B}^{-1}\tilde{g})^2,$$

and this implies

$$\frac{\|\tilde{d}_{nw}\|_2}{\|\tilde{d}_{cp}\|_2} \geq \frac{(\tilde{g}^T\tilde{B}^{-1}\tilde{g})(\tilde{g}^T\tilde{B}\tilde{g})}{(\tilde{g}^T\tilde{g})^2}.$$

Since $\tilde{B}$ is symmetric positive definite, its inverse exists and is also symmetric positive definite; furthermore, both matrices possess a symmetric positive definite square root (for instance, $\sqrt{\tilde{B}} = \tilde{Q}\sqrt{\tilde{\Lambda}}\tilde{Q}^T$, using the spectral expansion of $\tilde{B}$). Applying the Cauchy-Schwartz inequality to this situation gives

$$(\tilde{g}^T\tilde{B}^{-1}\tilde{g})(\tilde{g}^T\tilde{B}\tilde{g}) \;=\; \|\sqrt{\tilde{B}^{-1}}\tilde{g}\|_2^2\,\|\sqrt{\tilde{B}}\tilde{g}\|_2^2 \;\geq\; (\tilde{g}^T\tilde{g})^2, \tag{1.4.20}$$

which implies that

$$\|\tilde{d}_{nw}\|_2 \geq \|\tilde{d}_{cp}\|_2.$$

□

**Theorem 1.4.5** *If $\tilde{B}$ is positive definite in (1.4.5)-(1.4.6), then $\tilde{\phi}$ decreases monotonically along the dogleg path.*

*Proof.* The Cauchy point is constructed to be the minimizer along the direction of steepest descent, so $\tilde{\phi}$ decreases monotonically along the first path segment from $\tilde{d} = \emptyset$ to $\tilde{d} = \tilde{d}_{cp}$. To analyze the second path segment from $\tilde{d} = \tilde{d}_{cp}$ to $\tilde{d} = \tilde{d}_{nw}$, express it in terms of the scalar parameter $\tau$ as

$$\tilde{d}(\tau) = \tau \tilde{d}_{nw} + (1-\tau)\tilde{d}_{cp}, \quad \tau \in [0,1],$$

and consider $\tilde{\phi}(\tau)$, the objective along this path:

$$\begin{aligned} \tilde{\phi}(\tau) &= \tau \tilde{g}^T \tilde{d}_{nw} + (1-\tau)\tilde{g}^T \tilde{d}_{cp} \\ &\quad + \frac{1}{2}\tau^2 \tilde{d}_{nw}^T \tilde{B} \tilde{d}_{nw} + \tau(1-\tau)\tilde{d}_{nw}^T \tilde{B} \tilde{d}_{cp} + \frac{1}{2}(1-\tau)^2 \tilde{d}_{cp}^T \tilde{B} \tilde{d}_{cp} \\ &= (-\frac{1}{2}\tau^2 + \tau - \frac{1}{2})\frac{(\tilde{g}^T\tilde{g})^2}{\tilde{g}^T\tilde{B}\tilde{g}} + (\frac{1}{2}\tau^2 - \tau)\tilde{g}^T\tilde{B}^{-1}\tilde{g}. \end{aligned}$$

To complete the proof we must show that $\tilde{\phi}(\tau)$ is nonincreasing as $\tau$ varies from 0 to 1. This follows by applying inequality (1.4.20) to show that the derivative of $\tilde{\phi}$ with respect to $\tau$ is never positive:

$$\frac{d\tilde{\phi}}{d\tau} = (1-\tau)\left[\frac{(\tilde{g}^T\tilde{g})^2}{\tilde{g}^T\tilde{B}\tilde{g}} - \tilde{g}^T\tilde{B}^{-1}\tilde{g}\right] \leq 0.$$

A simpler proof of this property is to recognize that (1.4.5) is a convex objective when $\tilde{B}$ is positive definite; thus, it has a unique global minimum at the Newton point, and $\tilde{\phi}$ is strictly decreasing along every straight path that ends at the global minimum. □

Problem (1.4.1)-(1.4.2) can also be solved by Powell's dogleg method (note that by Sylvester's Theorem [75, pp. 341–342] the transformation (1.4.7) ensures that $B$ is positive definite whenever $\tilde{B}$ is, and vice versa). In this case the dogleg path is marked by the Cauchy point in (1.4.19) and the Newton point $d_{nw} = -B^{-1}g$, and its intersection with the ellipsoidal trust region boundary gives the solution point for the method. Theorems 1.4.4 and 1.4.5 are easily extended to problem (1.4.1)-(1.4.2).

**Theorem 1.4.6** *If $B$ is positive definite in (1.4.1)-(1.4.2), then*

$$\|Md_{cp}\|_2 \leq \|Md_{nw}\|_2$$

*and $\phi$ decreases monotonically along the dogleg path.*

*Proof.* The same arguments used in proving Theorems 1.4.4 and 1.4.5 work by substituting in (1.4.3) and (1.4.7) and carrying out the algebra. □

For future use in §1.5.1, let us consider the angle at $\tilde{d} = \emptyset$ between the Cauchy point and the Newton point; that is, the angle $\tilde{\theta}$ such that

$$\cos\tilde{\theta} = \frac{\tilde{d}_{cp}^T \tilde{d}_{nw}}{\|\tilde{d}_{cp}^T\|_2 \; \|\tilde{d}_{nw}\|_2} = \frac{\tilde{g}^T \tilde{B}^{-1} \tilde{g}}{\|\tilde{g}\|_2 \sqrt{\tilde{g}^T \tilde{B}^{-2} \tilde{g}}}.$$

The second equality, obtained by substituting the formulas for the Cauchy and Newton points, shows that this angle is always acute. Now it is well known that the steepest descent and Newton steps of an unconstrained optimization problem can become nearly orthogonal as the condition number of the Hessian becomes large. This suggests that the angle $\tilde{\theta}$ may become large for ill-conditioned $\tilde{B}$. In fact, we can compute the largest possible angle for a given $\tilde{B}$ and show that this is true.

**Theorem 1.4.7** *If $\tilde{B}$ is positive definite in (1.4.5)-(1.4.6) with condition number $\kappa_{\tilde{B}}$, then*

$$\cos\tilde{\theta} \geq 2\frac{\sqrt{\kappa_{\tilde{B}}}}{1+\kappa_{\tilde{B}}}, \quad \textit{so that} \quad \tilde{\theta} \leq \cos^{-1}\left(2\frac{\sqrt{\kappa_{\tilde{B}}}}{1+\kappa_{\tilde{B}}}\right).$$

*Proof.* Geometrically, the contours of the positive definite quadratic objective in (1.4.5) are concentric ellipsoids centered around the Newton point. The axes of the ellipsoids are oriented along the eigenvectors of $\tilde{B}$, and the length of each axis is inversely proportional to the square root of the corresponding eigenvalue. The point $\tilde{d} = \emptyset$ lies on some ellipsoidal contour, and the gradient $\tilde{g}$ at that point is normal to this ellipsoid, while the vector to the center of the ellipsoid gives the direction of $\tilde{d}_{nw}$. By varying the gradient $\tilde{g}$ we can consider all points on the same ellipsoidal contour, and hence all possible angles between these two vectors. To facilitate this search, let us rotate the coordinate system to line up with the orthonormal eigenvectors $\tilde{Q} = [\tilde{\nu}_1 \cdots \tilde{\nu}_q]$ of $\tilde{B}$ by defining the transformations

$$w = \tilde{Q}^T \tilde{d} \quad \text{and} \quad y = \tilde{Q}^T \tilde{g}$$

(since $\tilde{Q}^T\tilde{Q} = I$ this transformation preserves angles). Now the search for an upper bound on $\cos\tilde{\theta}$ can be phrased algebraically as

$$\min_{\tilde{g}\in\mathbf{R}^q} \frac{\tilde{d}_{cp}^T \tilde{d}_{nw}}{\|\tilde{d}_{cp}^T\|_2 \|\tilde{d}_{nw}\|_2} = \min_{y\in\mathbf{R}^q} \frac{w_{cp}^T w_{nw}}{\|w_{cp}^T\|_2 \|w_{nw}\|_2} = \min_{y\in\mathbf{R}^q} \frac{y^T \tilde{\Lambda}^{-1} y}{\|y\|_2 \sqrt{y^T \tilde{\Lambda}^{-2} y}}.$$

It should be clear from the geometric description that the largest angle occurs at a point near (but not exactly at) the sharpest corner of the ellipsoid; hence, we may confine the search for the optimum $y$ to the two-dimensional subspace spanned by the largest and smallest eigenvectors of $\tilde{B}$, since this gives the most eccentric ellipse. Remembering that $\tilde{\lambda}_1$ and $\tilde{\lambda}_q$ denote the smallest and largest eigenvalues of $\tilde{B}$, the problem reduces to

$$\cos\tilde{\theta} \;\geq\; \min_{y_1,y_q} \frac{y_1^2/\tilde{\lambda}_1 + y_q^2/\tilde{\lambda}_q}{\sqrt{y_1^2+y_q^2}\sqrt{y_1^2/\tilde{\lambda}_1^2 + y_q^2/\tilde{\lambda}_q^2}}. \tag{1.4.21}$$

Setting the gradient to zero and solving shows that $y_1^2 = (\tilde{\lambda}_1/\tilde{\lambda}_q)y_q^2$. Substituting this into the objective of (1.4.21) and noting that $\kappa_{\tilde{B}} = \tilde{\lambda}_q/\tilde{\lambda}_1$ then gives the desired bound. □

The corresponding angle $\theta$ for problem (1.4.1)-(1.4.2) is different from $\tilde{\theta}$ because the transformation (1.4.3) does not preserve angles between vectors. We can see from the relation

$$\cos\theta = \frac{d_{cp}^T d_{nw}}{\|d_{cp}^T\|_2 \, \|d_{nw}\|_2} = \frac{g^T(M^TM)^{-1}B^{-1}g}{\|(M^TM)^{-1}g\|_2\sqrt{g^TB^{-2}g}}$$

that $\theta$ is always acute, but that its largest value depends on $M^TM$ as well as $\kappa_B$.

The dogleg method is effective for large-scale problems if the Newton point can be computed efficiently. Two possible strategies for calculating $-\tilde{B}^{-1}\tilde{g}$ are the use of sparse Cholesky factors and conjugate gradient (CG) iterations. If the elements of the sparse matrix $\tilde{B}$ are available explicitly, then a sparse Cholesky factorization allows $-\tilde{B}^{-1}\tilde{g}$ to be found by solving two sparse triangular systems of equations. There is a risk of the factors filling in with new nonzero elements, thus requiring an excessive amount of computation time and memory storage. Sometimes the denseness of the factors can be limited by an alternate choice of pivots, though this may also reduce numerical accuracy (see [22], for instance). Conjugate gradient does not require an explicit $\tilde{B}$ matrix, only the ability to multiply the matrix by an arbitrary vector, and the memory requirements are minimal; however, if $\tilde{B}$ is ill-conditioned, then CG may take a huge number of iterations to get close to the Newton point. The remedy for this is preconditioning, but the formation of a good preconditioner usually requires an explicit form for $\tilde{B}$ and intimate knowledge of its structure. In general it is difficult to say whether sparse Cholesky factorization or conjugate gradient will find the Newton point efficiently – in the best cases they each can find a solution with $O(q)$ operations, but in the worst cases they may take $O(q^3)$ operations.

### 1.4.4 2-D Subspace Minimization

Byrd, Schnabel and Schultz proved in [71] that a trust region algorithm can possess excellent convergence properties when the trust region subproblems are solved over certain two-dimensional subspaces; i.e., the search for $d$ in (1.4.1)-(1.4.2) can be restricted, greatly simplifying the subproblem. For instance, if $B$ is positive definite and the Cauchy and Newton points are computed as in the dogleg method, then the two directions $d_{cp}$ and $d_{nw}$ determine a suitable two-dimensional subspace. Solving (1.4.1)-(1.4.2) over this subspace gives a solution at least as good as that of the dogleg method. More importantly, if $B$ is indefinite, then a direction of negative curvature (see [10] for one way to compute such a direction) can take the place of $d_{nw}$ in the two-dimensional subspace; hence, we have an approximate solution method for (1.4.1)-(1.4.2) that works for any matrix $B$.

The computational costs of this method are revealed by writing out the restricted version of the problem. Let $d_\alpha$ and $d_\beta$ denote the two directions that define the two-dimensional subspace. The restricted step $d$ is some linear combination of these directions, so the restricted version of (1.4.1)-(1.4.2) may be written in the form

$$\begin{aligned} \min_{\alpha,\beta\in\mathbf{R}} \quad & g^T d + \frac{1}{2} d^T B d \\ \text{subject to} \quad & d = \alpha d_\alpha + \beta d_\beta \\ & \|Md\|_2 \leq \Delta. \end{aligned}$$

Using the equality constraint to eliminate $d$ reduces this to

$$\begin{aligned} \min_{\alpha,\beta\in\mathbf{R}} \quad & [\alpha \ \ \beta] \begin{bmatrix} g^T d_\alpha \\ g^T d_\beta \end{bmatrix} + \frac{1}{2}[\alpha \ \ \beta] \begin{bmatrix} d_\alpha^T B d_\alpha & d_\alpha^T B d_\beta \\ d_\alpha^T B d_\beta & d_\beta^T B d_\beta \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \\ \text{subject to} \quad & [\alpha \ \ \beta] \begin{bmatrix} d_\alpha^T M^T M d_\alpha & d_\alpha^T M^T M d_\beta \\ d_\alpha^T M^T M d_\beta & d_\beta^T M^T M d_\beta \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \leq \Delta^2. \end{aligned}$$

The spectral expansion techniques described in §1.4.1 can be used to solve for $\alpha$ and $\beta$ with great accuracy because the eigenvalues and eigenvectors of a $2 \times 2$ matrix can be obtained exactly, and the work of solving this reduced problem is much less than $O(q)$ when $q$ is large. The dominant cost of the method is the computation of the terms $d_\alpha^T B d_\alpha, d_\alpha^T B d_\beta, \ldots$ For sparse matrices the work required is $O(q)$ for each of the six terms, and this is probably small compared to the cost of computing the directions $d_\alpha$ and $d_\beta$.

### 1.4.5 Steihaug's Method

Another approximate solution method valid for any symmetric matrix $B$ is an extension of the conjugate gradient method proposed by Steihaug [74] (some elements of this method were also suggested independently by Toint [76]). Conjugate gradient is designed to iterate to the minimizer of $\tilde{\phi}$ in (1.4.5) when $\tilde{B}$ is positive definite, but can break down if $\tilde{B}$ is indefinite because then $\tilde{\phi}$ has no bounded minimizer. By incorporating the trust region constraint (1.4.6), Steihaug extends CG in a well-defined manner to the more general problem.

Algorithm 1.4.8 shows the method for solving the spherically constrained trust region subproblem (1.4.5)-(1.4.6). The intermediate step $\tilde{d}_j$ is updated with each loop of the method until a final approximate solution $\tilde{d}$ is returned. The search directions $\{\tilde{p}_0, \tilde{p}_1, \ldots\}$ are all conjugate with respect to $\tilde{B}$, and each residual $\tilde{r}_j$ is equal to $-\nabla\tilde{\phi}(\tilde{d}_j)$. The differences between Steihaug's method and standard conjugate gradient are the two extra stopping conditions formulated as the first two IF statements within the LOOP. The first one stops the method if a search direction of zero or negative curvature along $\tilde{B}$ is found, and the second one halts iterations if $\tilde{d}_{j+1}$ violates the trust region constraint. In both cases, a final point $\tilde{d}$ is found by intersecting the current search direction with the the trust region boundary.

Notice that the first intermediate step (assuming $\|\tilde{r}_0\|_2 \geq \tilde{\epsilon}$, so that there is a first step) is given by

$$\tilde{d}_1 = \tilde{\alpha}_0 \tilde{p}_0 = \frac{\tilde{r}_0^T \tilde{r}_0}{\tilde{p}_0^T \tilde{B} \tilde{p}_0} \tilde{p}_0 = -\frac{\tilde{g}^T \tilde{g}}{\tilde{g}^T \tilde{B} \tilde{g}} \tilde{g}.$$

**Algorithm 1.4.8** *Steihaug's method for problem (1.4.5)-(1.4.6)*

> Constant $\tilde{\epsilon} > 0$ is given
> Start with $\tilde{d}_0 = \emptyset$, $\tilde{r}_0 = -\tilde{g}$, and $\tilde{p}_0 = \tilde{r}_0$
> IF $\|\tilde{r}_0\|_2 < \tilde{\epsilon}$ THEN RETURN $\tilde{d} = \tilde{d}_0$
> LOOP, starting with $j = 0$
>     IF $\tilde{p}_j^T \tilde{B} \tilde{p}_j \le 0$
>         THEN find $\tilde{\tau}$ so that $\tilde{d} = \tilde{d}_j + \tilde{\tau}\tilde{p}_j$ minimizes $\tilde{\phi}(\tilde{d})$
>             and satisfies $\|\tilde{d}\|_2 = \Delta$, and RETURN $\tilde{d}$
>     $\tilde{\alpha}_j = \tilde{r}_j^T \tilde{r}_j \,/\, \tilde{p}_j^T \tilde{B} \tilde{p}_j$
>     $\tilde{d}_{j+1} = \tilde{d}_j + \tilde{\alpha}_j \tilde{p}_j$
>     IF $\|\tilde{d}_{j+1}\|_2 \ge \Delta$
>         THEN find $\tilde{\tau} \ge 0$ so that $\|\tilde{d}_j + \tilde{\tau}\tilde{p}_j\|_2 = \Delta$,
>             and RETURN $\tilde{d} = \tilde{d}_j + \tilde{\tau}\tilde{p}_j$
>     $\tilde{r}_{j+1} = \tilde{r}_j - \tilde{\alpha}_j \tilde{B} \tilde{p}_j$
>     IF $\|\tilde{r}_{j+1}\|_2 \,/\, \|\tilde{r}_0\|_2 < \tilde{\epsilon}$ THEN RETURN $\tilde{d} = \tilde{d}_{j+1}$
>     $\tilde{\beta}_{j+1} = \tilde{r}_{j+1}^T \tilde{r}_{j+1} \,/\, \tilde{r}_j^T \tilde{r}_j$
>     $\tilde{p}_{j+1} = \tilde{r}_{j+1} + \tilde{\beta}_{j+1} \tilde{p}_j$
> CONTINUE, after incrementing $j$

This is exactly the same as the Cauchy step $\tilde{d}_{cp}$ given by (1.4.17) and the first case of (1.4.18); the second case of (1.4.18) is obtained if Algorithm 1.4.8 cuts $\tilde{d}_1$ short to satisfy the trust region constraint (i.e., when the second IF inside the LOOP is true). Steihaug showed that every iteration reduces $\tilde{\phi}$, so his algorithm fulfills the necessary condition for global convergence described at the end of §1.4.2. Let us prove that $\tilde{\phi}$ is always reduced.

**Theorem 1.4.9** *The sequence of objective values* $\{\tilde{\phi}(\tilde{d}_0), \ldots, \tilde{\phi}(\tilde{d}_j), \tilde{\phi}(\tilde{d}_{j+1}), \ldots, \tilde{\phi}(\tilde{d})\}$ *generated by Algorithm 1.4.8 is monotonically decreasing.*

*Proof.* First we must recall the well-known expanding subspace property of the conjugate gradient method, which states that the residual $\tilde{r}_j$ is orthogonal to the set of conjugate search directions $\{\tilde{p}_0, \ldots, \tilde{p}_{j-1}\}$ (see [54, pp. 241–242], for instance). Thus,

$$\tilde{r}_j^T \tilde{p}_j = \tilde{r}_j^T (\tilde{r}_j + \tilde{\beta}_j \tilde{p}_{j-1}) = \tilde{r}_j^T \tilde{r}_j,$$

and so

$$\tilde{p}_j^T \nabla \tilde{\phi}(\tilde{d}_j) = -\tilde{p}_j^T \tilde{r}_j = -\tilde{r}_j^T \tilde{r}_j,$$

which shows that $\tilde{p}_j$ is a descent direction at $\tilde{d}_j$. The next point generated after $\tilde{d}_j$ is either the iterate $\tilde{d}_{j+1}$ or the final step $\tilde{d}$, and is always some distance (say $\alpha$) along the search direction $\tilde{p}_j$. First, assume that $\tilde{p}_j^T \tilde{B} \tilde{p}_j > 0$ and parameterize the possible values of $\tilde{\phi}$ at the next point in terms of $\alpha$:

$$\begin{aligned}\tilde{\phi}(\tilde{d}_j + \alpha\tilde{p}_j) &= \tilde{g}^T\tilde{d}_j + \alpha\tilde{g}^T\tilde{p}_j + \frac{1}{2}\tilde{d}_j^T\tilde{B}\tilde{d}_j + \alpha\tilde{d}_j^T\tilde{B}\tilde{p}_j + \frac{1}{2}\alpha^2\tilde{p}_j^T\tilde{B}\tilde{p}_j \\ &= \tilde{\phi}(\tilde{d}_j) - \alpha\tilde{r}_j^T\tilde{p}_j + \frac{1}{2}\alpha^2\tilde{p}_j^T\tilde{B}\tilde{p}_j.\end{aligned}$$

The rate of change of $\tilde{\phi}$ along $\tilde{p}_j$ with respect to $\alpha$ is then given by

$$\frac{d}{d\alpha}\,\tilde{\phi}(\tilde{d}_j + \alpha\tilde{p}_j) = \alpha\tilde{p}_j^T\tilde{B}\tilde{p}_j - \tilde{r}_j^T\tilde{p}_j.$$

Notice that when $\alpha = \tilde{\alpha}_j$ the derivative vanishes, which implies that $\tilde{\alpha}_j$ is the one-dimensional minimizer in the direction $\tilde{p}_j$. Since $\tilde{r}_j^T\tilde{p}_j = \tilde{r}_j^T\tilde{r}_j > 0$, it follows that the derivative is nonpositive for values of $\alpha \in [0, \tilde{\alpha}_j]$. These values of $\alpha$ are the only ones allowed by Algorithm 1.4.8, so $\tilde{\phi}$ cannot increase on an iteration in which $\tilde{p}_j^T\tilde{B}\tilde{p}_j > 0$.

If $\tilde{p}_j^T\tilde{B}\tilde{p}_j \leq 0$, then the first IF in the LOOP halts Algorithm 1.4.8 and returns $\tilde{d} = \tilde{d}_j + \tilde{\tau}\tilde{p}_j$. The scalar $\tilde{\tau}$ is chosen to make $\|\tilde{d}\|_2 = \Delta$, for which there are two possible values:

$$\tilde{\tau} = \frac{-\tilde{d}_j^T\tilde{p}_j \pm \sqrt{(\Delta^2 - \tilde{d}_j^T\tilde{d}_j)\tilde{p}_j^T\tilde{p}_j + (\tilde{d}_j^T\tilde{p}_j)^2}}{\tilde{p}_j^T\tilde{p}_j}.$$

Now $\Delta^2 > \tilde{d}_j^T\tilde{d}_j$ must be true or the algorithm would have stopped on the previous iteration. Hence, both possible values of $\tilde{\tau}$ are real, with one value positive and the other negative. The objective at $\tilde{d}$ is given by

$$\tilde{\phi}(\tilde{d}_j + \tilde{\tau}\tilde{p}_j) = \tilde{\phi}(\tilde{d}_j) - \tilde{\tau}\tilde{r}_j^T\tilde{p}_j + \frac{1}{2}\tilde{\tau}^2\tilde{p}_j^T\tilde{B}\tilde{p}_j,$$

and this can always be made smaller than $\tilde{\phi}(\tilde{d}_j)$ by choosing the value of $\tilde{\tau}$ that has the same sign as $\tilde{r}_j^T\tilde{p}_j$. □

One other crucial property of Steihaug's method is that each iterate is farther away than the previous one. This property means that it is acceptable to stop iterating as soon as the trust region boundary is reached because no further iterates will be inside the trust region giving some better minimum. The proof of this property first requires a fundamental lemma.

**Lemma 1.4.10** *The sequences of vectors generated by Algorithm 1.4.8 satisfy*

$$\begin{aligned}&\tilde{d}_j^T\tilde{r}_j = 0 \quad \textit{for} \quad j \geq 0, \quad \textit{and} \\ &\tilde{d}_j^T\tilde{p}_j > 0 \quad \textit{for} \quad j \geq 1.\end{aligned}$$

*Proof.* Algorithm 1.4.8 computes $\tilde{d}_{j+1}$ recursively in terms of $\tilde{d}_j$, but when all the terms of this recursion are written explicitly we can see that

$$\tilde{d}_j = \tilde{d}_0 + \sum_{i=0}^{j-1}\tilde{\alpha}_i\tilde{p}_i = \sum_{i=0}^{j-1}\tilde{\alpha}_i\tilde{p}_i.$$

Multiplying by $\tilde{r}_j$ and applying the expanding subspace property cited earlier in Theorem 1.4.9 gives

$$\tilde{d}_j^T \tilde{r}_j = \sum_{i=0}^{j-1} \tilde{\alpha}_i \tilde{p}_i^T \tilde{r}_j = 0,$$

proving the first part of this theorem.

An induction proof establishes the second part. The induction begins with

$$\tilde{d}_1^T \tilde{p}_1 = (\tilde{\alpha}_0 \tilde{p}_0)^T (\tilde{r}_1 + \tilde{\beta}_1 \tilde{p}_0) = \tilde{\alpha}_0 \tilde{\beta}_1 \, \tilde{p}_0^T \tilde{p}_0 > 0,$$

again employing the expanding subspace property. Now make the inductive hypothesis that $\tilde{d}_j^T \tilde{p}_j > 0$ and show that this implies $\tilde{d}_{j+1}^T \tilde{p}_{j+1} > 0$. This is easily accomplished by using the newly proved fact that $\tilde{d}_{j+1}^T \tilde{r}_{j+1} = 0$:

$$\begin{aligned} \tilde{d}_{j+1}^T \tilde{p}_{j+1} &= \tilde{d}_{j+1}^T (\tilde{r}_{j+1} + \tilde{\beta}_{j+1} \tilde{p}_j) \\ &= \tilde{\beta}_{j+1} \tilde{d}_{j+1}^T \tilde{p}_j \\ &= \tilde{\beta}_{j+1} (\tilde{d}_j + \tilde{\alpha}_j \tilde{p}_j)^T \tilde{p}_j \\ &= \tilde{\beta}_{j+1} \tilde{d}_j^T \tilde{p}_j + \tilde{\alpha}_j \tilde{\beta}_{j+1} \tilde{p}_j^T \tilde{p}_j. \end{aligned}$$

The inductive hypothesis makes the last expression positive. □

**Theorem 1.4.11** *The sequence of vectors generated by Algorithm 1.4.8 satisfy*

$$0 = \|\tilde{d}_0\|_2 < \cdots < \|\tilde{d}_j\|_2 < \|\tilde{d}_{j+1}\|_2 < \cdots < \|\tilde{d}\|_2 \leq \Delta.$$

*Proof.* If the algorithm stops because $\tilde{p}_j^T \tilde{B} \tilde{p}_j \leq 0$ or $\|\tilde{d}_{j+1}\|_2 \geq \Delta$, then the final point $\tilde{d}$ is chosen to make $\|\tilde{d}\|_2 = \Delta$, the largest possible length any point can have. To cover all other possibilities in the algorithm we must show that $\|\tilde{d}_j\|_2 < \|\tilde{d}_{j+1}\|_2$ when $\tilde{d}_{j+1} = \tilde{d}_j + \tilde{\alpha}_j \tilde{p}_j$ and $j \geq 1$. Observe that

$$\|\tilde{d}_{j+1}\|_2^2 = (\tilde{d}_j + \tilde{\alpha}_j \tilde{p}_j)^T (\tilde{d}_j + \tilde{\alpha}_j \tilde{p}_j) = \|\tilde{d}_j\|_2^2 + 2\tilde{\alpha}_j \tilde{d}_j^T \tilde{p}_j + \tilde{\alpha}_j^2 \|\tilde{p}_j\|_2^2.$$

Lemma 1.4.10 ensures that $\|\tilde{d}_j\|_2^2$ is always smaller than $\|\tilde{d}_{j+1}\|_2^2$. □

¿From this theorem we see that the iterates of Steihaug's method sweep out points $\tilde{d}_j$ that move on some interpolating path from $\tilde{d}_1$ to $\tilde{d}$, a path in which every step increases its total distance from the start point. When $\tilde{B}$ is positive definite, this path may be compared to the path of Powell's dogleg method, because both methods move from $\tilde{d}_{cp}$ to $\tilde{d}_{nw}$ (disregarding the trust region constraint). In this sense, Steihaug characterizes his method as a multi-dimensional generalization of the dogleg method.

When $\tilde{B}$ is indefinite, the analysis in §1.4.1 following equation (1.4.15) shows that the optimal solution $\tilde{d}_*$ usually has nonzero projections on the eigenspaces of $\tilde{B}$ corresponding to negative eigenvalues. It is not clear whether Steihaug's method is able to accurately approximate this result, because the method stops as soon as a search direction $\tilde{p}_j$ is generated such that $\tilde{p}_j^T \tilde{B} \tilde{p}_j \leq 0$. This could happen before projections on the negative eigenspaces are as large as those for $\tilde{d}_*$; that is, Steihaug's method could terminate prematurely with a poor approximation to the exact solution. The following theorem lends some insight by showing that each nonzero projection on a negative eigenspace of $\tilde{B}$ tends to be increased by every iteration of Algorithm 1.4.8.

**Theorem 1.4.12** *Let the eigenvalues of $\tilde{B}$ be indexed in the order $\tilde{\lambda}_1 \leq \cdots \leq \tilde{\lambda}_q$ and let $\{\tilde{\nu}_1, \ldots, \tilde{\nu}_q\}$ be a corresponding set of orthonormal eigenvectors. If $\tilde{\lambda}_i < 0$, then*

$$|\tilde{\nu}_i^T \tilde{r}_{j+1}| > |\tilde{\nu}_i^T \tilde{r}_j| \quad \textit{and} \quad |\tilde{\nu}_i^T \tilde{p}_{j+1}| > |\tilde{\nu}_i^T \tilde{r}_j|, \quad \textit{for all} \quad j \geq 0.$$

*Proof.* As in §1.4.1, define the matrices $\tilde{Q} = [\tilde{\nu}_1 \cdots \tilde{\nu}_q]$ and $\tilde{\Lambda} = diag\{\tilde{\lambda}_1, \ldots, \tilde{\lambda}_q\}$. Premultiply the relation $\tilde{r}_{j+1} = \tilde{r}_j - \tilde{\alpha}_j \tilde{B} \tilde{p}_j$ by $\tilde{Q}^T$ and make use of the identities $\tilde{B}\tilde{Q} = \tilde{Q}\tilde{\Lambda}$ and $\tilde{Q}^T \tilde{Q} = I$ to obtain

$$\tilde{Q}^T \tilde{r}_{j+1} = \tilde{Q}^T \tilde{r}_j - \tilde{\alpha}_j \tilde{\Lambda} (\tilde{Q}^T \tilde{p}_j).$$

The $i$th row of this vector equation is

$$\tilde{\nu}_i^T \tilde{r}_{j+1} = \tilde{\nu}_i^T \tilde{r}_j - \tilde{\alpha}_j \tilde{\lambda}_i \tilde{\nu}_i^T \tilde{p}_j. \tag{1.4.22}$$

Substituting for $\tilde{r}_{j+1}$ from the relation $\tilde{p}_{j+1} = \tilde{r}_{j+1} + \tilde{\beta}_{j+1} \tilde{p}_j$ then gives

$$\tilde{\nu}_i^T \tilde{p}_{j+1} = \tilde{\nu}_i^T \tilde{r}_j + (\tilde{\beta}_{j+1} - \tilde{\alpha}_j \tilde{\lambda}_i) \tilde{\nu}_i^T \tilde{p}_j. \tag{1.4.23}$$

Now an induction argument is used to establish that the projections $\tilde{\nu}_i^T \tilde{r}_j$ and $\tilde{\nu}_i^T \tilde{p}_j$ have the same algebraic sign when $\tilde{\lambda}_i < 0$. If $j = 0$, then $\tilde{r}_0 = \tilde{p}_0$ and obviously $\tilde{\nu}_i^T \tilde{r}_0$ and $\tilde{\nu}_i^T \tilde{p}_0$ have the same sign. Making the inductive hypothesis that $\tilde{\nu}_i^T \tilde{r}_j$ has the same sign as $\tilde{\nu}_i^T \tilde{p}_j$, we must prove that this implies $\tilde{\nu}_i^T \tilde{r}_{j+1}$ and $\tilde{\nu}_i^T \tilde{p}_{j+1}$ also have the same sign. Equations (1.4.22) and (1.4.23) clearly show this result for $\tilde{\lambda}_i < 0$, since the right side of each equation is a sum of two terms with the same sign. By the same reasoning, equation (1.4.22) gives the first inequality of this theorem, and equation (1.4.23) gives the second one. The induction argument proves these results hold for all $j \geq 0$. □

To emphasize the significance of this theorem, it states that each nonzero projection of a search direction along a negative eigenspace of $\tilde{B}$ has its magnitude bounded from below by a strictly increasing sequence. Therefore, $|\tilde{\nu}_i^T \tilde{p}_j|$ tends to increase with $j$ and the expression

$$\tilde{p}_j^T \tilde{B} \tilde{p}_j = \sum_{i=1}^{q} \tilde{\lambda}_i (\tilde{\nu}_i^T \tilde{p}_j)^2$$

will eventually be made nonpositive by the negative eigenvalue terms, unless the trust region boundary is encountered first. This may be good or bad, depending on the problem. If $\tilde{p}_0 = -\tilde{g}$ has a small nonzero projection along a negative eigenspace of $\tilde{B}$, then the optimal solution $\tilde{d}_*$ also has a nonzero projection on the same eigenspace; in this case Theorem 1.4.12

suggests that a small initial projection will not be lost in truncation errors, which is good. But if $\tilde{p}_0$ is orthogonal to all negative eigenspaces, then $\tilde{d}_*$ may also be orthogonal. Now Theorem 1.4.12 implies that a false negative projection introduced by roundoff error into some search direction will be magnified by the iterations of Algorithm 1.4.8, possibly becoming large enough to prematurely halt the algorithm. The importance of these hypothesized phenomenon in practice is not known.

We may determine the form of Steihaug's method for solving the ellipsoidally constrained trust region subproblem (1.4.1)-(1.4.2) by applying the transformations (1.4.3) and (1.4.7) to Algorithm 1.4.8, and additionally defining the vectors

$$r_j \equiv (M^T M)^{1/2} \tilde{r}_j \quad \text{and} \quad p_j \equiv (M^T M)^{-1/2} \tilde{p}_j. \tag{1.4.24}$$

The result after making these substitutions is Algorithm 1.4.13. The residuals $r_j$ here equal $-\nabla \phi(d_j)$ and the search directions $\{p_0, p_1, \ldots\}$ generated by Algorithm 1.4.13 are conjugate with respect to $B$. The two extra stop tests introduced by Steihaug now enforce the ellipsoidal trust region constraint (1.4.2). All the search directions are scaled by the factor $(M^T M)^{-1}$, the same scaling factor that was derived for the Cauchy point in equation (1.4.19).

**Algorithm 1.4.13** *Steihaug's method for problem (1.4.1)-(1.4.2)*

Constant $\epsilon > 0$ is given
Start with $d_0 = \emptyset$, $r_0 = -g$, and $p_0 = (M^T M)^{-1} r_0$
IF $\sqrt{r_0^T (M^T M)^{-1} r_0} < \epsilon$ THEN RETURN $d = d_0$
LOOP, starting with $j = 0$
  IF $p_j^T B p_j \le 0$
    THEN find $\tau$ so that $d = d_j + \tau p_j$ minimizes $\phi(d)$
      and satisfies $\|Md\|_2 = \Delta$, and RETURN $d$
  $\alpha_j = r_j^T (M^T M)^{-1} r_j \,/\, p_j^T B p_j$
  $d_{j+1} = d_j + \alpha_j p_j$
  IF $\|M d_{j+1}\|_2 \ge \Delta$
    THEN find $\tau \ge 0$ so that $\|M(d_j + \tau p_j)\|_2 = \Delta$,
      and RETURN $d = d_j + \tau p_j$
  $r_{j+1} = r_j - \alpha_j B p_j$
  IF $\sqrt{r_{j+1}^T (M^T M)^{-1} r_{j+1}} \,/\, \sqrt{r_0^T (M^T M)^{-1} r_0} < \epsilon$ THEN RETURN $d = d_{j+1}$
  $\beta_{j+1} = r_{j+1}^T (M^T M)^{-1} r_{j+1} \,/\, r_j^T (M^T M)^{-1} r_j$
  $p_{j+1} = (M^T M)^{-1} r_{j+1} + \beta_{j+1} p_j$
CONTINUE, after incrementing $j$

Theorems 1.4.9 and 1.4.11 (as well as Lemma 1.4.10) have direct analogues for Algorithm 1.4.13. The objective function $\phi(d_j)$ is decreased with each iteration, and steps satisfy the inequalities

$$0 = \|Md_0\|_2 < \cdots < \|Md_j\|_2 < \|Md_{j+1}\|_2 < \cdots < \|Md\|_2 \leq \Delta.$$

These properties may be established by making substitutions from (1.4.3), (1.4.7) and (1.4.24) directly into the proofs already given for Algorithm 1.4.8.

### 1.4.6 Steihaug's Method with Preconditioning

Conjugate gradient is well-suited for large problems because it stores only a few vectors and requires only sparse matrix-vector products and vector-vector products. These same advantages hold for Steihaug's method (assuming $M^T M$ can be cheaply inverted). Typically, for CG to obtain a large reduction in the size of the residuals with only a small number of iterations a preconditioner is required. This section shows that a rigorously proper extension of Steihaug's method to incorporate preconditioning is not possible. Instead, an alternative hybrid method is developed, the usefulness of which may be limited by certain theoretical shortcomings.

Let us begin by reviewing preconditioning of the conjugate gradient method. Assume our task is to minimize the objective (1.4.1) with $B$ positive definite, using a preconditioner $C$. For the moment, the trust region constraint (1.4.2) is ignored. (The following derivation parallels that of [41, pp. 527–529].) The transformations

$$d \equiv C^{-1/2}\hat{d}, \quad g \equiv C^{1/2}\hat{g}, \quad \text{and} \quad B \equiv C^{1/2}\hat{B}\,C^{1/2}$$

(a nonsingular symmetric square root exists for every positive definite symmetric matrix $C$) define the preconditioned objective $\hat{g}^T\hat{d} + \frac{1}{2}\hat{d}^T\hat{B}\hat{d}$. Algorithm 1.4.14 shows the method of conjugate gradient for minimizing this preconditioned problem. Additionally defining the vectors

$$d_j \equiv C^{-1/2}\hat{d}_j, \quad r_j \equiv C^{1/2}\hat{r}_j, \quad \text{and} \quad p_j \equiv C^{-1/2}\hat{p}_j$$

produces Algorithm 1.4.15, the method of preconditioned conjugate gradient for minimizing the original objective $g^T d + \frac{1}{2}d^T Bd$. In particular, note that when the ideal preconditioner $C = B$ is used, then $\hat{B} = I$, $\hat{d}_* = -\hat{g}$, and $d_* = -B^{-1}g$.

Now let us bring the trust region constraint back into the picture and see what happens. If the constraint is spherical in the preconditioned problem so that $\|\hat{d}\|_2 \leq \Delta$, then Algorithm 1.4.14 generalizes into Steihaug's method by simply adding the two extra stop tests as they appear in Algorithm 1.4.8. The preconditioning transformation of this spherical constraint gives the ellipsoidal trust region constraint

$$\sqrt{d^T Cd} \leq \Delta, \quad \text{or} \quad \|C^{1/2}d\| \leq \Delta.$$

We see that adding the two transformed stop tests to Algorithm 1.4.15 produces an algorithm identical to Algorithm 1.4.13, Steihaug's method for ellipsoidal constraints, if $C^{1/2}$ is replaced by $M$.

**Algorithm 1.4.14** *Conjugate gradient method on preconditioned system*

Start with $\hat{d}_0 = \emptyset$, $\hat{r}_0 = -\hat{g}$, and $\hat{p}_0 = \hat{r}_0$
IF $\|\hat{r}_0\|_2 < \epsilon$ THEN RETURN $\hat{d} = \hat{d}_0$
LOOP, starting with $j = 0$
$\quad\hat{\alpha}_j = \hat{r}_j^T \hat{r}_j \,/\, \hat{p}_j^T \hat{B} \hat{p}_j$
$\quad\hat{d}_{j+1} = \hat{d}_j + \hat{\alpha}_j \hat{p}_j$
$\quad\hat{r}_{j+1} = \hat{r}_j - \hat{\alpha}_j \hat{B} \hat{p}_j$
$\quad$IF $\|\hat{r}_{j+1}\|_2 \,/\, \|\hat{r}_0\|_2 < \epsilon$ THEN RETURN $\hat{d} = \hat{d}_{j+1}$
$\quad\hat{p}_{j+1} = \hat{r}_{j+1} + (\hat{r}_{j+1}^T \hat{r}_{j+1} \,/\, \hat{r}_j^T \hat{r}_j) \hat{p}_j$
CONTINUE, after incrementing $j$

**Algorithm 1.4.15** *Preconditioned conjugate gradient method for minimizing (1.4.1)*

Start with $d_0 = \emptyset$, $r_0 = -g$, and $p_0 = C^{-1} r_0$
IF $\sqrt{r_0^T C^{-1} r_0} < \epsilon$ THEN RETURN $d = d_0$
LOOP, starting with $j = 0$
$\quad\alpha_j = r_j^T C^{-1} r_j \,/\, p_j^T B p_j$
$\quad d_{j+1} = d_j + \alpha_j p_j$
$\quad r_{j+1} = r_j - \alpha_j B p_j$
$\quad$IF $\sqrt{r_{j+1}^T C^{-1} r_{j+1}} \,/\, \sqrt{r_0^T C^{-1} r_0} < \epsilon$ THEN RETURN $d = d_{j+1}$
$\quad p_{j+1} = C^{-1} r_{j+1} + (r_{j+1}^T C^{-1} r_{j+1} \,/\, r_j^T C^{-1} r_j) p_j$
CONTINUE, after incrementing $j$

Usually the constraint of the preconditioned problem will *not* be spherical because a preconditioner is chosen to match the objective, not the constraint. For instance, if (1.4.5)-(1.4.6) is the original problem, then any preconditioner will transform (1.4.6) into an ellipsoidal constraint. By analogy with the development of Algorithm 1.4.15 from 1.4.14 just discussed, a preconditioned Steihaug method in this case should be given by applying Algorithm 1.4.13 to the ellipsoidally constrained preconditioned problem, then transforming this back into the original coordinates of (1.4.5)-(1.4.6); but in doing this the two operations cancel each other out! The scaling of search directions induced by a preconditioner (as seen in Algorithm 1.4.15) is undone by the scaling of the ellipsoidal constraint generated by the preconditioner. In other words, the power of the preconditioner to better condition a problem is cancelled by the requirement to scale solutions in conformity with the modified trust region constraint. Therefore, to preserve the preconditioning of the objective, we must drop the ellipsoidal scaling; that is, ignore the constraint of the preconditioned problem until the transformation to original coordinates is completed, then impose the original trust region constraint.

The result, Algorithm 1.4.16, is a hybrid method. The scaling of search directions is caused by the preconditioner, while the two extra stop tests reflect the trust region constraint. Although this algorithm captures the advantages of preconditioning, it suffers from two theoretical defects. One is that the step $d_1$ no longer equals the Cauchy step (1.4.19). If Algorithm 1.4.16 stops with $d = d_1$ (because $\Delta$ is small), then there is no assurance that this solution makes as much reduction in $\phi$ as $d_{cp}$ does; hence, the global convergence of a trust region algorithm may be lost. In practice, this should not be a problem if $C$ is truly a good preconditioner.

**Algorithm 1.4.16** *Hybrid method for solving (1.4.1)-(1.4.2) with preconditioner C*

Constant $\epsilon > 0$ is given
Start with $d_0 = \emptyset$, $r_0 = -g$, and $p_0 = C^{-1}r_0$
IF $\sqrt{r_0^T C^{-1} r_0} < \epsilon$ THEN RETURN $d = d_0$
LOOP, starting with $j = 0$
  IF $p_j^T B p_j \le 0$
    THEN find $\tau$ so that $d = d_j + \tau p_j$ minimizes $\phi(d)$
      and satisfies $\|Md\|_2 = \Delta$, and RETURN $d$
  $\alpha_j = r_j^T C^{-1} r_j \,/\, p_j^T B p_j$
  $d_{j+1} = d_j + \alpha_j p_j$
  IF $\|M d_{j+1}\|_2 \ge \Delta$
    THEN find $\tau \ge 0$ so that $\|M(d_j + \tau p_j)\|_2 = \Delta$,
      and RETURN $d = d_j + \tau p_j$
  $r_{j+1} = r_j - \alpha_j B p_j$
  IF $\sqrt{r_{j+1}^T C^{-1} r_{j+1}} \,/\, \sqrt{r_0^T C^{-1} r_0} < \epsilon$ THEN RETURN $d = d_{j+1}$
  $\beta_{j+1} = r_{j+1}^T C^{-1} r_{j+1} \,/\, r_j^T C^{-1} r_j$
  $p_{j+1} = C^{-1} r_{j+1} + \beta_{j+1} p_j$
CONTINUE, after incrementing $j$

The second flaw of Algorithm 1.4.16 is that steps no longer increase in size with the ellipsoidal norm; instead, they increase according to

$$0 = \|C^{1/2} d_0\|_2 < \cdots < \|C^{1/2} d_j\|_2 < \cdots.$$

Now there is no guarantee that a step which leaves the trust region will not move back inside on a later iterate; hence, Algorithm 1.4.16 may terminate prematurely with a poor estimate of $d_*$.

On the brighter side, when $B$ is positive definite and the Newton point $d_{nw} = -B^{-1}g$ lies far enough inside the trust region that the constraint is never encountered by an iterate, then Algorithm 1.4.16 is superior to Algorithm 1.4.13 by virtue of its preconditioner. Both methods reach a point close to $d_{nw}$, but Algorithm 1.4.16 should do so with fewer iterations. Note also that for the special case of $M^T M = C$, Algorithm 1.4.16 is identical to Algorithm 1.4.13. This suggests that if $C$ can be chosen to resemble the ellipsoidal shape and orientation of $M^T M$, then the impact of the defects of Algorithm 1.4.13 might be minimal.

A final problem with preconditioning Steihaug's method is the possible indefiniteness of $B$. A preconditioner must be symmetric and positive definite to act as a real transformation to better-conditioned coordinates (an indefinite preconditioner transforms to imaginary coordinates). If $B$ has negative eigenvalues then it becomes a more complicated task to design an appropriate positive definite preconditioner for Algorithm 1.4.16, because the negative eigenvalues are usually an essential part of the exact solution. The ideal preconditioner for (1.4.1)-(1.4.2) is $C = B + \mu_* M^T M$, making the first step satisfy (1.4.14), but of course $\mu_*$ is unknown. At the very least we must conclude that finding a preconditioner for Steihaug's method is more complicated than preconditioning conjugate gradient.

## 1.5 Implementation of the Byrd-Omojokun Algorithm

The preceding section surveyed a number of practical methods for efficiently solving large-scale trust region subproblems. Armed with these techniques, we return in this section to develop a detailed implementation of the Byrd-Omojokun algorithm that was presented in §1.3.3. This implementation follows the outline of Algorithm 1.3.2, but does much more than merely designate some method from the preceding section to solve each of the large-scale trust region subproblems in the algorithm. There are many complex interactions between the subproblems of Algorithm 1.3.2 that must be considered to construct an efficient and robust implementation.

In [61], Omojokun devised his own implementation for Algorithm 1.3.2 to conduct numerical tests on small problems. His implementation used dense matrix operations, relying in particular on QR factorizations of the constraint Jacobian $A_k^T$. While robust and effective for small problems, these techniques are not suitable for general large-scale problems. The implementation described in this section began with the work of Lalee [50] and was completed by the author. For convenience, the implementation is often referred to as "ETR", an abbreviation for "Equality constrained optimization using Trust Regions".

### 1.5.1 Solving the Vertical Subproblem

To find the vertical step, let us replace (1.3.14)-(1.3.15) with the equivalent problem

$$\min_{v \in \mathbf{R}^n} \quad c_k^T A_k^T v + \frac{1}{2} v^T A_k A_k^T v \tag{1.5.1}$$

$$\text{subject to} \qquad \|v\|_2 \leq \zeta \Delta_k, \tag{1.5.2}$$

and remember that the solution $v_k$ is to lie in the range space of $A_k$ (to make it orthogonal to the horizontal step). It is not hard to show that such a solution always exists. From (1.4.8), the exact solution $v_*$ of (1.5.1)-(1.5.2) must satisfy

$$(A_k A_k^T + \mu_* I) v_* = -A_k c_k, \quad \text{for some} \quad \mu_* \geq 0. \tag{1.5.3}$$

Now if $\mu_* = 0$, then $v_* = -A_k (A_k^T A_k)^{-1} c_k$ is a solution which clearly lies in the range space of $A_k$. On the other hand, if $\mu_* > 0$ then premultiplying both sides of (1.5.3) by $Z_k^T$ shows that $v_*$ has no component orthogonal to the columns of $A_k$, because $Z_k^T v_* = \emptyset$. ETR uses either the dogleg method or Steihaug's method to compute an approximate vertical step $v_k$, and in both cases this step is also in the range space of $A_k$.

Comparing (1.5.1)-(1.5.2) with the general trust region subproblem formulation (1.4.5)-(1.4.6), we see that $\tilde{g} = A_k c_k$, $\Delta = \zeta \Delta_k$, and that $\tilde{B} = A_k A_k^T$ is a positive semi-definite matrix. Substituting into (1.4.17)-(1.4.18), we obtain the Cauchy point

$$v_{cp} = -\alpha A_k c_k, \quad \text{with} \quad \alpha = \min \left\{ \frac{\zeta \Delta_k}{\|A_k c_k\|_2} \, , \, \frac{\|A_k c_k\|_2^2}{(A_k c_k)^T A_k A_k^T (A_k c_k)} \right\}.$$

To formulate a dogleg method for (1.5.1)-(1.5.2) we need to define an appropriate Newton step as the minimizer of (1.5.1). Since the matrix $A_k A_k^T$ has $n - m$ linearly independent null vectors, there is actually a whole manifold of minimizers for (1.5.1), each satisfying the linearized constraint equations $A_k^T v + c_k = \emptyset$. In the implementation the shortest step is chosen, uniquely given by

$$v_{nw} = -A_k (A_k^T A_k)^{-1} c_k. \tag{1.5.4}$$

Together, $v_{cp}$ and $v_{nw}$ define a dogleg method for the vertical subproblem.

At the end of §1.4.3 two efficient methods were discussed for inverting $A_k^T A_k$ to obtain $v_{nw}$: sparse Cholesky factorization and preconditioned conjugate gradient. Both methods are provided as options in ETR, allowing the user to decide which gives the fastest way of solving a particular problem. Harwell library subroutine MA27 [48] is used to compute a sparse Cholesky factorization after $A_k^T A_k$ is explicitly formed. At the start of the Byrd-Omojokun algorithm, the Harwell code constructs and stores a symbolic factorization of $A_0^T A_0$ based on its choice of pivots. Since the sparsity pattern of $A_k^T A_k$ remains the same on all iterations, the stored symbolic factorization can be efficiently reused to calculate $v_{nw}$ on subsequent iterations of the algorithm. But there is a danger in doing this, because the values of $A_k^T A_k$ may change so much as $k$ increases that the choice of pivots at $k = 0$ yields an ill-conditioned factorization; in this event MA27 should be allowed to look for a better pivoting sequence and construct a new symbolic factorization. To minimize the occurrence of this expensive operation, ETR allows a new symbolic factorization by MA27 only when the condition number of $A_k^T A_k$ exceeds $10^4$ and the Byrd-Omojokun algorithm is not taking successful Newton steps (this latter condition is to prevent interference with good algorithm behavior). A lower bound for the condition number of $A_k^T A_k$ is obtained by computing the angle between $v_{cp}$ and $v_{nw}$ at each iteration. If the angle exceeds 89.0° (see Theorem 1.4.7 for the derivation of this number), then MA27 looks for new pivots.

Computing $v_{nw}$ with Cholesky factors works well unless the factors become dense. This can happen during the formation of $A_k^T A_k$ (for instance, a single dense row in $A_k$ makes $A_k^T A_k$ completely dense), or during its factorization by MA27. Finding $v_{nw}$ with conjugate gradient circumvents these problems, and also allows some control in the accuracy of the calculation and thus control over the number of operations executed. The usual CG stopping condition (see Algorithm 1.4.14) is that the norm of the current residual be less than some small fraction of the norm of the starting residual. In (1.5.4) the starting residual is $c_k$. Since this goes to zero as $k$ increases, the stop tolerance of $\epsilon \|c_k\|_2$ can reduce computational work by using a relatively large value for $\epsilon$ (provided that $A_k$ is not ill-conditioned), yet not impede the asymptotic convergence rate of the Byrd-Omojokun algorithm (see [18]). Preconditioning can further improve the efficiency of the CG calculation, but it is difficult to construct preconditioners for $A_k^T A_k$ from knowledge of $A_k$, and the current software provides only a diagonal preconditioner.

The computation of $v_{nw}$ by either technique can suffer from roundoff errors when $A_k$ is nearly singular. Motivated by Theorem 1.4.4, the implementation safeguards the dogleg method by disregarding $v_{nw}$ and setting $v_k = v_{cp}$ if the calculated value $\|v_{nw}\|_2$ is smaller than $\|v_{cp}\|_2$.

Steihaug's method in the form of Algorithm 1.4.8 can also be applied to the vertical subproblem. The $n \times n$ Hessian $\tilde{B} = A_k A_k^T$ is of rank $m$, but it is apparent from Algorithm 1.4.8 that all search directions $\tilde{p}_j$ lie in the $m$-dimensional range space of $A_k$;

therefore, minimization is confined to an $m$-dimensional Krylov subspace and Steihaug's method should terminate in at most $m$ iterations (assuming exact arithmetic). For the vertical subproblem, Steihaug's method can truly be called a generalization of the dogleg method. No preconditioning has been tried for Steihaug's method, but even without it this method for computing $v_k$ is competitive with the dogleg method on many test problems.

### 1.5.2 Computing $Z_k$

Recall that $Z_k$ is the $n \times (n-m)$ matrix spanning the linear manifold tangent to all constraints at $x_k$; in particular, it satisfies $A_k^T Z_k = \emptyset$. A numerically stable way of calculating $Z_k$ from $A_k$ is by means of the QR factorization, which ensures that the columns of $Z_k$ are orthogonal. Although a QR factorization might remain sparse, it is likely to fill in, and a nearly dense $Z_k$ matrix could destroy the efficiency of the Byrd-Omojokun algorithm for solving large problems. The idea pursued in ETR is that of direct elimination (see for example [34, p. 234]). First, partition $A_k^T$ into

$$A_k^T = [B_k \;\; N_k], \tag{1.5.5}$$

where the $m \times m$ basis matrix $B_k$ is nonsingular (for simplicity (1.5.5) assumes that the basis $B_k$ is formed by the first $m$ columns of $A_k^T$, but in general some permutation of columns may be necessary). Now define $Z_k$ to be

$$Z_k = \begin{bmatrix} -B_k^{-1} N_k \\ I \end{bmatrix}, \tag{1.5.6}$$

which satisfies $A_k^T Z_k = \emptyset$, although the columns of $Z_k$ are not orthogonal. ETR uses Harwell subroutine MA28 [25] to first choose $m$ linearly independent columns of $A_k^T$ for defining the basis (permuting columns as needed), and then to compute the sparse LU factorization of $B_k$. MA28 selects the basis by considering both the resulting sparsity and the size of the pivots in the LU factorization of $B_k$. Forming $Z_k$ explicitly from this factorization is not advisable because $B_k^{-1} N_k$ requires $n-m$ back-solves to compute, and because $Z_k$ could be dense. Instead, the implicit representation (1.5.6) of $Z_k$ is employed to compute matrix-vector products of the form $Z_k u$ and $Z_k^T x$. For example, $y = Z_k u$ is obtained by computing $b = N_k u$, and then solving $B_k y = -b$ using the sparse LU factors of $B_k$.

MA28 constructs and stores a symbolic LU factorization to take advantage of situations where matrix values change but the sparsity pattern remains the same. This will be the case for $B_k$ in the Byrd-Omojokun algorithm as long as the same $m$ columns are used to serve as the basis matrix. At the beginning of the algorithm columns are chosen to define $B_0$ and it is symbolically factored, then for subsequent iterates this same basis and its factorization are used to define $Z_k$, saving many computations. But it is possible for iterates to change enough that $B_k$ becomes ill-conditioned, in which case it is necessary to choose a different set of columns to define the basis. The following heuristic was devised to determine when to change basis and symbolically refactor.

**Heuristic for changing basis.** When solving systems of the form $B_k y = b$ or $B_k^T y = b$ (which arise in the solution of the horizontal subproblem), the ratio $\|b\|_2 / \|y\|_2$ is computed.

This is a number that lies somewhere between the largest and smallest singular values of $B_k$. During the lifetime of a particular choice of basis the heuristic stores the largest and smallest ratios seen, then divides these to form an estimate on the condition number of the current $B_k$. If the estimate becomes greater than $10^2$, then on the next iteration MA28 is allowed to try and choose a better basis and symbolically refactor. To prevent this heuristic from interfering with good algorithm behavior it is overruled, and the basis left unchanged, when the step $d_k$ is accepted and lies inside the trust region (i.e., when the algorithm is taking successful Newton steps). In §1.6.2 this heuristic is justified by its performance on the test problems.

### 1.5.3 Solving the Horizontal Subproblem

For convenience let us define $\bar{g}_k \equiv g_k + W_k v_k$ and $\bar{\Delta}_k^2 \equiv \Delta_k^2 - \|v_k\|_2^2$. Then the horizontal subproblem (1.3.19)-(1.3.20) can be written as

$$\min_{u \in \mathbf{R}^{n-m}} \quad \bar{g}_k^T Z_k u + \frac{1}{2} u^T Z_k^T W_k Z_k u \tag{1.5.7}$$

$$\text{subject to} \qquad \|Z_k u\|_2 \leq \bar{\Delta}_k. \tag{1.5.8}$$

This is a trust region subproblem with ellipsoidal constraint; in terms of the general formulation (1.4.1)-(1.4.2) we have $M = Z_k$, $B = Z_k^T W_k Z_k$, $g = Z_k^T \bar{g}_k$, $p = n$, and $q = n - m$. Using (1.4.19) we see that the Cauchy point is given by

$$u_{cp} = -\alpha \, (Z_k^T Z_k)^{-1} Z_k^T \bar{g}_k, \tag{1.5.9}$$

with $\alpha$ determined by the following rule:

SET $\bar{u} = (Z_k^T Z_k)^{-1} Z_k^T \bar{g}_k$
IF $\bar{u}^T Z_k^T W_k Z_k \bar{u} \leq 0$
  THEN $\alpha = \dfrac{\bar{\Delta}_k}{\|Z_k \bar{u}\|_2}$
  ELSE $\alpha = \min \left\{ \dfrac{\bar{\Delta}_k}{\|Z_k \bar{u}\|_2} \, , \; \dfrac{\bar{g}_k^T Z_k (Z_k^T Z_k)^{-1} Z_k^T \bar{g}_k}{\bar{u}^T Z_k^T W_k Z_k \bar{u}} \right\}$ .

We saw in §1.5.2 that explicitly computing $(Z_k^T Z_k)^{-1}$ can be very expensive, but the cost of multiplying $Z_k$ and $Z_k^T$ by a vector is much lower; therefore, we invert $Z_k^T Z_k$ approximately by the conjugate gradient method. This means that $u_{cp}$ is just an approximation to the Cauchy point, raising the question whether global convergence of the Byrd-Omojokun algorithm is still assured. Analysis in [12] suggests that an approximation to $u_{cp}$ is adequate for global convergence so long as the error in its computation tends to zero as fast as $Z_k^T \bar{g}_k$. Therefore, the stop tolerance in the conjugate gradient iteration used for computing (1.5.9) is set to a small fraction of $\|Z_k^T \bar{g}_k\|_2$.

The full horizontal step should make at least as much reduction in (1.5.7) as the Cauchy point $u_{cp}$ does, and preferably more in order to accelerate convergence. The best choice for improving the horizontal step is to use Steihaug's method. This allows $W_k$ to be indefinite or singular (which is possible using second derivatives), and requires only the ability to multiply $Z_k$ and $Z_k^T$ by vectors. To define Steihaug's method we make the appropriate substitutions from (1.5.7)-(1.5.8) into Algorithm 1.4.13. This will generate a set of iterates $\{u_0, \ldots, u_j, \ldots\}$ belonging to $\mathbf{R}^{n-m}$; the final iterate is then multiplied by $Z_k$ to define the horizontal step.

A close examination reveals that this Steihaug iteration can be made more efficient by directly generating iterates and search directions of the form $Z_k u_j$ and $Z_k p_j$. To see this, note that the matrices $B$ and $M$ appearing in Algorithm 1.4.13 are given in this case by $B = Z_k^T W_k Z_k$ and $M = Z_k$. Thus, it is necessary to first multiply the quantities $u_j$ (or $d_j$ in the notation of Algorithm 1.4.13) and $p_j$ by $Z_k$ at every place they appear in the iteration. Instead of computing $u_j$ and $p_j$ themselves, we can work with $Z_k u_j$ and $Z_k p_j$ directly, saving the cost of multiplying by $Z_k$ at every iteration of Steihaug's method. These ideas are implemented in Algorithm 1.5.1, where the symbols $(Zu)_j$ and $(Zp)_j$ denote the iterates and search directions in the desired form. By construction, all search directions $(Zp)_j$ lie in the null-space of the constraint matrix, i.e., in the span of $Z_k$. Algorithm 1.5.1 also expresses the ellipsoidally scaled residual explicitly as a vector $t_j$ for greater convenience.

**Algorithm 1.5.1** *Method for solving the horizontal subproblem (1.5.7)-(1.5.8)*

Constants $\epsilon_{h_0} > 0$ and $\epsilon_h > 0$ are given
Start with $(Zu)_0 = \emptyset$, $r_0 = -Z_k^T \bar{g}_k$, $t_0 = (Z_k^T Z_k)^{-1} r_0$, and $(Zp)_0 = Z_k t_0$
IF $\sqrt{r_0^T t_0} < \epsilon_{h_0}$
    THEN use (1.5.9) and RETURN the Cauchy step $Z_k u_{cp}$
LOOP, starting with $j = 0$
    IF $(Zp)_j^T W_k (Zp)_j \leq 0$
        THEN find $\tau$ so that $(Zu) = (Zu)_j + \tau (Zp)_j$ minimizes (1.5.7)
            and satisfies $\|(Zu)\|_2 = \bar{\Delta}_k$ , and RETURN $(Zu)$
    $\alpha_j = r_j^T t_j \,/\, (Zp)_j^T W_k (Zp)_j$
    $(Zu)_{j+1} = (Zu)_j + \alpha_j (Zp)_j$
    IF $\|(Zu)_{j+1}\|_2 \geq \bar{\Delta}_k$
        THEN find $\tau \geq 0$ so that $\|(Zu)_j + \tau (Zp)_j\|_2 = \bar{\Delta}_k$
            and RETURN $(Zu)_j + \tau (Zp)_j$
    $r_{j+1} = r_j - \alpha_j Z_k^T W_k (Zp)_j$
    $t_{j+1} = (Z_k^T Z_k)^{-1} r_{j+1}$
    IF $\sqrt{r_{j+1}^T t_{j+1}} < \epsilon_h \sqrt{r_0^T t_0}$ THEN RETURN $(Zu)_{j+1}$
    $\beta_{j+1} = r_{j+1}^T t_{j+1} \,/\, r_j^T t_j$
    $(Zp)_{j+1} = Z_k t_{j+1} + \beta_{j+1} (Zp)_j$
CONTINUE, after incrementing $j$

As in the computation of the Cauchy point (1.5.9), the inverse of $Z_k^T Z_k$ is calculated approximately using a conjugate gradient subiteration. This inner loop is terminated when its residual becomes smaller than a fraction of $\epsilon_h \sqrt{r_0^T t_0}$, which is the stop tolerance in

Algorithm 1.5.1. Since $r_0^T t_0$ is proportional to $\|Z_k^T g_k\|_2^2$, the analysis in [18] shows that this tolerance will give the Byrd-Omojokun algorithm a Q-quadratic rate of convergence in the case when second derivatives are used for $W_k$.

**Unscaled Steihaug method.** The approximate solution to the horizontal subproblem found by Algorithm 1.5.1 turns out to be quite adequate, but the inversion of $Z_k^T Z_k$ by conjugate gradient is very time consuming in many problems. This motivates us to search for less expensive alternatives for computing $u_k$ that preserve the robustness of the trust region algorithm. A successful idea, called the unscaled Steihaug method, is to eliminate the scaling matrix $(Z_k^T Z_k)^{-1}$ from Algorithm 1.5.1, thereby setting $t_{j+1} = r_{j+1}$. This can be viewed as using the preconditioning matrix $Z_k^T Z_k$ in the hybrid Algorithm 1.4.16; however, this "preconditioner" is not chosen for the usual purpose of improving the condition number of $Z_k^T W_k Z_k$, but only to reduce the cost of the iteration.

In the discussion following Algorithm 1.4.16 two theoretical defects of the hybrid method were mentioned: the first step $(Zu)_1$ no longer equals $Z_k u_{cp}$, and the step lengths $\|(Zu)_j\|_2$ do not increase monotonically. Because of the "deconditioning" effect of $Z_k^T Z_k$, both of these defects can be serious problems. Therefore, we should try the unscaled Steihaug method but discard the result if it leaves the trust region, instead going back and solving with Algorithm 1.5.1. To minimize wasted effort, the unscaled Steihaug method is tried only if the last step $d_k$ was accepted and $\Delta_{k+1}$ is the same as $\Delta_k$. In particular, this ensures the method is tried as the iterates converge to a solution and the trust region becomes inactive. Numerical tests comparing both variations of the Steihaug method are given in §1.6.1. The results show that the computational savings of avoiding the inner CG loop to invert $Z_k^T Z_k$ outweigh the extra iterations caused by deconditioning – the unscaled Steihaug method seems to always be faster.

If $W_k$ is positive definite, which is true if $\ell$-BFGS approximations are used instead of second derivatives (see §1.5.4), then we may use the dogleg method to solve the horizontal subproblem. The Newton step is $u_{nw} = -(Z_k^T W_k Z_k)^{-1} Z_k^T \bar{g}_k$. This linear system is best solved by a conjugate gradient iteration, which turns out to be equivalent to the unscaled Steihaug method without its extra stop tests; thus, the dogleg method takes full advantage of the fast "deconditioner". Test results in §1.6.1 show that it indeed executes faster than the unscaled Steihaug iteration, even though the dogleg step is probably a worse approximation to the true solution of the horizontal subproblem.

### 1.5.4 Limited Memory Approximations

ETR supplies an alternative when second derivatives are unavailable or prohibitively expensive to calculate. A limited memory BFGS ($\ell$-BFGS) approximation to $\nabla_x^2 L_k$ is maintained and used in the role of $W_k$. The estimate is initialized with the identity matrix, then updated implicitly as described in [53]. Only information reflecting the last $t$ updates is kept, which requires $O(tn)$ storage locations and $O(tn)$ operations for updating $W_k$ and multiplying $W_k$ by a vector. Limited memory approximations give rise to a linear convergence rate, so there can be a significant loss in performance compared to using second derivatives.

Previous research [38], [53] has shown that the choice $t \in [4, 10]$ works well, and our testing suggests $\ell$-BFGS is a viable option within the framework of this algorithm.

ETR uses the compact representations of limited memory BFGS matrices described in [9]. Every (implicit) BFGS update is of the form

$$W_{k+1} = W_k - \frac{W_k d_k d_k^T W_k}{d_k^T W_k d_k} + \frac{y_k y_k^T}{y_k^T d_k}, \tag{1.5.10}$$

where $d_k = x_{k+1} - x_k$ and $y_k = (g_{k+1} - A_{k+1}\lambda_{k+1}) - (g_k - A_k\lambda_k)$. The matrix remains positive definite when $y_k^T d_k$ is positive, but this condition may not always be true. The standard damping formula of Powell [67] is then employed to modify $y_k$ enough to force $y_k^T d_k$ to be positive:

IF $d_k^T y_k < 0.2\, d_k^T W_k d_k$

THEN $y_k \leftarrow \theta y_k + (1-\theta) W_k d_k, \quad$ where $\quad \theta = \dfrac{0.8\, d_k^T W_k d_k}{d_k^T W_k d_k - d_k^T y_k}$ .

There is also a danger that the last term in (1.5.10) becomes too large, causing $W_{k+1}$ to blow up in magnitude. To avoid this the update is not performed if

$$10^{-8}\, y_k^T y_k \;\geq\; y_k^T d_k.$$

Nocedal and Overton show in [60] that skipping updates with this rule should not degrade the asymptotic convergence rate of the algorithm.

### 1.5.5 Estimating Lagrange Multipliers

At every new iterate $x_{k+1}$ ETR computes least squares Lagrange multiplier estimates $\lambda_{k+1}$ by solving

$$(A_{k+1}^T A_{k+1})\lambda_{k+1} = A_{k+1}^T g_{k+1}. \tag{1.5.11}$$

This linear system is solved by either sparse Cholesky factorization or by the conjugate gradient method (CGLS in this case). The accuracy of the Lagrange multipliers needs to be just good enough to allow proper evaluation of the first-order KKT conditions at the current solution estimate $x_{k+1}$. Therefore, when the conjugate gradient method is used, the iteration is stopped when the residual of (1.5.11) becomes less than a fraction of $\max\{\|c_k\|_\infty, \|g_k - A_k\lambda_k\|_\infty\}$.

An alternative first-order estimate of the multipliers makes use of the basis matrix $B_{k+1}$ described in §1.5.2. If $g_{k+1}$ is partitioned as $[g_{k+1}^B \;\; g_{k+1}^N]^T$ to correspond to $A_{k+1}^T = [B_{k+1} \;\; N_{k+1}]$, then Lagrange multipliers can be estimated by solving

$$B_{k+1}^T \lambda_{k+1} = g_{k+1}^B. \tag{1.5.12}$$

The LU factors of $B_{k+1}$ obtained by MA28 for the representation of $Z_{k+1}$ are readily used to cheaply solve (1.5.12), making it less costly than solving (1.5.11); however, test results in §1.6.4 indicate that (1.5.12) is a less robust estimate at points far from a solution. Since the cost of computing multipliers is small compared to other subproblem costs, ETR uses (1.5.11) as its default.

### 1.5.6 Merit Function

The merit function is used to decide whether the step $d_k$ makes sufficient progress towards the solution of problem (EP). Following Byrd and Omojokun [61],

$$\psi(x,\mu) = f(x) + \mu\|c(x)\|_2 \tag{1.5.13}$$

is chosen as the merit function, where $\mu > 0$ is called the penalty parameter. Since the last term in (1.5.13) is not squared, this merit function is not differentiable. It is, however, exact: in a neighborhood of a solution point $x_*$ with $\mu > \|\lambda_*\|_\infty$, the minimizer of $\psi(x,\mu)$ is precisely $x_*$. The $\ell_2$ norm used to penalize the constraints in the merit function is chosen to be compatible with the $\ell_2$ norm in the formulation (1.3.15) of the vertical step. Note that it is unbiased with respect to the number of constraints.

We must specify how the penalty parameter will be chosen at each iteration of the algorithm. As is commonly done (see for example [47], [68]), ETR first computes a step $d_k$ and then chooses $\mu$ large enough that $d_k$ results in a reduction of a model merit function. To be more precise, let us replace $f$ by the model objective (1.3.3) and linearize the constraints in (1.5.13) to define the model merit function at $x_k$:

$$\tilde{\psi}_k(d,\mu) \equiv d^T g_k + \frac{1}{2} d^T W_k d + \mu\|A_k^T d + c_k\|_2, \tag{1.5.14}$$

where, as before, $W_k$ denotes $\nabla_x^2 L_k$ or an $\ell$-BFGS approximation to it. The predicted reduction p_red in the model merit function $\tilde{\psi}_k$ is defined by

$$\begin{aligned} \text{p\_red} &= \tilde{\psi}_k(0,\mu) - \tilde{\psi}_k(d_k,\mu) \\ &= -d_k^T g_k - \frac{1}{2} d_k^T W_k d_k + \mu\left(\|c_k\|_2 - \|A_k^T v_k + c_k\|_2\right), \end{aligned} \tag{1.5.15}$$

where the expansion $d_k = v_k + Z_k u_k$ and the relation $A_k^T Z_k = \emptyset$ are used. The term inside the parentheses is positive because the dogleg or Steihaug methods used in the vertical step computation reduce the objective $\|A_k^T v + c_k\|_2$ from its initial value $\|c_k\|_2$ at $v = \emptyset$. A trial value for the penalty parameter corresponding to this step $d_k$ is

$$\mu^+ = \max\left\{\mu_k,\ 0.1 + \frac{d_k^T g_k + \frac{1}{2} d_k^T W_k d_k}{\|c_k\|_2 - \|A_k^T v_k + c_k\|_2}\right\}, \tag{1.5.16}$$

where $\mu_k$ is the penalty parameter at the previous iteration (the algorithm starts with $\mu_0 > 0$). It is clear from (1.5.15) that for this value of the penalty parameter the predicted reduction p_red is positive. It is now necessary to verify that this choice leads to an actual decrease in the merit function (and not just in the model). Define the actual reduction a_red in the merit function by

$$\begin{aligned} \text{a\_red} &= \psi(x_k,\mu^+) - \psi(x_k + d_k,\mu^+) \\ &= f(x_k) - f(x_k + d_k) + \mu^+(\|c(x_k)\|_2 - \|c(x_k + d_k)\|_2). \end{aligned} \tag{1.5.17}$$

If a_red is sufficiently positive, $d_k$ is accepted and ETR uses $\mu_{k+1} = \mu^+$; otherwise, the trial value $\mu^+$ is discarded along with $d_k$.

There is a danger of $v_k$ being so small that the denominator of (1.5.16) approaches zero and $\mu^+$ blows up (although Lemma 1.7.3 proves $\mu^+$ is always bounded). This happens only if the constraints are already nearly satisfied; hence, there is no point in further penalizing the constraints in this case and we are justified in keeping $\mu^+ = \mu_k$. There is also no assurance that (1.5.16) will make $\mu$ large enough to force convergence to a feasible point. The implementation includes a rule to artificially increase $\mu$ when iterates appear to be leaving feasibility in favor of an infeasible (usually unbounded) minimizer of $f(x)$. In this case $\mu$ is increased enough to make p_red negative for the last "bad" step, forcing a new step to be calculated that gives greater weight to satisfying the constraints. These guidelines for choosing the penalty parameter are sufficient for robust algorithm performance, but the following modifications can often help the algorithm reach a solution faster.

**Heuristic for modifying the penalty parameter.** For certain problems with nonlinear constraints, convergence can be very slow because the penalty parameter is not increased enough. Formula (1.5.16) computes $\mu^+$ using a linearization of the constraints, but if these are sufficiently nonlinear that $\|c(x_k)\|_2 - \|c(x_k + d_k)\|_2$ is not as large as $\|c_k\|_2 - \|A_k^T d_k + c_k\|_2$, then the step may increase the merit function (that is, make a_red negative) and be rejected. The algorithm then finds acceptable steps only for sufficiently small trust regions, and convergence to a solution is slowed. In these cases it is advantageous to give more weight to constraint satisfaction by further increasing the penalty parameter. Then a small decrease $\|c(x_k)\|_2 - \|c(x_k + d_k)\|_2$ is enough to make a_red positive and allow large steps to a solution. However, there is a risk that if $\mu_k$ becomes unnecessarily large, the iterates will follow the constraints too closely. This often results in an excessive number of iterations for convergence in the neighborhood of a solution. The heuristic used in ETR accounts for both situations. It accelerates the increase of $\mu^+$ when iterates appear to be simultaneously increasing the objective function and reducing the feasibility error, and it decreases $\mu^+$ safely near a solution. Let $k - s$, with $s \geq 1$, be the index of the most recent iterate at which the step $d_{k-s}$ was accepted by the algorithm.

**Algorithm 1.5.2** *Heuristic for modifying the penalty parameter*

Compute $\mu^+$ using (1.5.16)
IF $\mu^+ > \mu_k$ AND $\mu^+ < 5\mu_k$ AND $\mu_k > \mu_{k-s}$ AND $\|c_k\|_2 > \frac{1}{5}\|c_{k-s}\|_2$
AND the last two steps tried were not both successful
THEN $\mu^+ \leftarrow \min\{5\mu_k, \mu^+ + 25(\mu^+ - \mu_{k-s})\}$
IF $\mu^+ = \mu_k$ AND $\|v_k\|_2 < \zeta\Delta_k/10$ AND $\|c_k\|_\infty < 10^4\epsilon$
THEN $\mu^+ \leftarrow \max\{\mu_0, \bar{\mu}, \|\lambda_k\|_2\}$
Here $\epsilon$ is the stop tolerance of Algorithm 1.3.2
and $\bar{\mu}$ is the second term inside the brackets in (1.5.16).

The first IF statement allows a further increase in $\mu^+$ if not much progress ($\|c_k\|_2 > \frac{1}{5}\|c_{k-s}\|_2$) has been made towards feasibility – a condition that indicates that the constraint error is not being weighed strongly enough in the merit function – and if one of the last two steps failed, implying that the linearized constraint model is not a good fit. Safeguards are included to prevent the penalty parameter from changing too rapidly ($\mu^+ < 5\mu_k$), and to permit increases only when the penalty parameter is already on the rise ($\mu_k > \mu_{k-s}$). The second IF statement permits reductions of $\mu^+$ only when iterates are close to satisfying the equality constraints. The reduced $\mu^+$ is never smaller than the value $\bar{\mu}$ needed to assure that the step is a descent direction for the merit function, or smaller than $\|\lambda_k\|_2$. The latter bound ensures that the directional derivative of $\psi$ at a first-order KKT point is positive in every direction; thus, the bound helps prevent iterates from leaving the neighborhood of a KKT point. In §1.6.3 numerical results are presented comparing this heuristic with the simpler rule (1.5.16).

### 1.5.7 Second-order Correction

The merit function (1.5.13) is nondifferentiable and therefore suffers from the Maratos effect [55], which can result in poor performance on some problems. The effect can be overcome by computing a second-order correction term (see [34, pp. 393–396] for details) and adding it to $d_k$, giving the new trial step

$$d_{\mathrm{SOC}} = d_k - A_k(A_k^T A_k)^{-1} c(x_k + d_k). \qquad (1.5.18)$$

Computation of the second-order correction term is costly, so it used only to assure a fast asymptotic convergence rate. ETR tries $d_{\mathrm{SOC}}$ only if the step $d_k$ has been rejected, $x_k$ is already nearly feasible, and $\|v_k\|_2$ is small compared to $\|Z_k u_k\|_2$. These conditions attempt to identify the occurrence of the Maratos effect near a feasible point, and are much simpler than the rules suggested by Fletcher in [34]. Combining this logic with the ideas of §1.5.6, the final IF statement of Algorithm 1.3.2 is more precisely defined by the following rules.

Compute $\mu^+$ from Algorithm 1.5.2, a_red from (1.5.17), and p_red from (1.5.15)
IF $\frac{\text{a\_red}}{\text{p\_red}} \geq \eta$
    THEN $x_{k+1} = x_k + d_k$ , $\mu_{k+1} = \mu^+$, and $\Delta_{k+1} \geq \Delta_k$
    ELSE IF $\|v_k\|_2 \leq 0.8\zeta\Delta_k$ AND $\|v_k\|_2 \leq 0.1\,\|Z_k u_k\|_2$
        THEN Compute $d_{\mathrm{SOC}}$ using (1.5.18)
            Recalculate a_red at $d_{\mathrm{SOC}}$
            IF $\frac{\text{a\_red}}{\text{p\_red}} \geq \eta$
                THEN $x_{k+1} = x_k + d_{\mathrm{SOC}}$ , $\mu_{k+1} = \mu^+$, and $\Delta_{k+1} \geq \Delta_k$
                ELSE $x_{k+1} = x_k$ , $\mu_{k+1} = \mu_k$ , and $\Delta_{k+1} \leq \gamma\|d_k\|_2$
        ELSE $x_{k+1} = x_k$ , $\mu_{k+1} = \mu_k$ , and $\Delta_{k+1} \leq \gamma\|d_k\|_2$

Notice that p_red is not recomputed at $d_{\mathrm{SOC}}$, while a_red is, and that the acceptance test for a step with second-order correction involves both quantities. This is a subtlety required for proving global convergence (see Theorem 1.7.7). Also note that if $d_{\mathrm{SOC}}$ is accepted, it can violate the trust region constraint; however, the violation should not be large because $v_k$ was within its trust region.

### 1.5.8 Modifying the Trust Region Size

Updates to the trust region size take place at the end of each iteration of the ETR algorithm. So far, modifications have been described generally as either $\Delta_{k+1} \geq \Delta_k$ or $\Delta_{k+1} \leq \gamma\|d_k\|_2$, with $\gamma \in (0,1)$. Other general modification schemes exist (see [20] or [34]), but performance seems satisfactory with this simple framework. When a step $d_k$ (or $d_{\mathrm{SOC}}$) is accepted the trust region size is increased according to the specific rule:

IF $\frac{\text{a\_red}}{\text{p\_red}} \geq 0.9$
  THEN $\Delta_{k+1} = \max\{10\,\|d_k\|_2, \Delta_k\}$
ELSE IF $\frac{\text{a\_red}}{\text{p\_red}} \geq 0.3$
  THEN $\Delta_{k+1} = \max\{2\,\|d_k\|_2, \Delta_k\}$
ELSE $\Delta_{k+1} = \Delta_k$

When a step is rejected, $\Delta_{k+1}$ is reduced to a fraction $\gamma \in [0.1, 0.5]$ of the failed step length. The precise value is computed by backtracking along $d_k$ to a suitable point using a linear or quadratic function, similar to [32]. For linear backtracking, let $r(\|d\|_2)$ be a linear function of step length that approximates the ratio a_red/p_red. We know the value of $r(\|d_k\|_2)$, and using $r(0) = 1$ completely defines the linear function. The best choice for $\gamma$ is the value that makes $r$ equal to the step acceptance threshold $\eta$; that is, $r(\gamma\|d_k\|_2) = \eta$. Solving the linear equation gives

$$\gamma = \frac{1-\eta}{1-(\text{a\_red}/\text{p\_red})},$$

and this value is safeguarded to lie within $[0.1, 0.5]$ before setting $\Delta_{k+1} = \gamma\|d_k\|_2$.

For quadratic backtracking it is easier to approximate p_red, the predicted reduction of the merit function given by (1.5.15), as a smooth quadratic function $p(\|d\|_2)$. In this case $p(\|d_k\|_2)$ is known, $p(0) = 0$, and a third piece of information is provided by estimating $\dot{p}(0)$, the slope of p_red at $d = \emptyset$. The quadratic has a single stationary point that is easily computed: if it corresponds to the minimum of the quadratic, then this point is chosen for $\gamma$; if not, then $\gamma$ is chosen as small as possible. The sign of the second derivative of the quadratic determines the nature of the stationary point, so the rule is:

IF $\frac{\text{a\_red}}{\text{p\_red}} - \dot{p}(0)\|d_k\|_2 - 1 > 0$
  THEN $\gamma = \frac{\dot{p}(0)}{2\,[\dot{p}(0) - (\text{a\_red}/\text{p\_red})]}$, safeguarded so that $\gamma \in [0.1, 0.5]$
  ELSE $\gamma = 0.1$

Now p_red is a nonsmooth function, so $\dot{p}(0)$ must be interpreted as the directional derivative of p_red at $x_k$ in the direction $d_k$. This quantity is bounded by

$$\begin{aligned} D\,\text{p\_red}(x_k; d_k) &\equiv \lim_{\alpha \to 0^+} \frac{1}{\alpha}\Big\{[\tilde{\psi}_k(0, \mu_{k+1}) - \tilde{\psi}_k(\alpha d_k, \mu_{k+1})] - [\tilde{\psi}_k(0, \mu_{k+1}) - \tilde{\psi}_k(0, \mu_{k+1})]\Big\} \\ &= -d_k^T g_k + \lim_{\alpha \to 0^+} \frac{\mu_{k+1}}{\alpha}\Big(\|c_k\|_2 - \|\alpha A_k^T d_k + c_k\|_2\Big) \\ &\leq -d_k^T g_k + \mu_{k+1}\|A_k^T d_k\|_2. \end{aligned}$$

The maximum value is used for $\dot{p}(0)$.

Both the linear and quadratic backtracking schemes work well, and a combination of the two is used for the test results in §1.6. If $\|d_k\|_2 < \Delta_k$, then quadratic backtracking is employed; otherwise, the linear formula is used. The reasoning behind this is that when the step is cut off by the trust region, it suggests that lower order terms in a power series representation are dominant; hence, the linear approximation seems more appropriate.

### 1.5.9 Summary of Algorithm ETR

The previous discussion has provided a detailed description of the implementation of the Byrd-Omojokun algorithm, including several options for solving the subproblems that arise. Below is an expanded description of Algorithm 1.3.2, which closely reflects the form of the code.

**Algorithm ETR**: *Trust region algorithm for large-scale equality constrained optimization*

Constants $\epsilon > 0$, and $\eta, \zeta, \gamma \in (0,1)$ are given
Start with some $x_0$, and choose $\Delta_0 > 0$ and $\mu_0 > 0$
LOOP, starting with $k = 0$
  Compute $f_k$, $c_k$, $g_k$ and $A_k$
  IF The heuristic in §1.5.2 indicates the basis $B_k$ for $A_k$ is ill-conditioned
    THEN Find a new basis and its LU factors
    ELSE Compute the LU factors of the current basis
  (The null-space basis $Z_k$ is now defined by (1.5.6))
  Compute a multiplier estimate
    Option 1: solve for $\lambda_k = (A_k^T A_k)^{-1} A_k^T g_k$ by Cholesky factorization
    Option 2: solve for $\lambda_k = (A_k^T A_k)^{-1} A_k^T g_k$ by CG
    Option 3: use $\lambda_k = B_k^{-T} g_k^B$ as in (1.5.12)
  IF $\|g_k - A_k \lambda_k\|_\infty < \epsilon$ AND $\|c_k\|_\infty < \epsilon$ THEN STOP
  Solve vertical subproblem (1.5.1)-(1.5.2) for $v_k$
    Option 1: use dogleg method, computing $v_{nw}$ in (1.5.4) by Cholesky
    Option 2: use dogleg method, computing $v_{nw}$ in (1.5.4) by CG
    Option 3: use Steihaug's method
  Compute $W_k$
    Option 1: set $W_k = \nabla_x^2 L_k(x_k, \lambda_k)$
    Option 2: update the $\ell$-BFGS approximation $W_k$ as in §1.5.4
  Solve horizontal subproblem (1.5.7)-(1.5.8) for $u_k$. If $W_k$ is $\nabla_x^2 L_k(x_k, \lambda_k)$, then use the unscaled Steihaug method of §1.5.3; else, use the dogleg method.
  Set $d_k = v_k + Z_k u_k$
  Compute $\mu^+$ using the heuristic given in Algorithm 1.5.2
  Compute a_red from (1.5.17) and p_red from (1.5.15)

IF $\frac{\text{a\_red}}{\text{p\_red}} \geq \eta$
THEN $x_{k+1} = x_k + d_k$ , $\mu_{k+1} = \mu^+$, and $\Delta_{k+1} \geq \Delta_k$
ELSE IF $\|v_k\|_2 \leq 0.8\zeta\Delta_k$ AND $\|v_k\|_2 \leq 0.1\,\|Z_k u_k\|_2$
THEN Compute the second-order correction $d_{\text{SOC}}$:
If Option 1 was used in the vertical step computation, solve the system (1.5.18) by Cholesky; else, solve by CG
Recalculate a_red (1.5.17) at $d_{\text{SOC}}$
IF $\frac{\text{a\_red}}{\text{p\_red}} \geq \eta$
THEN $x_{k+1} = x_k + d_{\text{SOC}}$ , $\mu_{k+1} = \mu^+$, and $\Delta_{k+1} \geq \Delta_k$
ELSE $x_{k+1} = x_k$ , $\mu_{k+1} = \mu_k$ , and $\Delta_{k+1} \leq \gamma\|d_k\|_2$
ELSE $x_{k+1} = x_k$ , $\mu_{k+1} = \mu_k$ , and $\Delta_{k+1} \leq \gamma\|d_k\|_2$
(The expansion and contraction of the trust region $\Delta_{k+1}$ is made according to the rules of §1.5.8)
CONTINUE, after incrementing $k$

A key link between the many options in ETR is the method of solving the linear systems involving the matrix $A_k^T A_k$. These systems occur in the dogleg solution of the vertical subproblem $v_k$, in the computation of the Lagrange multipliers $\lambda_k$, and in the second-order correction step $d_{\text{SOC}}$. Two options are provided for solving these linear systems: Cholesky factorization and CG. The software is consistent in its choice of linear solver, using the same option for every system. If Cholesky factorization is used, then the Cholesky factors are saved and used for solving the systems until a new iterate generates a new matrix $A_k$.

A final important detail of the ETR algorithm concerns the stop tests used for subproblems that are solved by CG or Steihaug's method. These are summarized in Table 1.1. The stopping condition in the third column is the number that the norm of a residual must be smaller than (i.e., it corresponds to the right-hand side of the condition $\|r_{j+1}\|_2 < \epsilon\|r_0\|_2$ or its equivalent). In fact, two residuals are required to be less than this number for acceptance of a solution.

| *subproblem* | *initial residual* $r_0$ | *stop threshold (in terms of* $r_0$*)* |
|---|---|---|
| $\lambda_k$ | $A_k^T g_k$ | $\max\{10^{-7}, 10^{-8} * \text{worst\_err}_{k-1}\}$ |
| $v_{nw}$ | $c_k$ | $\max\{10^{-8}, 10^{-5} * \min\{1, \|r_0\|_2\}\}$ |
| $v_k$ by Steihaug | $A_k c_k$ | $\max\{10^{-10}, 10^{-9} * \min\{1, \|r_0\|_2\}\}$ |
| $(Z_k^T Z_k)^{-1}$ in (1.5.9) | $Z_k^T \bar{g}$ | $\max\{10^{-10}, 10^{-8} * \min\{1, \|r_0\|_2\}\}$ |
| $u_k$ by Alg 1.5.1 | $\bar{g}^T Z_k (Z_k^T Z_k)^{-1} Z_k^T \bar{g}$ | $10^{-2} * \min\{1, \|r_0\|_2\}$ |
| $(Z_k^T Z_k)^{-1}$ in Alg 1.5.1 | $r_{j+1}$ | $\max\{10^{-7}, 10^{-2} * \|r_0\|_2\}$ |
| $d_{\text{SOC}}$ | $c_k$ | $\max\{10^{-8}, 10^{-7} * \min\{1, \|r_0\|_2\}\}$ |

Table 1.1: Stopping Thresholds for CG and Steihaug Subproblems
Note that $\text{worst\_err}_k \equiv \max\{\|c_k\|_\infty, \|g_k - A_k\lambda_k\|_\infty\}$

The stopping conditions are safeguarded for extreme values of the starting residual, but are not scale invariant with respect to the variables; that is, they assume that the unknowns are all roughly of order one. This is true for many of the test problems discussed in the next section. A more sophisticated implementation could use scale-invariant stop tests, or provide automatic scaling of problems.

## 1.6 Test Results

The ETR software implementation presented in the last section was tested on a set of difficult nonlinear equality constrained problems. This section first presents numerical results that justify the choice of three options introduced earlier: the use of the unscaled Steihaug method for solving the horizontal subproblem; the heuristic for automatically choosing a new basis; and the heuristic for aggressively updating the penalty parameter. After establishing that these options perform well, the performance of the fully developed algorithm is examined on the set of test problems.

The test problems are drawn from the CUTE collection [3] (most of the problems can also be obtained as FORTRAN subroutines from the author). Table 1.2 presents a brief overview of the problem set. An asterisk by the problem name means the original CUTE problem has been modified for test purposes; for example, nonlinear inequalities may have been dropped or changed into equalities, or the problem parameters may have been altered to obtain a more ill-conditioned or badly scaled problem (Appendix B describes many of the problems and their modifications in greater detail). The CUTE software interface computes second derivatives, and provides a means of checking results by comparing them to solutions obtained with the LANCELOT software [17]. The number of nonzero elements in the sparse matrices $A$ and $W$ is denoted by $nnz(A)$ and $nnz(W)$, respectively.

| *problem name* | $n$ | $m$ | $nnz(A)$ | $nnz(W)$ | *objective* | *constraints* |
|---|---|---|---|---|---|---|
| *HS99LNP | 7 | 2 | 14 | 28 | nonlinear | nonlinear |
| *HS100LNP | 7 | 2 | 10 | 9 | nonlinear | nonlinear |
| *HS104LNP | 8 | 4 | 13 | 16 | nonlinear | nonlinear |
| *HS111LNP | 10 | 3 | 14 | 55 | nonlinear | nonlinear |
| GENHS28 | 300 | 298 | 894 | 599 | quadratic | linear |
| HAGER2 | 1001 | 501 | 1501 | 1501 | quadratic | linear |
| HAGER3 | 8001 | 4001 | 12001 | 20001 | quadratic | linear |
| ORTHREGA | 517 | 256 | 1792 | 2304 | quadratic | quadratic |
| ORTHREGC | 505 | 250 | 1750 | 2250 | quadratic | quadratic |
| ORTHREGD | 203 | 100 | 500 | 906 | quadratic | nonlinear |
| *ORTHRGDM | 4003 | 2000 | 10000 | 18006 | quadratic | nonlinear |
| *ORTHRGDS | 203 | 100 | 500 | 906 | quadratic | nonlinear |
| *OPTCTRL3 | 122 | 83 | 283 | 120 | quadratic | quadratic |
| *OPTCTRL6 | 122 | 83 | 283 | 120 | quadratic | quadratic |

Table 1.2: Description of Test Problem Set

All computations reported in this section were performed in FORTRAN double precision on a Sparc2 Sun workstation containing 32 Mbytes of RAM (the processor was rated at 4.2 Mflops and 24.2 SPECmarks). The various parameters and tolerances occurring in the ETR algorithm were selected assuming double precision calculations (machine epsilon $\approx 10^{-16}$) and a first-order convergence stop tolerance of $\epsilon = 10^{-5}$. The step acceptance threshold was $\eta = 0.1$, and the division of the step between vertical and horizontal portions was controlled by setting $\zeta = 0.8$. The starting values of the penalty parameter and trust region radius were always $\mu_0 = 1$ and $\Delta_0 = 1$, respectively. The stop tests used for subproblems solved by CG and Steihaug's method are given in Table 1.1 of the previous section. It is important to emphasize that these stopping tolerances were chosen to be consistent with the algorithm stop tolerance of $\epsilon = 10^{-5}$. Should $\epsilon$ be different, some subproblem tolerances might also need to be adjusted.

### 1.6.1 Horizontal Subproblem Methods

Let us begin by considering the unscaled Steihaug method introduced in section §1.5.3 to reduce the cost of the horizontal step computation. Following the restrictions discussed in §1.5.3, this method is used in place of Algorithm 1.5.1, the standard Steihaug method, only in circumstances under which it generates the same final horizontal step. Thus, the two methods differ only in computational cost. Since the unscaled Steihaug method ignores the matrix $(Z_k^T Z_k)^{-1}$, it computes iterates more cheaply, but may also take more iterations to reach a final horizontal step (the "deconditioning" effect).

| | standard Steihaug | | unscaled Steihaug | |
|---|---|---|---|---|
| *problem name* | *inner iters* | *CPU time* | *inner iters* | *CPU time* |
| HS99LNP | 4.5 | - | 4.8 | - |
| HS100LNP | 3.6 | - | 3.6 | - |
| HS104LNP | 3.2 | - | 3.1 | - |
| HS111LNP | 4.6 | - | 5.2 | - |
| GENHS28 | 1.3 | - | 1.3 | - |
| HAGER2 | 2.8 | 1.2 sec | 2.8 | 1.2 sec |
| ORTHREGA | 3.3 | 32.0 sec | 3.7 | 30.7 sec |
| ORTHREGC | 5.1 | 33.0 sec | 5.8 | 30.0 sec |
| ORTHREGD | 2.0 | 2.7 sec | 2.4 | 2.5 sec |
| ORTHRGDM | 2.7 | 31.7 sec | 5.2 | 19.6 sec |
| OPTCTRL3 | 9.2 | 3.2 sec | 7.5 | 2.3 sec |
| OPTCTRL6 | 10.8 | 5.0 sec | 7.7 | 3.0 sec |

Table 1.3: Comparison of Standard Steihaug and Unscaled Steihaug Methods

Table 1.3 gives the average number of iterations per horizontal subproblem needed to reach a solution (*inner iters*), and the total CPU time for solving all horizontal subproblems (*CPU time*). Some problems are solved so quickly that the execution time cannot be reliably measured. The tests used second derivatives for $W_k$ and averaged three runs for each problem, one for each of the three vertical subproblem solution options (i.e., dogleg using CG, dogleg using Cholesky factorization, and Steihaug's method). Table 1.3 shows that deconditioning usually causes the unscaled Steihaug method to make more inner iterations in solving each horizontal subproblem, but that it still enjoys an overall savings in execution time. The savings is small because the restrictions in §1.5.3 often prevent the unscaled Steihaug method from being used.

When $W_k$ is a positive definite $\ell$-BFGS approximation to $\nabla_x^2 L_k(x_k, \lambda_k)$, then the horizontal subproblem can be solved by the dogleg method. Every Newton point is found by ignoring $(Z_k^T Z_k)^{-1}$, so the dogleg method has the computational savings of the unscaled Steihaug method on every horizontal subproblem. However, the dogleg step is not always the same as the horizontal step generated by Steihaug's method, and this sometimes degrades the overall performance of the Byrd-Omojokun algorithm. Table 1.4 compares the unscaled Steihaug method to the dogleg method. It lists the average number of inner iterations to find each horizontal step (*iters*), the CPU time for solving all horizontal subproblems (*h CPU*), the number of function and constraint evaluations required to solve the problem (*f,c*), and the total CPU time used (*ttl CPU*). The tests all used $\ell$-BFGS approximations for $W_k$ and averaged three runs for each problem as before.

| | unscaled Steihaug | | | | dogleg | | | |
|---|---|---|---|---|---|---|---|---|
| *problem name* | *iters* | *h CPU* | *f,c* | *ttl CPU* | *iters* | *h CPU* | *f,c* | *ttl CPU* |
| HS100LNP | 4.1 | - | 22.0 | - | 5.0 | - | 22.0 | - |
| HS104LNP | 3.0 | - | 24.0 | - | 3.8 | - | 33.3 | - |
| HS111LNP | 5.6 | - | 30.0 | - | 6.0 | - | 30.0 | - |
| GENHS28 | 2.0 | - | 8.0 | - | 2.6 | - | 8.0 | - |
| HAGER2 | 1.7 | 1.1 sec | 7.0 | 36.4 sec | 3.0 | 0.8 sec | 7.0 | 36.1 sec |
| ORTHREGA | 7.3 | 52.0 sec | 87.7 | 260.4 sec | 18.7 | 40.1 sec | 83.3 | 237.3 sec |
| ORTHREGC | 6.9 | 26.6 sec | 50.7 | 161.5 sec | 14.5 | 19.1 sec | 50.7 | 118.5 sec |
| ORTHREGD | 5.1 | 1.2 sec | 12.3 | 6.2 sec | 6.3 | 0.8 sec | 12.3 | 5.8 sec |
| ORTHRGDM | 5.1 | 16.6 sec | 9.0 | 211.9 sec | 6.3 | 10.2 sec | 9.0 | 209.6 sec |
| OPTCTRL3 | 7.8 | 9.8 sec | 65.3 | 25.1 sec | 10.2 | 4.4 sec | 52.7 | 16.0 sec |
| OPTCTRL6 | 9.9 | 16.6 sec | 124.0 | 45.1 sec | 10.5 | 11.8 sec | 141.0 | 42.6 sec |

Table 1.4: Comparison of Unscaled Steihaug and Dogleg Methods

The test results in Table 1.4 show that the dogleg method is always faster, even though it occasionally increases the number of function evaluations. In accordance with the description of Algorithm ETR given in §1.5.9, all subsequent tests solve the horizontal subproblem by the unscaled Steihaug method when second derivatives are used, or by the dogleg method when an $\ell$-BFGS approximation is used.

### 1.6.2 Change of Basis Heuristic

This section tests the performance of the heuristic described in §1.5.2. The heuristic is a strategy for deciding when to choose a new basis $B_k$ from $A_k$ using Harwell subroutine MA28. It is compared here to the alternative strategies of always using the first basis $B_0$

(that is, never changing the basis) and always letting MA28 choose a basis freely. Only problems with nonlinear constraints are tested, since $A_k$ is constant for linear constraints.

In the first test the condition number of the basis matrix $B_k$ was calculated at every iteration from the singular value decomposition. The largest condition number encountered in solving a problem with each strategy is shown in Table 1.5. Two runs were made on each problem, one using exact Hessians and another using $\ell$-BFGS approximations, with the linear systems involving $A_k^T A_k$ always solved by CG.

| *problem name* | *never change basis* | *always new basis* | *ETR heuristic* |
|---|---|---|---|
| HS99LNP | $1.7 \times 10^2$ | $2.2 \times 10^2$ | $2.2 \times 10^2$ |
| HS100LNP | $9.1 \times 10^0$ | $8.7 \times 10^0$ | $9.1 \times 10^0$ |
| HS104LNP | $8.8 \times 10^6$ | $1.3 \times 10^1$ | $6.7 \times 10^4$ |
| HS111LNP | $1.3 \times 10^1$ | $1.4 \times 10^1$ | $1.3 \times 10^1$ |
| ORTHREGA | $5.4 \times 10^8$ | $2.5 \times 10^6$ | $1.3 \times 10^7$ |
| ORTHREGC | $5.8 \times 10^5$ | $1.7 \times 10^3$ | $1.3 \times 10^4$ |
| ORTHREGD | $4.6 \times 10^4$ | $1.1 \times 10^4$ | $4.6 \times 10^4$ |
| OPTCTRL3 | $3.8 \times 10^2$ | $3.6 \times 10^2$ | $3.6 \times 10^2$ |
| OPTCTRL6 | $5.2 \times 10^2$ | $3.5 \times 10^2$ | $5.0 \times 10^2$ |

Table 1.5: Highest Condition Number of Basis Matrix Seen

As expected, the condition number is best controlled by choosing a new basis at every iteration. A better conditioned basis also makes $Z_k^T Z_k$ better conditioned, which should reduce the number of CG iterations needed to approximately invert $Z_k^T Z_k$ (recall from §1.5.3 that this inversion is needed to calculate $Z_k u_{cp}$ and each scaled residual vector $t_j$ in Algorithm 1.5.1). Table 1.6 confirms this supposition, but also shows that choosing a new basis at every iteration is very expensive. Table 1.6 lists the number of times a new basis is computed (*nb*), the percent of total CPU time spent by Harwell subroutine MA28 choosing and factoring a basis (*%f*), the number of times per iteration that $Z_k$ or $Z_k^T$ is multiplied by a vector (*Z/it*), and the percent of total CPU time used to solve the horizontal subproblem (*%h*). The same two runs were made on each problem as for Table 1.5, and the two sets of data were averaged.

The test results show a direct tradeoff between the time spent choosing a new basis and the time spent solving the horizontal subproblem. There is no apparent difference in robustness of the algorithm among the different strategies. Tables 1.5 and 1.6 suggest that the heuristic of §1.5.2 adequately controls the condition number of the basis and results in an acceptable computing time; therefore, it is used in all subsequent tests.

### 1.6.3 Penalty Parameter Heuristic

The remaining heuristic in ETR modifies the penalty parameter $\mu$ as described by Algorithm 1.5.2 in §1.5.6. The alternative to this heuristic is the standard updating rule (1.5.16). Table 1.7 compares the number of function and constraint evaluations $(f, c)$ and the number

| *problem name* | never change basis *%f* | *Z/it* | *%h* | always new basis *nb* | *%f* | *Z/it* | *%h* | ETR heuristic *nb* | *%f* | *Z/it* | *%h* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| HS99LNP | - | 29.0 | - | 5.0 | - | 28.5 | - | 2.0 | - | 28.5 | - |
| HS100LNP | - | 15.5 | - | 13.5 | - | 15.5 | - | 1.0 | - | 15.5 | - |
| HS104LNP | - | 18.3 | - | 15.5 | - | 17.9 | - | 2.5 | - | 17.9 | - |
| HS111LNP | - | 14.9 | - | 19.0 | - | 16.5 | - | 1.0 | - | 14.9 | - |
| ORTHREGA | 5% | 455.5 | 77% | 35.0 | 64% | 64.5 | 14% | 9.5 | 41% | 79.1 | 25% |
| ORTHREGC | 5% | 576.8 | 80% | 26.0 | 42% | 70.2 | 28% | 3.5 | 18% | 85.1 | 42% |
| ORTHREGD | 9% | 38.0 | 26% | 11.0 | 34% | 36.2 | 19% | 2.5 | 14% | 37.2 | 24% |
| ORTHRGDM | 73% | 25.7 | 10% | 8.0 | 91% | 25.3 | 3% | 1.5 | 76% | 26.9 | 9% |
| OPTCTRL3 | 10% | 42.7 | 32% | 24.0 | 41% | 42.6 | 21% | 5.5 | 18% | 42.9 | 29% |
| OPTCTRL6 | 10% | 30.2 | 31% | 54.0 | 45% | 36.1 | 20% | 3.5 | 11% | 27.5 | 29% |

Table 1.6: Performance of Strategies for Choosing a Basis

of gradient evaluations $(g, A)$ required for convergence using (1.5.16) and the strategy of Algorithm 1.5.2. The results were obtained by averaging all successful runs for each problem using the three options for solving the vertical subproblem on both exact Hessians and $\ell$-BFGS approximations (a total of six possible runs per problem). Only problems for which the penalty parameter changed on at least one iteration from its initial value of $\mu_0 = 1$ are included in the table.

| *problem name* | rule (1.5.16) *f,c* | *g,A* | Algorithm 1.5.2 *f,c* | *g,A* |
|---|---|---|---|---|
| HS99LNP | 125.0 | 74.7 | 125.0 | 74.0 |
| HS100LNP | 16.5 | 13.5 | 16.5 | 13.5 |
| HS104LNP | 25.8 | 21.2 | 25.0 | 21.3 |
| HS111LNP | 22.0 | 17.8 | 20.5 | 17.5 |
| ORTHREGA | 202.8 | 118.3 | 66.3 | 32.7 |
| ORTHREGC | 40.5 | 23.0 | 40.5 | 23.0 |
| OPTCTRL3 | 25.8 | 24.2 | 26.0 | 24.0 |
| OPTCTRL6 | 70.0 | 61.0 | 61.2 | 58.2 |

Table 1.7: Comparison of Two Strategies for Updating the Penalty Parameter

The results justify the inclusion of the heuristic in ETR, particularly for two of the test problems. The dramatic improvement seen in problem ORTHREGA is due to the nature of this test problem. The constraints are nonlinear in such a manner that $\|c(x_k + d_k)\|_2$ is smallest for a step $d_k$ with no horizontal component; i.e., the horizontal step undoes some of the progress made by the vertical step. In addition, the start point for this problem requires steps be taken that increase the objective $f$ while decreasing the constraint violation. Such a step is accepted only if the constraint reduction outweighs the increase in $f$. The "weight" is $\mu^+$, which when calculated by (1.5.16) is implicitly based on a linearization of the constraints; consequently, the nonlinear effect of the horizontal step on the constraints is ignored and $\mu^+$ is often too small for the step $d_k$ to be accepted. Rejected steps decrease the trust region size, which eventually becomes small enough that the linear terms of $c$ dominate the higher-order terms and allow a step to be accepted even with the small value of $\mu^+$. These tiny steps succeed in solving the problem, but only after many iterations. The heuristic overcomes this by making $\mu^+$ larger in these circumstances. The extra increase given to $\mu^+$ by Algorithm 1.5.2 is admittedly ad-hoc, and could more properly be computed from constraint information at $c(x_k + d_k)$, but this is an expensive operation (similar to the second-order correction discussed in §1.5.7).

Test problem OPTCTRL6 illustrates the value of the second part of Algorithm 1.5.2, which decreases the penalty parameter near a solution. In this problem the ill conditioning of the objective caused equation (1.5.16) to increase $\mu_k$ to $O(10^7)$ during the middle iterations of a run using $\ell$-BFGS approximations for $W_k$. The heuristic safely decreased this value to $O(10^4)$ after iterates achieved constraint feasibility, which in this example occurred to an accuracy of $10^{-10}$ well before the neighborhood of a solution was reached. Now if $\mu_k$ is very large, then nonlinear effects on the constraints due to the horizontal step are greatly magnified. A step $d_k$ may be rejected even though it decreases $f$ because $\|c(x_k + d_k)\|_2$ may be slightly greater than $\|c_k\|_2$. Second-order corrections help, but make the feasibility error smaller for the next iteration and thus even more sensitive to nonlinear effects. Ultimately, $\|c_k\|_2$ may approach machine epsilon ($10^{-16}$ in these experiments), where it cannot be reduced further; at this point the algorithm fails to find a successful step. Using the heuristic, $\mu$ is decreased when appropriate and an improvement in robustness obtained for both OPTCTRL3 and OPTCTRL6.

### 1.6.4 Cheaper Lagrange Multiplier Estimates

The default method for estimating Lagrange multipliers in ETR is to solve the least squares problem $A_k \lambda_k = g_k$. A computationally cheaper but less accurate method was described in §1.5.5. Table 1.8 shows the results of comparing the two methods for the test problems. The data for each problem considers six runs using the three different options for solving the vertical subproblem for both exact Hessians and $\ell$-BFGS approximations. Table 1.8 reports the average percent of CPU time spent computing lambda (*%lambda*), the average total CPU time required to solve each problem (*ttl CPU*), and the number of runs that failed to converge (*failures*).

In general the cheaper method greatly reduced the cost of computing Lagrange multipliers, but gave such inaccurate estimates that the problems took more iterations to solve; thus, the overall computational cost usually became worse. The cheaper estimate also made the algorithm less robust, failing to converge in five instances compared to only one failure using the least squares estimate. However, the cheaper method made a significant improvement in solving HAGER2 and HAGER3. These two problems have ill-conditioned linear

| problem name | least squares method (1.5.11) | | | cheaper method (1.5.12) | | |
|---|---|---|---|---|---|---|
| | *%lambda* | *ttl CPU* | *failures* | *%lambda* | *ttl CPU* | *failures* |
| GENHS28 | 37% | 2.1 sec | 0 | 3% | 1.4 sec | 0 |
| HAGER2 | 51% | 31.4 sec | 0 | 1% | 12.7 sec | 0 |
| HAGER3 | 50% | 2924.4 sec | 0 | $\approx$ 0% | 1310.9 sec | 0 |
| ORTHREGA | 6% | 224.3 sec | 0 | $\approx$ 0% | 481.1 sec | 1 |
| ORTHREGC | 9% | 107.4 sec | 0 | $\approx$ 0% | 328.5 sec | 0 |
| ORTHREGD | 14% | 7.0 sec | 0 | $\approx$ 0% | 23.7 sec | 0 |
| ORTHRGDM | 6% | 176.0 sec | 0 | $\approx$ 0% | 430.0 sec | 1 |
| OPTCTRL3 | 26% | 10.4 sec | 0 | 1% | 10.7 sec | 1 |
| OPTCTRL6 | 39% | 20.4 sec | 1 | 1% | 10.7 sec | 2 |

Table 1.8: Comparison of Methods for Estimating Lagrange Multipliers

constraints that make the least squares estimate particularly expensive to compute. Considering all the evidence, the cheaper multiplier estimate of (1.5.12) can only be recommended in specific cases, and it is not used in any further experiments reported in this paper.

The equation for the Lagrange multipliers corresponding to the exact Newton step that solves the model equality constrained subproblem was also tried as an estimate for $\lambda_{k+1}$. This formula is given by (1.2.18), and is not much more expensive to compute than the least squares estimate. It performs badly on most test problems, especially when the exact Hessian of the Lagrangian is not used. This is not an unreasonable result, because the exact Newton step is not usually the step taken by the algorithm; thus, (1.2.18) has no validity except when second derivatives are used and iterates are near a local solution.

### 1.6.5 Rank Deficiency of $A_k$

A weakness of the ETR implementation is its strong need for the constraint Jacobian $A_k$ to be full rank at every iteration. If the matrix is truly rank deficient, then the null space matrix $Z_k$ cannot be formed by the method described in §1.5.2 and execution must terminate. If $A_k$ is full rank but nearly deficient, the ill-conditioning can seriously impair algorithm performance. To study this, the start point of problem ORTHREGD was modified to give a new problem, ORTHRGDS, whose constraint Jacobian is singular at certain local solution points. As explained in Appendix B, the start point of this problem can be chosen to cause convergence to various solution points where $A_*$ has different numbers of columns whose components are all zero.

Table 1.9 summarizes the behavior of the algorithm as it tries to solve these problems. The rows of the table are arranged in order of increasing rank deficiency, so that *rank def* equals the number of zero column vectors in $A_*$. The problem was attempted with two different stop tolerances, $10^{-5}$ and $10^{-6}$, for which the number of function and constraint evaluations required for convergence to the respective tolerance is reported. Second derivatives were used with each of the three options for solving the vertical subproblem: the dogleg method via Cholesky factorization (*Cholesky*), Steihaug's method (*Steihaug*), and the dogleg method using conjugate gradient (*CG*).

| | $\epsilon = 10^{-5}$ | | | $\epsilon = 10^{-6}$ | | |
|---|---|---|---|---|---|---|
| *rank def* | *Cholesky* | *Steihaug* | *CG* | *Cholesky* | *Steihaug* | *CG* |
| 0 | 13 | 15 | 13 | 13 | 15 | 13 |
| 2 | 74 | 27 | 74 | 80 | 27 | 80 |
| 4 | 127 | 30 | 127 | 242 | failed | 242 |
| 6 | 91 | failed | 91 | 135 | failed | 135 |
| 8 | failed | failed | 173 | failed | failed | 111 |
| 10 | 73 | 76 | 73 | failed | failed | failed |
| 12 | 82 | failed | 82 | failed | failed | failed |
| 14 | 82 | failed | 82 | failed | failed | failed |
| 16 | failed | failed | failed | failed | failed | failed |

Table 1.9: Number of $f, c$ Evaluations to Solve Problem ORTHRGDS

The algorithm consistently reached the neighborhood of the solution, but had difficulty in satisfying the stop conditions to the required tolerance. Iterates seemed to wander aimlessly around an area close to the solution, and eventually enough steps were rejected that the trust region became too small to continue. Although the three methods for solving the vertical subproblem essentially square the condition number of $A_k$, the feasibility error $\|c_k\|_\infty$ decreased steadily at a slow linear rate in all cases. It was the KKT error $\|g_k - A_k\lambda_k\|_\infty$ that behaved erratically, suggesting that ill-conditioning of the basis matrix $B_k$ may be the biggest source of difficulty. In any event, the robustness of ETR clearly needs improving for problems with linearly dependent constraint gradients.

### 1.6.6 Comprehensive Tests

Having confirmed the validity of all heuristics and investigated the behavior of the algorithm in extreme situations, we are ready to describe its performance on the complete set of test problems. The values of all parameters used by the algorithm are given just after Table 1.2 and at the end of §1.5.9. They do not vary except for the initial trust region size for problems HS104LNP and HS111LNP. Tables 1.10 and 1.11 show the number of times that the function and constraints were evaluated (*f,c*), the number of evaluations of their gradients (*g,A*), and total CPU time (*ttl CPU*). Table 1.10 presents results using exact second derivatives to define the Hessian of the Lagrangian $W_k$, and Table 1.11 using $\ell$-BFGS approximations. Data is presented for all three methods for solving the vertical subproblem: dogleg using the sparse Cholesky factorization of $A_k^T A_k$, Steihaug's method, and dogleg with $A_k^T A_k$ inverted by conjugate gradient.

To solve problems HS104LNP and HS111LNP a small modification was made to the parameters controlling the algorithm: the maximum growth of the trust region was limited to a factor much smaller than the default value of 10 (growth factors are described in

| | $v_k$ by dogleg-Cholesky | | | $v_k$ by Steihaug | | | $v_k$ by dogleg-CG | | |
|---|---|---|---|---|---|---|---|---|---|
| *problem name* | *f,c* | *g,A* | *ttl CPU* | *f,c* | *g,A* | *ttl CPU* | *f,c* | *g,A* | *ttl CPU* |
| HS99LNP | 5 | 5 | - | 365 | 214 | 4 sec | 5 | 5 | - |
| HS100LNP | 11 | 8 | - | 11 | 8 | - | 11 | 8 | - |
| HS104LNP | 12 | 11 | - | 12 | 11 | - | 12 | 11 | - |
| HS111LNP | 11 | 11 | - | 11 | 11 | - | 11 | 11 | - |
| GENHS28 | 4 | 4 | 1 sec | 4 | 4 | 2 sec | 4 | 4 | 2 sec |
| HAGER2 | 4 | 4 | 3 sec | 4 | 4 | 39 sec | 4 | 4 | 37 sec |
| HAGER3 | 5 | 5 | 59 sec | 5 | 5 | 4120 sec | 5 | 5 | 3238 sec |
| ORTHREGA | 43 | 20 | 279 sec | 61 | 26 | 221 sec | 44 | 19 | 133 sec |
| ORTHREGC | 32 | 16 | 160 sec | 27 | 14 | 64 sec | 32 | 16 | 64 sec |
| ORTHREGD | 13 | 10 | 10 sec | 15 | 11 | 7 sec | 13 | 10 | 8 sec |
| ORTHRGDM | factors too dense | | | 7 | 7 | 148 sec | 7 | 7 | 144 sec |
| OPTCTRL3 | 7 | 7 | 4 sec | 7 | 7 | 5 sec | 7 | 7 | 5 sec |
| OPTCTRL6 | 8 | 8 | 5 sec | 8 | 8 | 6 sec | 8 | 8 | 6 sec |

Table 1.10: Performance Results Using Exact Hessians

| | $v_k$ by dogleg-Cholesky | | | $v_k$ by Steihaug | | | $v_k$ by dogleg-CG | | |
|---|---|---|---|---|---|---|---|---|---|
| *problem name* | *f,c* | *g,A* | *ttl CPU* | *f,c* | *g,A* | *ttl CPU* | *f,c* | *g,A* | *ttl CPU* |
| HS99LNP | did not converge | | | did not converge | | | did not converge | | |
| HS100LNP | 22 | 19 | - | 22 | 19 | - | 22 | 19 | - |
| HS104LNP | 31 | 28 | - | 31 | 28 | - | 52 | 39 | - |
| HS111LNP | 30 | 24 | - | 30 | 24 | - | 30 | 24 | - |
| GENHS28 | 8 | 8 | 2 sec | 8 | 8 | 3 sec | 8 | 8 | 3 sec |
| HAGER2 | 7 | 7 | 3 sec | 7 | 7 | 55 sec | 7 | 7 | 51 sec |
| HAGER3 | 8 | 8 | 56 sec | 8 | 8 | 5654 sec | 8 | 8 | 4420 sec |
| ORTHREGA | 83 | 43 | 434 sec | 86 | 44 | 141 sec | 81 | 44 | 137 sec |
| ORTHREGC | 36 | 26 | 214 sec | 58 | 32 | 73 sec | 58 | 34 | 69 sec |
| ORTHREGD | 13 | 12 | 9 sec | 11 | 11 | 3 sec | 13 | 12 | 5 sec |
| ORTHRGDM | factors too dense | | | 9 | 9 | 209 sec | 9 | 9 | 203 sec |
| OPTCTRL3 | 49 | 41 | 10 sec | 63 | 59 | 22 sec | 46 | 40 | 16 sec |
| OPTCTRL6 | did not converge | | | 74 | 69 | 27 sec | 208 | 198 | 59 sec |

Table 1.11: Performance Results Using $\ell$-BFGS Approximations

§1.5.8). These two problems originally contained bound constraints that were inactive at the solution, and a smaller trust region was necessary to avoid generating iterates where $f$ was not defined. Further details may be found in Appendix B.

The results in Tables 1.10 and 1.11 show the value of having three different options for solving the vertical subproblem. In problems HAGER2 and HAGER3 the constraint basis $B_k$ is ill-conditioned and large; consequently, the conjugate gradient iterations used in the Steihaug and dogleg-CG options were very inefficient. On the other hand, the Cholesky factorization of $A_k^T A_k$ was very time-consuming in the ORTHREG family of problems, in which a great amount amount of fill-in occurred. In fact, for problem ORTHRGDM the memory requirements of the Cholesky factorization exceeded the capacity of the workstation. As these problems illustrate, the choice of iterative versus direct methods for solving the vertical subproblem depends on the nature of the problem's constraints, and can be critical to performance.

Tha algorithm failed to converge to the desired tolerance of $10^{-5}$ in four instances, all using limited memory approximations for the Hessian of the Lagrangian. In each case the algorithm halted because the trust region size became too small ($< 10^{-15}$) for computations to continue. Problem OPTCTRL6 is deliberately constructed (see Appendix B) to have a highly ill-conditioned objective, and the algorithm actually came quite close to solving it before failing. Problem HS99 is highly nonlinear and known to be difficult to solve without second derivative information.

The convergence rate of the algorithm when second derivatives were used appeared to be quadratic for all the problems except HS111LNP. The trust region always became inactive during the final iterations, and the two errors $\|c_k\|_\infty$ and $\|g_k - A_k\lambda_k\|_\infty$ contracted independently of one another. The ORTHREG family of problems have the kind of constraint curvature that gives rise to the Maratos effect, but second-order correction steps were successful in maintaining a quadratic convergence rate. When $\ell$-BFGS approximations were used for $W_k$ the rate of convergence was linear in $\|g_k - A_k\lambda_k\|_\infty$, but still usually quadratic in $\|c_k\|_\infty$. The trust region constraint did not always become inactive in these cases.

Finally, the performance of ETR is compared to that of LANCELOT [17]. This comparison is not meant to suggest that ETR is as refined or reliable as the LANCELOT software – it is not. But the comparison reveals that the performance of ETR on these test problems is quite good, and differs from that of LANCELOT in an important way.

The basic subproblem that is iteratively solved by LANCELOT is described in §1.3.2. Computationally, this trust region subproblem is solved approximately by a very efficient two-step procedure. First, the gradient is projected onto the boundary of the trust region hypercube and used to estimate which variables of the subproblem solution are at their bounds. These variables are fixed at the trust region boundary, and then the remaining free variables are minimized. The solution of this smaller dimension subproblem is found by either direct methods or conjugate gradient. The LANCELOT software offers the user a variety of preconditioners for enhancing the CG solution method.

Table 1.12 compares the performance of LANCELOT to that of ETR when second derivatives are used and the "best" subproblem solution option for each problem is chosen by the user. For LANCELOT this means each problem was solved with one of eleven different options for minimizing the unconstrained subproblem, including modified Cholesky factorization (a direct method), unpreconditioned CG, and preconditioned forms of CG. The best result, defined as the one taking the smallest number of function and constraint evaluations ($f$,$c$) to converge, is reported in the table (the same option usually had the fastest execution time as well). For ETR, the best results from Table 1.10 among the three options for solving the vertical subproblem are recorded. Both algorithms used the same stop tolerance of $\epsilon = 10^{-5}$. The CPU time (*ttl CPU*) for LANCELOT in Table 1.12 is the time reported in the SUMMARY.d output file.

| | best LANCELOT | | | best ETR | | |
|---|---|---|---|---|---|---|
| *problem* | $f$,$c$ | $g$,$A$ | *ttl CPU* | $f$,$c$ | $g$,$A$ | *ttl CPU* |
| HS99LNP | 75 | 62 | 2 sec | 5 | 5 | - |
| HS100LNP | 20 | 20 | - | 11 | 8 | - |
| HS104LNP | 21 | 22 | - | 12 | 11 | - |
| HS111LNP | 40 | 40 | 1 sec | 11 | 11 | - |
| GENHS28 | 6 | 7 | 3 sec | 4 | 4 | 1 sec |
| HAGER2 | 4 | 5 | 7 sec | 4 | 4 | 3 sec |
| HAGER3 | 11 | 12 | 71 sec | 5 | 5 | 59 sec |
| ORTHREGA | 92 | 84 | 52 sec | 44 | 19 | 133 sec |
| ORTHREGC | 30 | 27 | 23 sec | 27 | 14 | 64 sec |
| ORTHREGD | 44 | 40 | 6 sec | 13 | 10 | 8 sec |
| ORTHRGDM | 29 | 26 | 92 sec | 7 | 7 | 144 sec |
| OPTCTRL3 | 37 | 38 | 4 sec | 7 | 7 | 4 sec |
| OPTCTRL6 | 46 | 47 | 5 sec | 8 | 8 | 5 sec |

Table 1.12: Comparison of Two Algorithms Using Exact Hessians

The most interesting feature of the results in Table 1.12 is that ETR needs problem information at significantly fewer sample points than LANCELOT in order to reach a solution. This indicates that it behaves in a fundamentally different manner, making much greater use of problem information at each point before generating a guess of the next point. The careful use of problem information also requires more total CPU time than LANCELOT, although further improvements to the algorithm might decrease this figure.

## 1.7 Convergence Analysis

This section states and proves basic convergence properties of the Byrd-Omojokun algorithm. These proofs specify certain necessary conditions that must be met by an implementation of the Byrd-Omojokun algorithm, some of which are not rigorously fulfilled by the implementation presented in §1.5. Thus, there is a gap between theory and the ETR algorithm, a gap that results from the use of efficient approximation techniques in ETR to solve large-scale problems. The loss of theoretical robustness in exchange for efficient large-scale problem performance seems to be a reasonable trade. The purpose of this section is to present the convergence theory, which is fairly standard, and to precisely understand why it may not apply to ETR.

### 1.7.1 Necessary Conditions

The convergence proofs make certain assumptions about the equality constrained problem (EP) being solved. These are listed here for convenience. Let $\Omega$ be an open subset of $\mathbf{R}^n$ that contains the start point of the problem $x_0$.

(A1) For all $x \in \Omega$, $f(x)$ and each $c^i(x)$ are at least twice continuously differentiable.

(A2) For all $x \in \Omega$, the gradients and Hessians of $f(x)$ and each $c^i(x)$ are bounded; specifically,

$$\begin{aligned} \|g(x)\|_2 &\leq \beta_g \\ \|\nabla^2 f(x)\|_2 &\leq \beta_f \\ \|A(x)\|_2 = \|A(x)^T\|_2 &\leq \beta_A \\ \|\nabla^2 c^i(x)\|_2 &\leq \beta_c, \quad \text{for } i = 1, \ldots, m. \end{aligned}$$

(A3) For all $x \in \Omega$, the constraint gradients are linearly independent; that is, $A(x)$ is full rank and $A(x)^T A(x)$ is positive definite. Furthermore, the smallest eigenvalue of $A(x)^T A(x)$ is bounded away from zero. The implicit representation of $Z(x)$ in terms of $A(x)$ defined by equation (1.5.6) is also bounded. In summary,

$$\begin{aligned} \|[A(x)^T A(x)]^{-1}\|_2 &\leq \beta_{A^T A} \\ \|Z(x)\|_2 = \|Z(x)^T\|_2 &\leq \beta_Z \\ \|[Z(x)^T Z(x)]^{-1}\|_2 &\leq \beta_{Z^T Z}. \end{aligned}$$

(A4) Let $\mathcal{F} = \{x : x \in \Omega \ \mathsf{AND} \ c^i(x) = 0, \ i = 1, \ldots, m\}$ be the set of feasible points in $\Omega$. The level sets of $f(x)$ in $\mathcal{F}$ are bounded; i.e., the set $\{x : x \in \mathcal{F} \ \mathsf{AND} \ f(x) \leq f(y)\}$ is bounded for every $y \in \mathcal{F}$.

Assumption (A3) is somewhat restrictive on (EP), but is not uncommon. It is essential that $A(x)$ be full rank in ETR because of the way $v_{nw}$ and $\lambda$ are computed in equations (1.5.4) and (1.5.11), and because of the implicit representation of $Z(x)$ defined in (1.5.6).

Assumption (A4) guarantees that problem (EP) has local solutions in the sense of Definition 1.2.1. It is needed to prove Lemma 1.7.4. Many authors use a less restrictive form of

(A4) for their convergence analysis, requiring only that $f$ be bounded from below on the feasible set $\mathcal{F}$. Unlike (A4), this weaker assumption allows feasible minimizers to be unbounded even though $f$ is bounded from below (for example, min $e^{-x_1-x_2}$ subject to $x_2=0$). Assumption (A4) serves to restrict the pool of candidate problems to those which have solutions computable by an iterative algorithm.

The convergence analysis is developed for the general Byrd-Omojokun Algorithm 1.3.2 given in §1.3.3. The implementation of this algorithm (of which ETR in §1.5.9 is an example) is deliberately left unspecified in this section except for certain necessary conditions. These conditions are collected in the list below.

(C1) The vertical subproblem (1.5.1)-(1.5.2) is solved by a method that reduces the objective (1.5.1) at least as much as the Cauchy point $v_{cp}$ does. Also, the method never allows the vertical step to be larger than $v_{nw}$, which is defined in (1.5.4).

(C2) The horizontal subproblem (1.5.7)-(1.5.8) is solved by a method that reduces the objective (1.5.7) at least as much as the Cauchy point $Z_k u_{cp}$ does.

(C3) The trust region remains finite; i.e., $\Delta_k \leq \Delta_{max}$ for all $k$.

(C4) When a step $d_k$ is accepted, the penalty parameter $\mu_k$ is updated by the rule

$$\begin{array}{ll} \mathsf{IF} \ \|v_k\|_2 > 0 & \\ \quad \mathsf{THEN} & \mu_{k+1} = \max\left\{\mu_k,\ 0.1 + \dfrac{d_k^T g_k + \frac{1}{2} d_k^T W_k d_k}{\|c_k\|_2 - \|A_k^T v_k + c_k\|_2}\right\} \\ \quad \mathsf{ELSE} & \mu_{k+1} = \mu_k\ . \end{array}$$

Furthermore, $\mu_0 > 0$.

(C5) For all $x \in \Omega$, Lagrange multipliers $\lambda(x)$ are bounded.

(C6) The matrix $W(x,\lambda)$ used in the Byrd-Omojokun algorithm is symmetric and bounded for every $x \in \Omega$; specifically,

$$\|W(x,\lambda)\|_2 \leq \beta_W.$$

(C7) There exists a number $K_1 \geq 0$ such that for all iterations $k \geq K_1$ the penalty parameter is large enough that the minimum of the merit function $\psi(x,\mu_k) = f(x) + \mu_k \|c(x)\|_2$ occurs only at feasible points in $\mathcal{F}$.

The restriction in condition (C1) on the size of the vertical step is needed to prove Lemma 1.7.3. It is satisfied by all the methods used in ETR to solve the vertical subproblem.

Condition (C3) is a reasonable safeguard for both theory and practice. The test results for ETR used $\Delta_{max} = 1000$, although no trust region ever became this large.

Condition (C4) comes from equation (1.5.16) and is necessary to make $d_k$ a direction of descent on the model merit function $\tilde{\psi}$; however, the ETR heuristic for further modifying $\mu$ (see §1.5.6) can violate (C4). The implications of using this heuristic will be discussed further at the end of §1.7.2.

Note that condition (C5) is unnecessary if assumptions (A2) and (A3) hold and the Lagrange multipliers are calculated by the least squares formula $\lambda_k = (A_k^T A_k)^{-1} A_k^T g_k$.

Condition (C6) is true if $W(x, \lambda)$ is taken to be the Hessian of the Lagrangian, $\lambda(x)$ satisfies (C5), and assumptions (A2) and (A3) are true. Recalling from (1.2.5) that the Lagrangian is defined as $\mathcal{L} = f - \lambda^T c$, we see that

$$\begin{aligned}
\|\nabla_x^2 \mathcal{L}\|_2 &= \|\nabla^2 f(x) - \sum_{i=1}^{m} \lambda^i \, \nabla^2 c^i(x)\|_2 \\
&\leq \|\nabla^2 f(x)\|_2 + \|\sum_{i=1}^{m} \lambda^i \, \nabla^2 c^i(x)\|_2 \\
&\leq \beta_f + \beta_c \sum_{i=1}^{m} |\lambda^i| \\
&\leq \beta_f + \sqrt{m}\, \beta_c \beta_{A^T A} \beta_A \beta_g .
\end{aligned}$$

Condition (C7) is needed to prove that $\psi$ is bounded from below in Lemma 1.7.4. A more conventional alternative to (C7) is to replace it and (A4) with the outright assumption that $\psi$ is bounded from below. The arrangement here is preferred because it better explains what is necessary for the merit function to be bounded from below. The conventional alternative is not a stronger or weaker assumption, just less edifying. In any event, (C7) appears at first glance to imply a restriction on the problem (EP). But the nature of this restriction, if any, is not obvious, in part because the Byrd-Omojokun algorithm does not operate by minimizing the merit function (as do some of the algorithms mentioned in §1.3.2). It computes steps by constructing orthogonal vertical and horizontal components that satisfy (C1) and (C2), then *judges* the length of these steps with the merit function. In conjunction with assumption (A3), it appears possible to prove that the Byrd-Omojokun algorithm generates a sequence of iterates that have a feasible cluster point without resorting to condition (C7) at all. To the author's knowledge, this conjecture has not been considered before. A proof of its verity is beyond the scope of this dissertation.

### 1.7.2 First-Order Global Convergence

The goal in this section is to prove that the Byrd-Omojokun algorithm converges from any starting point to a point satisfying the first-order KKT conditions

$$c(x_*) = \emptyset \quad \text{and} \quad \nabla f(x_*) = A(x_*)\, \lambda_* .$$

This can be accomplished with an implementation of the algorithm that solves the vertical and horizontal subproblems with no greater accuracy than that of their Cauchy points; i.e., using only first-order problem information (such a simple implementation will probably converge slowly, but it will converge). The logical argument for this type of proof was first developed by Powell [66] for unconstrained minimization using trust regions. The same basic argument is used for most trust region global convergence proofs (an excellent example is [71], again for unconstrained problems), although the introduction of equality constraints makes things more complicated. The proof here is modelled on [11].

The process of proving first-order global convergence is rather lengthy, and is roughly outlined as follows. First, a number of useful bounds are proved, including an upper bound on the sequence of penalty parameters $\{\mu_0, \dots, \mu_k, \dots\}$. Then it is proved that the algorithm always finds a step to move away from a point not satisfying the first-order KKT conditions, and that these steps make a finite reduction in the merit function. ¿From here global convergence follows by reasoning that the minimum of the merit function is bounded and that the iterates of the Byrd-Omojokun algorithm make finite reductions in $\psi$; hence, the iterates must converge to a point satisfying the first-order KKT conditions for (EP).

To begin with, two essential lemmas are established. They bound the progress made by steps that satisfy (C1) and (C2) in solving the vertical and horizontal subproblems. These lemmas are extensions of Powell's Theorem [66] for unconstrained methods, which is given in Appendix A.4.

**Lemma 1.7.1** *If problem (EP) satisfies (A1)-(A3), and if the vertical step $v$ of the Byrd-Omojokun algorithm satisfies condition (C1), then for every $x_k \in \Omega$,*

$$\|c_k\|_2 - \|A_k^T v + c_k\|_2 \geq \frac{1}{4\beta_A \beta_{A^TA}} \min\left\{ \frac{\|c_k\|_2}{\beta_A^3 \beta_{A^TA}}, \zeta\Delta_k \right\}.$$

*Proof.* The proof follows that of Omojokun [61, Lemma 4.4]. Start by using (C1) in Theorem A.4.1 (Powell's Theorem) to show that the reduction in the objective of the vertical subproblem (1.5.1)-(1.5.2) satisfies

$$\|c_k\|_2^2 - \|A_k^T v + c_k\|_2^2 \geq \frac{1}{2}\|A_k c_k\|_2 \min\left\{ \frac{\|A_k c_k\|_2}{\|A_k A_k^T\|_2}, \zeta\Delta_k \right\}.$$

A simpler consequence of (C1) is that $v$ minimizes $\|A_k^T v + c_k\|_2$ as much as $v_k = \emptyset$; therefore, $\|c_k\|_2^2 \geq \|A_k^T v + c_k\|_2^2$, and

$$\begin{aligned} \|c_k\|_2 - \|A_k^T v + c_k\|_2 &= \frac{\|c_k\|_2^2 - \|A_k^T v + c_k\|_2^2}{\|c_k\|_2 + \|A_k^T v + c_k\|_2} \\ &\geq \frac{\|c_k\|_2^2 - \|A_k^T v + c_k\|_2^2}{2\|c_k\|_2} \\ &\geq \frac{1}{4}\frac{\|A_k c_k\|_2}{\|c_k\|_2} \min\left\{ \frac{\|A_k c_k\|_2}{\|A_k A_k^T\|_2}, \zeta\Delta_k \right\}. \end{aligned}$$

From taking norms of the identity $c = (A^TA)^{-1}A^TAc$ we have

$$\|A_k c_k\|_2 \geq \frac{\|c_k\|_2}{\|(A_k^T A_k)^{-1}\|_2 \ \|A_k^T\|_2}.$$

and from equations (A.2.3) and (A.2.2) we see that $\|A_k A_k^T\|_2 = \|A_k\|_2^2$. Substitution of these relations and the use of (A2) and (A3) yields the desired final bound. □

**Lemma 1.7.2** *Let problem (EP) satisfy (A1)-(A3), and let the horizontal step $u$ of the Byrd-Omojokun algorithm satisfy condition (C2). Also assume that the algorithm satisfies (C5) and (C6). For every $x_k \in \Omega$, then*

$$-\bar{g}_k^T Z_k u - \frac{1}{2} u^T Z_k^T W_k Z_k u \geq \frac{\|Z_k^T \bar{g}_k\|_2}{2\beta_Z} \min \left\{ \frac{\|Z_k^T \bar{g}_k\|_2}{\beta_Z^3 \beta_{Z^T Z} \beta_W}, \, \bar{\Delta}_k \right\},$$

*where $\bar{g}_k = g_k + W_k v_k$ and $\bar{\Delta}_k = \sqrt{\Delta_k^2 - \|v_k\|_2^2}$.*

*Proof.* The horizontal subproblem (1.5.7)-(1.5.8) is

$$\min_{u \in \mathbf{R}^{n-m}} \quad \bar{g}_k^T Z_k u + \frac{1}{2} u^T Z_k^T W_k Z_k u \quad \text{subject to} \quad \|Z_k u\|_2 \leq \bar{\Delta}_k.$$

By Theorem A.4.1 (Powell's Theorem), the reduction in this objective made by a step in accordance with condition (C2) is at least

$$-\bar{g}_k^T Z_k u - \frac{1}{2} u^T Z_k^T W_k Z_k u \geq \frac{1}{2}\sqrt{\tau} \, \min \left\{ \frac{\sqrt{\tau}}{\|(Z_k^T Z_k)^{-1/2} Z_k^T W_k Z_k (Z_k^T Z_k)^{-1/2}\|_2}, \, \Delta_k \right\},$$

where

$$\tau = \bar{g}_k^T Z_k (Z_k^T Z_k)^{-1} Z_k^T \bar{g}_k.$$

Using the triangle inequality $\|Z_k^T \bar{g}_k\|_2^2 = \bar{g}_k^T Z_k (Z_k^T Z_k)^{-1} (Z_k^T Z_k) Z_k^T \bar{g}_k \leq \tau \|Z_k^T Z_k\|_2$ along with (A.2.2) gives

$$\tau \geq \frac{\|Z_k^T \bar{g}_k\|_2^2}{\|Z_k^T Z_k\|_2} = \frac{\|Z_k^T \bar{g}_k\|_2^2}{\|Z_k\|_2^2}.$$

The bounds of (A2), (A3) and (C6) then produce the desired result. □

Now we are able to prove that the rule for increasing $\mu_{k+1}$ described in condition (C4) will not increase the penalty parameter indefinitely.

**Lemma 1.7.3** *If problem (EP) satisfies (A1)-(A3), and if the Byrd-Omojokun algorithm satisfies conditions (C1)-(C6), then*

$$\mu_k \leq \mu_{max} \quad \textit{for all } \; k,$$

*where*

$$\mu_{max} = \max\{\mu_0, \, 0.1 + 4\beta_A^4 \beta_{A^T A}^2 \beta_g + 2\beta_W \beta_A \beta_{A^T A} \max\{\Delta_{max}, \, \beta_A^4 \beta_{A^T A}^2 \beta_c\}\}.$$

*Proof.* It is clear from (C4) that the penalty parameters form a monotonically increasing sequence, and that every increase satisfies

$$\mu_{k+1} = 0.1 + \frac{d_k^T g_k + \frac{1}{2} d_k^T W_k d_k}{\|c_k\|_2 - \|A_k^T v_k + c_k\|_2}, \quad \text{with } \; \|c_k\|_2 > 0.$$

The numerator of this expression can be bounded by substituting $d_k = Z_k u_k + v_k$ and using Lemma 1.7.2 to eliminate terms corresponding to the objective (1.5.7) of the horizontal subproblem:

$$\begin{aligned} d_k^T g_k + \frac{1}{2} d_k^T W_k d_k &= (g_k + W_k v_k)^T Z_k u_k + \frac{1}{2} u_k^T Z_k^T W_k Z_k u_k + v_k^T g_k + \frac{1}{2} v_k^T W_k v_k \\ &\le v_k^T g_k + \frac{1}{2} v_k^T W_k v_k. \end{aligned}$$

Condition (C1) with Lemma 1.7.1 puts a lower bound on the denominator, and combining this with the improved numerator bound gives

$$\mu_{k+1} \le 0.1 + 4\beta_A \beta_{A^T A} \frac{v_k^T g_k + \frac{1}{2} v_k^T W_k v_k}{\min\left\{ \frac{\|c_k\|_2}{\beta_A^3 \beta_{A^T A}}, \zeta \Delta_k \right\}}.$$

There are two cases for determining the smallest possible denominator of this expression. First, suppose the minimum is given by the $\zeta\Delta_k$ term. The vertical subproblem trust region constraint (1.5.2) always keeps $\|v_k\|_2 \le \zeta\Delta_k$, and therefore

$$\begin{aligned} \mu_{k+1} &\le 0.1 + 4\beta_A \beta_{A^T A} \frac{\|v_k\|_2 \ \|g_k\|_2 + \frac{1}{2}\|v_k\|_2^2 \ \|W_k\|_2}{\zeta \Delta_k} \\ &\le 0.1 + 4\beta_A \beta_{A^T A} (\|g_k\|_2 + \frac{1}{2}\zeta \Delta_k \|W_k\|_2) \\ &\le 0.1 + 4\beta_A \beta_{A^T A} (\beta_g + 0.5 \Delta_{max} \beta_W). \end{aligned} \tag{1.7.1}$$

Now suppose that the minimum of the denominator is given by the left-hand term. By (C1), the largest possible vertical step at any iteration is $v_{nw} = -A_k (A_k^T A_k)^{-1} c_k$. We see then that

$$\begin{aligned} \mu_{k+1} &\le 0.1 + 4\beta_A^4 \beta_{A^T A}^2 \frac{\|v_k\|_2 \ \|g_k\|_2 + \frac{1}{2}\|v_k\|_2^2 \ \|W_k\|_2}{\|c_k\|_2} \\ &\le 0.1 + 4\beta_A^4 \beta_{A^T A}^2 \frac{(\|g_k\|_2 + \frac{1}{2}\beta_A \beta_{A^T A} \|c_k\|_2 \ \|W_k\|_2)\beta_A \beta_{A^T A} \|c_k\|_2}{\|c_k\|_2} \\ &\le 0.1 + 4\beta_A^4 \beta_{A^T A}^2 (\beta_g + 0.5 \beta_A \beta_{A^T A} \beta_c \beta_W). \end{aligned} \tag{1.7.2}$$

Combining (1.7.1) and (1.7.2) proves the lemma. □

As a consequence of Lemma 1.7.3, the merit function can be bounded from below under appropriate circumstances.

**Lemma 1.7.4** *If problem (EP) satisfies (A1)-(A4), and if the Byrd-Omojokun algorithm satisfies conditions (C1)-(C7), then the merit function $\psi(x, \mu_k) = f(x) + \mu_k \|c(x)\|_2$ is bounded from below for all iterates with $k$ sufficiently large.*

*Proof.* By (C4) the penalty parameter forms a monotonically increasing sequence, and by Lemma 1.7.3 this sequence is bounded from above; hence, there is some $K_2 \geq 0$ such that $\mu_k$ equals a constant $\bar{\mu}$ for all $k \geq K_2$. Then from (C7) we see that $\psi(x_k, \mu_k) = \psi(x_k, \bar{\mu})$ has its minimizers at points in the feasible set $\mathcal{F}$ for all $k \geq \max\{K_1, K_2\}$.

Now an elementary result of real analysis is that a continuous function defined on a compact set (that is, a closed and bounded set) is always bounded from below. The continuity of $c$ and assumption (A4) imply that every level set of $f$ in $\mathcal{F}$ is a compact set (or possibly several compact sets if $\mathcal{F}$ consists of many disjoint sets). Therefore, the continuous function $f$ must be bounded from below on $\mathcal{F}$. But at every point in $\mathcal{F}$ the merit function $\psi$ is equal to $f$. Thus, the minimizers of $\psi$, which occur in $\mathcal{F}$ for $k \geq \max\{K_1, K_2\}$, are bounded from below. □

The next lemma uses the continuity of $f$ and $c$ to show that the difference between the reduction in the actual and the model merit functions is proportional to the square of the size of the step $d$. This is a reasonable result because the model merit function (1.5.14) is derived from a truncated Taylor series of the actual merit function (1.5.13). The lemma depends only on second derivatives being bounded, allowing the algorithm to use any bounded approximation $W_k$ for the Hessian of the Lagrangian.

**Lemma 1.7.5** *Assume that problem (EP) satisfies (A1)-(A3), and that the step $d$ generated at $x_k$ by the Byrd-Omojokun algorithm satisfies conditions (C1) and (C2). Also assume that the algorithm satisfies (C3)-(C6). If $x_k \in \Omega$ and $x_k + d \in \Omega$, then*

$$|\text{a\_red} - \text{p\_red}| \leq \frac{1}{2}(\beta_f + \beta_W + \mu_{max} m^{3/2} \beta_c) \, \|d\|_2^2,$$

*where* a_red *is the actual reduction in the merit function due to the step $d$,* p_red *is the reduction in the model merit function due to the same step, and $\mu_{max}$ is from Lemma 1.7.3.*

*Proof.* The quantities a_red and p_red are defined in equations (1.5.17) and (1.5.15). Their difference is

$$\text{a\_red} - \text{p\_red} = f_k - f(x_k + d) - g_k^T d - \frac{1}{2} d^T W_k d + \mu^+ (\|A_k^T d + c_k\|_2 - \|c(x_k + d)\|_2),$$

where $\mu^+$ is the value of the penalty parameter after applying (C4). The terms corresponding to the objective of the merit function can be simplified by making a Taylor series expansion of $f(x_k + d)$ using the formula (A.3.3):

$$\begin{aligned} & f_k - f(x_k + d) - g_k^T d - \frac{1}{2} d^T W_k d \\ = \quad & f_k - [f_k + g_k^T d + \int_0^1 \int_0^1 t \, d^T \nabla^2 f(x_k + utd) d \;\; du \; dt] - g_k^T d - \frac{1}{2} d^T W_k d \\ = \quad & - \int_0^1 \int_0^1 t \, d^T \nabla^2 f(x_k + utd) d \;\; du \; dt - \frac{1}{2} d^T W_k d. \end{aligned}$$

Therefore,

$$
\begin{aligned}
|\text{a\_red} - \text{p\_red}| &\leq \left| \int_0^1 \int_0^1 t\, d^T \nabla^2 f(x_k + utd) d \;\; du\; dt \right| + \frac{1}{2} \left| d^T W_k d \right| \\
&\quad + \mu^+ \left| \|A_k^T d + c_k\|_2 - \|c(x_k + d)\|_2 \right| \\
&\leq \frac{1}{2}(\beta_f + \beta_W) \|d\|_2^2 + \mu^+ \left| \|A_k^T d + c_k\|_2 - \|c(x_k + d)\|_2 \right| \\
&\leq \frac{1}{2}(\beta_f + \beta_W) \|d\|_2^2 + \mu^+ \|A_k^T d + c_k - c(x_k + d)\|_2,
\end{aligned}
$$

the triangle inequality being used in the last step. The right-hand terms involving the constraints are all column vectors whose $i$th component takes the form $d^T \nabla c_k^i + c_k^i - c^i(x_k + d)$. Substituting another Taylor series for $c^i(x_k + d)$ and simplifying gives

$$
A_k^T d + c_k - c(x_k + d) = - \int_0^1 \int_0^1 t\, d^T \nabla^2 c^i(x_k + utd) d \;\; du\; dt.
$$

Now by the boundedness of the constraint Hessians we have

$$
\begin{aligned}
\|A_k^T d + c_k - c(x_k + d)\|_2 &\leq \sqrt{m}\, \|A_k^T d + c_k - c(x_k + d)\|_1 \\
&= \sqrt{m} \sum_{i=1}^{m} \left| \int_0^1 \int_0^1 t\, d^T \nabla^2 c^i(x_k + utd) d \;\; du\; dt \right| \\
&\leq \frac{1}{2} m^{3/2} \beta_c \|d\|_2^2.
\end{aligned}
$$

Using this and Lemma 1.7.3 proves the lemma. □

Now we will prove that the Byrd-Omojokun algorithm always takes a step away from a point $x_k$ that does not satisfy both first-order KKT conditions. Recall that a step $d$ is accepted by the algorithm if the ratio a_red/p_red is as big as the threshold $\eta$, where $\eta \in (0, 1)$. Lemma 1.7.5 shows that a_red and p_red are the same to first-order in $\|d\|_2$, so as $\|d\|_2$ goes to zero second-order differences vanish and a sufficiently small step must be accepted. The next lemma ties this concept to first-order KKT violations at $x_k$ by showing that the algorithm can always generate sufficiently small but finite-size steps at such points.

**Lemma 1.7.6** *Assume problem (EP) satisfies (A1)-(A3), and that the Byrd-Omojokun algorithm satisfies conditions (C1)-(C6). Let $x_k \in \Omega$ be a point generated by the Byrd-Omojokun algorithm such that for some $\epsilon_k > 0$ either $\|c_k\|_\infty \geq \epsilon_k$ , or $\|c_k\|_\infty = 0$ and $\|Z_k^T g_k\|_\infty \geq \epsilon_k$ . Then the algorithm finds an acceptable step $d_j$ after a finite number of iterations, and*

$$
\Delta_j \geq \Delta_{min} = \min \left\{ \Delta_0,\ \frac{1 - \eta}{\beta_f + \beta_W + \mu_{max} m^{3/2} \beta_c} \min \left\{ \frac{\zeta}{20 \beta_A \beta_{A^T A}},\ \frac{\epsilon_k}{\beta_Z} \right\} \right\},
$$

*where $\mu_{max}$ is from Lemma 1.7.3.*

*Proof.* This proof proceeds by analyzing the behavior of the sequence of trust region sizes $\{\Delta_0, \ldots, \Delta_k, \ldots\}$ generated by the Byrd-Omojokun algorithm. Let us start by considering the finite subset $\{\Delta_0, \ldots, \Delta_i, \ldots, \Delta_k\}$. The first size $\Delta_0$ is positive, and Algorithm 1.3.2 allows the trust region size to decrease only when a step is rejected. If $d_i$ is a rejected step, then $\Delta_{i+1} \leq \gamma \|d_i\|_2$, with $\gamma \in (0,1)$. The hypotheses of this lemma do not allow any iterate prior to $x_k$ to satisfy both first-order KKT conditions exactly, because then the algorithm would already have succesfully terminated. Hence, the iterates $x_0, \ldots, x_i, \ldots, x_k$ each violate a first-order KKT condition by some $\epsilon_i > 0$. But if $\|c_i\|_\infty \geq \epsilon_i$, then $\|c_i\|_2 \geq \|c_i\|_\infty \geq \epsilon_i$ and (C1) with Lemma 1.7.1 requires the vertical step $v_i$ to be nonzero. Alternatively, if $\|c_i\|_\infty = 0$ and $\|Z_i^T g_i\|_\infty \geq \epsilon_i$, then $v_i = \emptyset$, but (C2) with Lemma 1.7.2 requires the horizontal step $Z_i u_i$ to be nonzero. Therefore, the algorithm generates a step $d_i = v_i + Z_i u_i$ of finite size at each iteration, and the trust region size decreases to no less than a fraction of this finite size. It follows that $\Delta_k > 0$ for $k$ finite.

Now with $\Delta_k$ established as positive, we must prove that the sequence $\{\Delta_k, \Delta_{k+1}, \ldots\}$ does not converge to zero by rejecting every proposed step.

Suppose first that $\|c_k\|_\infty \geq \epsilon_k$ at $x_k$, and the algorithm is at iteration number $j > k$, having rejected all steps from iterations $k$ through $j-1$. Applying condition (C4) and Lemma 1.7.1 to p_red for the current step $d_j$ yields

$$\begin{aligned}
\text{p_red} &= -g_k^T d_j - \frac{1}{2} d_j^T W_k d_j + \mu^+ (\|c_k\|_2 - \|A_k^T d_j + c_k\|_2) \\
&\geq 0.1 (\|c_k\|_2 - \|A_k^T d_j + c_k\|_2) \\
&\geq \frac{1}{40 \beta_A \beta_{A^T A}} \min \left\{ \frac{\epsilon_k}{\beta_A^3 \beta_{A^T A}}, \zeta \Delta_j \right\}.
\end{aligned} \tag{1.7.3}$$

The trust region size $\Delta_j$ is constantly shrinking by the factor $\gamma$ as steps are rejected, so that

$$\Delta_j \leq \gamma \|d_{j-1}\|_2 \leq \gamma \Delta_{j-1} \leq \gamma^2 \|d_{j-2}\|_2 \leq \cdots \leq \gamma^{j-k} \|d_k\|_2. \tag{1.7.4}$$

Hence, there is some $J_1 > k$ such that $\zeta \Delta_{J_1}$ is the smaller term in (1.7.3), and so after rejecting enough steps,

$$\text{p\_red} \geq \frac{\zeta \Delta_j}{40 \beta_A \beta_{A^T A}}, \quad \text{for } j \geq J_1.$$

Now apply Lemma 1.7.5 to obtain

$$\left| \frac{\text{a\_red}}{\text{p\_red}} - 1 \right| = \frac{|\text{a\_red} - \text{p\_red}|}{\text{p\_red}} \leq \tau_1 \frac{\|d_j\|_2^2}{\Delta_j} \leq \tau_1 \Delta_j, \tag{1.7.5}$$

where

$$\tau_1 = \frac{20}{\zeta} \beta_A \beta_{A^T A} (\beta_f + \beta_W + \mu_{max} m^{3/2} \beta_c).$$

This inequality holds for all rejected steps $d_j$ with $j \geq J_1$. By (1.7.4), $\Delta_j$ goes to zero as $j$ goes to infinity, so we see that

$$\lim_{j\to\infty} \left| \frac{\text{a\_red}}{\text{p\_red}} - 1 \right| = 0.$$

This implies that as $j \to \infty$ the ratio a_red/p_red converges to one. But when some step causes the ratio to exceed the tolerance $\eta \in (0, 1)$, then that step will be accepted; therefore, we conclude that only a finite number of steps can be rejected if $\|c_k\|_\infty \geq \epsilon_k$. Looking again at (1.7.5), we see that a step $d_j$ can be rejected only if

$$\tau_1 \Delta_j > 1 - \eta.$$

Therefore, the Byrd-Omojokun algorithm will not permit the trust region size to be smaller than $(1 - \eta)/\tau_1$ when $\|c_k\|_\infty \geq \epsilon_k$.

Now consider the other possibility at $x_k$, that $\|c_k\|_\infty = 0$ and $\|Z_k^T g_k\|_\infty \geq \epsilon_k$. Again presume that the algorithm is at iteration number $j > k$, having rejected all steps from iterations $k$ through $j - 1$. In this case $v_j = \emptyset$ and $d_j = Z_k u_j$; thus, by Lemma 1.7.2,

$$\begin{aligned} \text{p\_red} &= -g_k^T Z_k u_j - \frac{1}{2} u_j^T Z_k^T W_k Z_k u_j \\ &\geq \frac{\epsilon_k}{2\beta_Z} \min\left\{ \frac{\epsilon_k}{\beta_Z^3 \beta_{Z^TZ} \beta_W}, \Delta_j \right\}. \end{aligned} \tag{1.7.6}$$

There is some $J_2 > k$ such that $\Delta_{J_2}$ is the smaller term in (1.7.6), so that after rejecting enough steps,

$$\text{p\_red} \geq \frac{\epsilon_k}{2\beta_Z} \Delta_j, \quad \text{for } j \geq J_2.$$

Using Lemma 1.7.5 now gives

$$\left| \frac{\text{a\_red}}{\text{p\_red}} - 1 \right| \leq \tau_2 \Delta_j, \quad \text{with } \tau_2 = \frac{\beta_Z}{\epsilon_k} (\beta_f + \beta_W + \mu_{max} m^{3/2} \beta_c).$$

Reasoning as before, we see that

$$\lim_{j\to\infty} \left| \frac{\text{a\_red}}{\text{p\_red}} - 1 \right| = 0,$$

which implies that only a finite number of steps can be rejected before the threshold $\eta < 1$ is reached by some $d_j$. We conclude that the algorithm does not permit the trust region size to be smaller than $(1 - \eta)/\tau_2$ when $\|c_k\|_\infty = 0$ and $\|Z_k^T g_k\|_\infty \geq \epsilon_k$. Combining this lower bound with the bound obtained for the case when $\|c_k\|_\infty \geq \epsilon_k$ shows that

$$\Delta_j \geq (1 - \eta) \min\left\{ \frac{1}{\tau_1}, \frac{1}{\tau_2} \right\}$$

for $j \geq k$ under the hypotheses of this lemma. This bound accounts for all reductions in the trust region size due to rejected steps. As argued at the beginning of this proof, there is no other way for the algorithm to make the trust region size smaller than its initial value of $\Delta_0$; thus, we have determined $\Delta_{min}$. □

Now we have all the ingredients necessary to prove global convergence. The preceding lemma proves that the Byrd-Omojokun algorithm takes a finite-size step away from a point $x_k$ that violates the first-order KKT conditions. A finite-size step also makes a finite reduction in the merit function $\psi$, and from this global convergence may be deduced. Note that the method of proof does not depend on directional derivatives of $\psi$ at the final iterate. As mentioned earlier, this reflects the fact that the Byrd-Omojokun algorithm does not directly minimize the merit function.

**Theorem 1.7.7** *Assume problem (EP) satisfies (A1)-(A3), and that the Byrd-Omojokun algorithm satisfies conditions (C1)-(C6). Then as the iteration number $k \to \infty$, the Byrd-Omojokun algorithm generates a sequence of points $\{x_0, \ldots, x_k, \ldots\}$ such that*

$$\lim_{k\to\infty} \inf\{\|c_k\|_\infty + \|Z_k^T g_k\|_\infty\} = 0.$$

*Proof.* Let us start by examining the reduction in the merit function made by a successful step from a point $x_k$ where $\|c_k\|_\infty + \|Z_k^T g_k\|_\infty > 0$. Suppose first that $\|c_k\|_\infty = \epsilon > 0$. Lemma 1.7.6 showed that the algorithm finds an acceptable step from $x_k$ after a finite number of iterations. To be accepted, a step must make a reduction in the merit function that satisfies a_red $\geq \eta$ p_red. Using (1.7.3) in this inequality yields

$$\text{a\_red} \geq \frac{\eta}{40\beta_A\beta_{A^TA}} \min\left\{\frac{\epsilon}{\beta_A^3\beta_{A^TA}},\, \zeta\Delta_{min}\right\}.$$

Now suppose that $\|c_k\|_\infty = 0$ and $\|Z_k^T g_k\|_\infty = \epsilon > 0$. Lemma 1.7.6 again proves that a step from $x_k$ is accepted in a finite number of iterations, and from (1.7.6) the reduction in the merit function satisfies

$$\text{a\_red} \geq \frac{\eta\epsilon}{2\beta_Z} \min\left\{\frac{\epsilon}{\beta_Z^3\beta_{Z^TZ}\beta_W},\, \Delta_{min}\right\}.$$

In either case we see that a consequence of Lemma 1.7.6 is that an accepted step makes a finite reduction in the merit function. This is true even if the accepted step contains a second-order correction (see §1.5.7), because p_red is computed for the step $d_k$, not $d_{\text{SOC}}$. Thus, (1.7.3) and (1.7.6) are not affected by a second-order correction.

To complete the proof, let us make the hypothesis that all iterates in the sequence $\{x_0, \ldots, x_k, \ldots\}$ are bounded away from satisfying both first-order KKT conditions; that is, assume $\|c_k\|_\infty + \|Z_k^T g_k\|_\infty \geq \epsilon > 0$ for all $k$. Lemma 1.7.4 proves that the merit function $\psi$ is bounded from below for all $k \geq \max\{K_1, K_2\}$. But we have just shown that at every point violating the KKT conditions a step is found after a finite number of iterations that makes a finite reduction in $\psi$. If no step satisfies both first-order KKT conditions, then $\psi$ is reduced indefinitely, contradicting the fact that it is bounded from below. Therefore, not all iterates are bounded away from satisfying both first-order KKT conditions, and for some subsequence of iterates the quantity $\|c_k\|_\infty + \|Z_k^T g_k\|_\infty$ converges to zero. □

Note that Theorem 1.7.7 could be made slightly stronger by proving that

$$\lim_{k \to \infty} \|c_k\|_\infty + \|Z_k^T g_k\|_\infty = 0.$$

To do this requires bounding the reduction of $\psi$ in terms of $\epsilon$, as is done in [71] for unconstrained optimization algorithms. Formulating this bound appears to be quite complicated, and Theorem 1.7.7 is still strong enough to show that the Byrd-Omojokun algorithm reaches a first-order solution of (EP).

Theorem 1.7.7 makes a precise statement of the assumptions necessary for the Byrd-Omojokun algorithm to generate a sequence of iterates that converge to a point satisfying the first-order KKT conditions for problem (EP). Assumptions (A1)-(A4) restrict the set of possible problems that can be solved by the algorithm, ruling out ill-posed problems. However, they also rule out some well-posed problems; for example ORTHREGD, which violates (A3) at its solution point. This particular assumption can probably be relaxed to some degree and a weaker form of global convergence proved, perhaps similar to [19].

Conditions (C1)-(C7) restrict the implementation of the general Byrd-Omojokun algorithm. The ETR implementation does not satisfy the conditions, and its shortcomings will now be explained. Recall from §1.5.3 that the horizontal subproblem does not compute $u_{cp}$ exactly because $(Z_k^T Z_k)^{-1}$ is approximated by an inner CG iteration. This means (C2) is not strictly observed. However, (C2) can be relaxed in the manner suggested in [12] to require the exact Cauchy reduction only as $k \to \infty$. This revised condition is met by ETR. Condition (C4) is violated by the heuristic of §1.5.6, which decreases the penalty parameter $\mu$ in some circumstances. This could allow $\mu$ to alternately be decreased by the heuristic at one iteration and then increased by (1.5.16) at a later iteration, cycling up and down without staying fixed long enough for the minimum of the merit function to be reached. The heuristic can be easily safeguarded from cycling indefinitely by using the idea of El-Alem [28]. Since cycling of $\mu$ has not been observed with ETR, this improvement has not yet been made.

This section has provided only a theorem for global convergence to first-order KKT points of problem (EP). The Byrd-Omojokun algorithm can be extended to generate a sequence of iterates that also converge to a point satisfying the second-order KKT condition – this requires modifying the stop test in Algorithm 1.3.2 and placing an additional condition on the method used to solve the horizontal subproblem. With these refinements, global convergence to first- and second-order points can be proved. Unfortunately, the current implementation of ETR does not satisfy the additional condition for solving the horizontal subproblem. The condition requires that a reduction in the horizontal subproblem objective be made if $Z_k^T W_k Z_k$ is indefinite, even when $Z_k^T g_k = \emptyset$. The method of Steihaug used in ETR cannot do this; hence, ETR does not provide second-order global convergence.

## 1.8 Summary and Conclusions

During the last ten years much work has been devoted to the development of trust region methods for equality constrained optimization. However, with the exception of LANCELOT [17], none of the algorithms proposed have received a careful implementation suitable for solving large problems. This dissertation undertook the task of implementing a proposed method, the algorithm of Byrd [8] and Omojokun [61], and observing its performance on large-scale and ill-conditioned problems. Sections 1.2, 1.3 and 1.7 presented a theoretical analysis of the Byrd-Omojokun algorithm and placed it in the context of other trust region research. Sections 1.4 and 1.5 supplied a detailed implementation of the algorithm with emphasis on solving large problems efficiently. The test results in section 1.6 demonstrated that the ETR algorithm performs well and behaves significantly different from the LANCELOT algorithm. Thus, the dissertation work has resulted in a viable software alternative for solving large-scale equality constrained optimization problems.

The success of this study so far justifies and motivates further work on the topic. Ideas for future investigations may be grouped into two categories: improvements to the ETR algorithm, and research into areas of algorithm analysis. Besides enhancements to the software portability and user interface, the ETR algorithm can probably be improved in its performance and robustness by the following:

- Investigating the use of a sparse QR factorization of $A_k^T$. If the factorization does not become too dense, this provides a $Z_k$ matrix that is much better conditioned.
- Implementing a range space or dual method for solving the Byrd-Omojokun subproblem (1.3.16)-(1.3.18). This eliminates the need to compute $Z_k$ and to have an explicit constraint Jacobian $A_k$ (only the ability to multiply $A_k$ and $A_k^T$ by a vector is required). This method can be shown to compute the exact same iterates as those generated by ETR when the horizontal subproblem is solved by the dogleg method using an $\ell$-BFGS approximation for $W_k$.
- Applying preconditioners to the many subproblems solved by conjugate gradient iterations. This is a nontrivial task, since the matrices to be preconditioned are products such as $A_k^T A_k$ and $Z_k^T W_k Z_k$.
- Solving linear systems involving $A_k^T A_k$ by the augmented system approach [24], [35]. The augmented system matrix provides a better-conditioned alternative to solving the normal equations, and may preserve the sparsity of $A_k$ better than the Cholesky factorization of $A_k^T A_k$.
- Trying a limited memory SR1 approximation for $W_k$ as an alternative to the $\ell$-BFGS method implemented. Since the true Hessian of the Lagrangian can be indefinite at a solution, SR1 might be a more natural fit than BFGS; on the other hand, the horizontal subproblem can no longer be solved by the inexpensive dogleg method.
- Using a reduced Hessian approximation of the Lagrangian for solving the horizontal subproblem. This should be more efficient than the current technique of approximat-

ing the full Hessian $W_k$, projecting it onto the tangent space of the constraints, and then applying the dogleg method. However, a reduced Hessian approach makes the generalization of ETR to problems with inequality constraints (discussed in the next chapter) much more complicated.

- Developing a heuristic that varies the parameter $\zeta$ each iteration depending on the rate at which constraint feasibility is being reached. Recall that $\zeta$ controls the fraction of the trust region that can be used by a vertical step, and consequently the minimum size of the trust region in the horizontal subproblem. Preliminary data indicates that test problems HS111LNP, ORTHREGA, ORTHREGC and ORTHREGD can be solved with significantly fewer iterations for a properly tuned value of $\zeta$. These are all problems that start at a point far from satisfying the nonlinear equality constraints.

Some interesting questions in algorithm analysis were uncovered in this disquisition and not fully resolved. The most important ones are:

- What is the proper course of action when $A_k$ is rank deficient? In some cases the algorithm may be hopelessly trapped at an infeasible stationary point, while in others (for instance, test problem ORTHRGDS) it may simply need an alternative way of computing the vertical step $v_k$ and the matrix $Z_k$. A related issue is to define the conditions under which the Byrd-Omojokun algorithm can converge to a solution if $A_k$ is not assumed full rank.

- Can global convergence of the algorithm be proved without requiring conditions on the behavior of the merit function penalty parameter? This was conjectured at the end of §1.7.1.

- How can the horizontal subproblem be preconditioned, or more generally, how can Steihaug's method for unconstrained trust region problems be preconditioned and prevented from terminating with poor solution estimates? Recall that these issues were raised in §1.4.6. A heuristic for determining when to terminate the method might be successful, but more fundamental results also seem obtainable.

- Can the Byrd-Omojokun algorithm and the ETR implementation be made less sensitive to scaling? Trust region methods are inherently scale-dependent, but this does not mean the problem is intractable.

# Chapter 2

# Inequality Constraints

## 2.1 Introduction

This chapter explores an extension of the work described in the previous chapter, seeking to solve optimization problems subject to inequality constraints as well as equality constraints. The results reported are preliminary but interesting enough to warrant further investigation. It should be emphasized that the method presented here is only one of many possible extensions of the Byrd-Omojokun theory and ETR algorithm. It is likely to be an efficient method for only certain types of problems, but these problems are important and difficult to solve by other means. Consistent with the theme of this dissertation, the method is especially suited to large-scale problems.

The general nonlinear constrained optimization problem is

$$\begin{array}{rlll} \min\limits_{x \in \mathbf{R}^n} & f(x) & & \\ \text{subject to} & c^i(x) = 0, & \text{for} & i = 1, \ldots, m_{eq} \\ & c^i(x) \geq 0, & \text{for} & i = 1, \ldots, m_{ineq}. \end{array}$$

Through the use of slack variables, this can be converted to an equivalent problem with more unknowns but containing only equality constraints and bound constraints on the variables [39]. This section therefore takes the optimization problem to be in the form

$$\begin{array}{rlll} \min\limits_{x \in \mathbf{R}^n} & f(x) & & \\ \text{subject to} & c^i(x) = 0, & \text{for} & i = 1, \ldots, m \\ & u^i \geq x^i \geq l^i, & \text{for} & i = 1, \ldots, n. \end{array} \tag{BEP}$$

The functions $f$ and $c^i$ are assumed twice continuously differentiable, and $n \geq m \geq 0$. Variables may be unbounded in either direction, so the scalar constants $l^i$ and $u^i$ are such that $l^i \in [-\infty, u^i)$ and $u^i \in (l^i, +\infty]$; in particular, $l^i < u^i$.

To make the best use of the ETR software it is desirable to somehow solve (BEP) using only equality constrained subproblems of the form (EP). A standard approach is to use the active set method [39], [34], solving subproblems in which each bound constraint is treated either as an equality constraint ("active") or temporarily ignored ("inactive"). This method has the desired property of solving (BEP) iteratively with subproblems possessing only equality constraints. Unlike reduced Hessian algorithms, ETR is easily extended to accommodate an active set method because the changing number of "active" constraints in each subproblem poses no extra computational or theoretical burdens. Indeed, Omojokun developed a successful active set extension in his work [61], but only for small problems. The active set method can be efficient for large problems, but only if a good estimate of the final active set of constraints is known at the start. Otherwise, it is inefficient because any constraints wrongly guessed "active" are made "inactive" one at a time, a complete subproblem being solved for each change in the active set. This has the potential of being combinatorially slow if a large number of the possible active set choices are tried before reaching a solution. The remainder of this paper focuses on an entirely different method that should be efficient for large problems lacking information about the final active set.

## 2.2 Derivation of the Primal Barrier Method

The new method is related to interior point ideas that have been applied recently to solve large-scale linear programming problems (a good reference is the review article by Gonzaga [42]). ¿From a nonlinear programming perspective it is related to barrier methods (see [39] or [34], or the classic reference [30]), as emphasized in the review paper [80]. Mixing terminology from both areas, the method is referred to here as a "primal barrier" extension of the Byrd-Omojokun algorithm.

Roughly speaking, the primal barrier method generates iterates converging to a solution that attempt to avoid all inequality bounds for as long as is practical. The interior path marked by these iterates has the advantage of being shorter than an active set path which makes its way along the boundaries of the problem. Like interior point methods in linear programming, the primal barrier method is also free of any combinatorial complications. It is derived below as a generalization of Karmarkar-like interior point methods used in linear programming. In carrying out a generalization to nonlinear nonconvex problems, many details important to linear programming algorithms but having little or no value in this broader context are ignored; hence, the following presentation does not attempt to capture all the features of Karmarkar-like methods.

### 2.2.1 Primal Barrier Methods for Linear Programming

Consider the linear programming equivalent of problem (BEP):

$$\begin{array}{ll} \min\limits_{x\in\mathbf{R}^n} & f(x) = g^T x \\ \text{subject to} & A^T x = b \\ & \quad x^i \geq 0, \quad \text{for } i = 1, \ldots, n \ , \end{array} \tag{LP}$$

where $g \in \mathbf{R}^n$, $b \in \mathbf{R}^m$ and $A^T \in \mathbf{R}^{m\times n}$ are all constant. Note that in (LP) the Jacobian matrix $A^T$ is $m \times n$ with $n \geq m$; this notation is used to maintain consistency with the previous chapter.

There are three basic ideas underlying primal interior point methods for linear programming: affine scaling, addition of a constraint barrier term, and proper choice of the step length. The first two are present in the new algorithm for nonconvex problems, but the essence of the last idea is not. Let us sketch out an algorithm embodying these three principles for solving (LP). The algorithm generates a sequence of iterates $\{x_0, \ldots, x_k, \ldots\}$ by solving subproblems derived from (LP) that contain only equality constraints. All of the iterates lie in the interior of the bound constraint inequalities; that is, $x_k^i > 0$ for every iteration $k$ and for every component $i = 1, \ldots, n$.

Affine scaling, a concept attributed to Dikin [21], generates the direction of the next iterate by solving a scaled subproblem derived from (LP). The current iterate $x_k$ is assumed to satisfy $A^T x_k = b$ and to be in the positive orthant of $\mathbf{R}^n$. The scaling transformation at such a point is defined by

$$\begin{aligned} \hat{x} &= D_k^{-1} x \\ \hat{g} &= D_k g \\ \hat{A}^T &= A^T D_k, \end{aligned} \tag{2.2.1}$$

where $D_k$ is the diagonal scaling matrix

$$D_k = \operatorname{diag}\{x_k^1, \ldots, x_k^n\}.$$

A search direction $d_k$ is found by solving an affinely scaled subproblem, and then the next iterate is computed as $x_{k+1} = x_k + \alpha_k d_k$ for step length $\alpha_k > 0$. The scaled subproblem is

$$\min_{\hat{d} \in \mathbf{R}^n} \ \hat{g}^T \hat{d} \qquad \text{subject to } \hat{A}^T \hat{d} = \emptyset, \tag{2.2.2}$$

which can be obtained by applying the transformation (2.2.1) to (LP) using $x = x_k + d$, and dropping the bound constraints. The dropped bound constraints will not be violated so long as the step length in scaled coordinates is less than one. This is true because the current iterate $x_k$ transforms to the point $\hat{x}_k = [1 \ \ \cdots \ \ 1]^T$, which is exactly a unit step away from every bound constraint.

As it stands, the solution of (2.2.2) is unbounded. We are only interested in finding the direction to this solution – its length will be determined by other criteria. Adding the constraint $\|\hat{d}\|_2 = 1$ to (2.2.2) makes the solution bounded without biasing its direction. The solution with this added constraint is given by

$$\hat{d}_k = -\frac{\bar{P}_{\hat{A}} \hat{g}}{\|\bar{P}_{\hat{A}} \hat{g}\|_2}, \tag{2.2.3}$$

where $\bar{P}_{\hat{A}} = I - \hat{A}(\hat{A}^T \hat{A})^{-1} \hat{A}^T$ is the operator that orthogonally projects vectors onto the null space of $\hat{A}^T$. The search direction in the original problem coordinates is then

$$d_k = D_k \hat{d}_k = -\frac{[I - D_k^2 A (A^T D_k^2 A)^{-1} A^T] D_k^2 g}{\|\bar{P}_{\hat{A}} \hat{g}\|_2}. \tag{2.2.4}$$

Observe that when $x_k^i$ is very close to its bound constraint, the $i$th component of $D_k^2 g$ and $d_k^i$ are both very small. This illustrates the general effect of affine scaling, which is to deflect iterates away from the bounds.

Affine scaling by itself defines an interior point method for solving (LP). Start at a point $x_0$ that is in the interior and satisfies $A^T x_0 = b$, then generate new iterates by computing $d_k$ from (2.2.4) and setting $x_{k+1} = x_k + \alpha_k d_k$, with step length $\alpha_k < 1$. Assuming the solution to (LP) is nondegenerate, this algorithm can be shown to be globally convergent with a linear rate of convergence (see [42, p. 178]). The main problem with using affine scaling alone is that it cannot move away from a bound if an iterate $x_k$ is close to a bound. Being close makes the subproblem (2.2.2) ill-conditioned and difficult to solve. Remaining close to a bound makes the interior path longer and slows down the rate of convergence.

The second major idea of interior point methods, adding a constraint barrier term, provides a specific mechanism for iterates to move away from bound constraints. The idea is to add a term to the objective of the problem that imposes a penalty for being too close to the bounds; hence, the minimum of the modified objective is pushed into the interior. The barrier term is often called a potential function, and its effect on the solution described as "centering". The most commonly used barrier is the logarithmic function [37]. Adding it to the objective of (LP) gives the new problem

$$\begin{array}{lll} \displaystyle\min_{x\in\mathbf{R}^n} & g^T x - \rho \displaystyle\sum_{i=1}^{n} \ln x^i & \\ \text{subject to} & A^T x = b & \\ & x^i \geq 0, & \text{for } i = 1, \ldots, n, \end{array} \tag{2.2.5}$$

where $\rho \geq 0$ is the weight of the barrier. When $\rho = 0$ then (2.2.5) reduces to (LP), but when $\rho > 0$ the solution of (2.2.6) lies in the interior and all the bound constraints are inactive. Now imagine solving a sequence of problems of the form (2.2.5), each with a different $\rho_k$ chosen from a sequence $\{\rho_0, \ldots, \rho_k, \ldots\}$ that converges to zero. It can be proved that if the feasible region of (LP) is bounded (or, equivalently, if the feasible region of the dual of (LP) has a nonempty interior), then the sequence of solutions to these problems converges to the solution of (LP). This result can be generalized to the larger class of convex optimization problems [80].

The path followed by this sequence of solutions is called the central path, a name that reflects the geometric interpretation of the solution of (2.2.5). If the feasible region of (LP) is bounded and just the barrier term is minimized in (2.2.6), i.e.,

$$\min_{x\in\mathbf{R}^n} \; -\sum_{i=1}^{n} \ln x^i \qquad \text{subject to } \; A^T x = b,$$

then the solution lies at the center of the feasible region (formally known as the analytic center). Thus, one end of the central path (when $\rho$ is large) is far from all the bounds, while the other end (when $\rho = 0$) lies at the solution of (LP) where many bounds are active.

The barrier term combines with affine scaling in a particularly useful manner. Consider the barrier subproblem (2.2.5) with its bounds dropped:

$$\min_{x\in\mathbf{R}^n} \; g^T x - \rho \sum_{i=1}^{n} \ln x^i \qquad \text{subject to } \; A^T x = b. \tag{2.2.6}$$

This is a nonlinear problem with equality constraints, but it is convex inside the positive orthant of $\mathbf{R}^n$ and the barrier is infinitely high at all the edges of the positive orthant; therefore, it has a unique solution in the interior of the feasible region (see [80] for a more precise statement and proof). These properties imply that we can iterate to the solution of the subproblem using Newton's method starting from any initial interior point. Let us calculate the Newton step $d_{nw}$ from a feasible point $x_k$ and examine its direction.

Section §1.2.3 derives Newton's method for solving problems with equality constraints in terms of the Lagrangian, which for (2.2.6) is

$$\mathcal{L}(x,\lambda) = g^T x - \rho \sum_{i=1}^{n} \ln x^i - \lambda^T (A^T x - b).$$

The multipliers $\lambda$ can be treated as dual variables for (2.2.6), but in the primal barrier method investigated here their interpretation is unimportant because they end up being eliminated. The Newton step from a given point $[x_k \;\; \lambda_k]^T$ is found as in (1.2.7)-(1.2.8) by solving the augmented system

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L}(x_k,\lambda_k) & \nabla^2_{x\lambda}\mathcal{L}(x_k,\lambda_k) \\ \nabla^2_{\lambda x}\mathcal{L}(x_k,\lambda_k) & \nabla^2_{\lambda\lambda}\mathcal{L}(x_k,\lambda_k) \end{bmatrix} \begin{bmatrix} d_{nw} \\ \lambda_{nw} \end{bmatrix} = - \begin{bmatrix} \nabla^2_{x}\mathcal{L}(x_k,\lambda_k) \\ \nabla^2_{\lambda}\mathcal{L}(x_k,\lambda_k) \end{bmatrix}.$$

Evaluating all the derivatives gives

$$\begin{bmatrix} \rho D_k^{-2} & -A \\ -A^T & \emptyset \end{bmatrix} \begin{bmatrix} d_{nw} \\ \lambda_{nw} \end{bmatrix} = - \begin{bmatrix} g - \rho D_k^{-1} e - A\lambda_k \\ A^T x_k - b \end{bmatrix},$$

where $e$ is the vector $[1 \;\; \cdots \;\; 1]^T$ whose components are all equal to one. Recalling that $x_k$ is feasible and defining the quantity $\lambda_{k+1} = \lambda_k + \lambda_{nw}$, the augmented system simplifies to the two vector equations

$$\begin{aligned} \rho D_k^{-2} d_{nw} - A\lambda_{k+1} &= -g + \rho D_k^{-1} e \\ A^T d_{nw} &= \emptyset. \end{aligned}$$

Multiplying the top equation by $A^T D_k^2$ and using the bottom equation solves for $\lambda_{k+1}$, which can be eliminated to finally give

$$\begin{aligned} d_{nw} &= \frac{1}{\rho} D_k^2 [A(A^T D_k^2 A)^{-1} A^T D_k^2 - I](g - \rho D_k^{-1} e) \\ &= -\frac{1}{\rho} D_k \bar{P}_{\hat{A}} (\hat{g} - \rho e). \end{aligned} \tag{2.2.7}$$

The surprising feature of the Newton step is that it contains the scaling transformation matrix $D_k$ even though affine scaling was not applied. It turns out that a vector in the same direction is obtained by taking the gradient of the objective of (2.2.6) at $x_k$, affinely scaling it to get a subproblem similar to (2.2.2), and then solving for the scaled steepest descent direction. Hence, the interior point search direction can be found either by adding the nonlinear logarithmic barrier term to the objective and computing a Newton step, or by adding a *linearized* version of the logarithmic barrier to the objective, affinely scaling that subproblem, and computing its direction of scaled steepest descent.

At this point a primal interior point algorithm for solving (LP) can be outlined.

**Algorithm 2.2.1** *General description of a primal barrier interior point method*

> Start with $x_0$ such that $x_0^i > 0$ for $i = 1, \ldots, n$ and $A^T x_0 = b$
> Choose $\rho_0 > 0$
> LOOP, starting with $k = 0$
>   IF $x_k$ is sufficiently close to a solution of (LP) THEN STOP
>   Compute $\hat{g}$ and $\hat{A}$ from the affine scaling (2.2.1)
>   Compute the step $d_{nw}$ using (2.2.7)
>   Choose a step length $\alpha_k > 0$
>   $x_{k+1} = x_k + \alpha_k d_{nw}$
>   Choose $\rho_{k+1} < \rho_k$
> CONTINUE, after incrementing $k$

A third essential idea in primal barrier methods for solving (LP) concerns the selection of $\rho_k$ and $\alpha_k$ in Algorithm 2.2.1. At a minimum, the sequence of barrier weights must converge to zero and the step lengths must generate iterates that stay in the interior. It turns out that for linear programming the choice of $\rho_k$ and $\alpha_k$ can be linked and defined by certain guidelines, and then the convergence of Algorithm 2.2.1 can be proved for $x_0$ sufficiently close to the central path.

This concludes the overview of primal barrier interior point methods for linear programming. Much more can be said about this subject, and Algorithm 2.2.1 is not a very strong representation of the sophisticated algorithms used to solve (LP) in practice, but it is sufficient as a conceptual outline of the method that is to be generalized.

### 2.2.2 Generalization to Nonlinear Programming

Before carrying out the mathematical generalization of Algorithm 2.2.1, let us mention a few of the things that can happen in nonlinear nonconvex problems that do not occur in linear programming. Perhaps the most important distinction to keep in mind is that the solution does not have to be at a vertex, but may be in the interior of the bounded region. It is also possible for a variable to have no bound constraints, or to have two, an upper and a lower, which may be very close together (this is the situation in certain test problems of §2.4). Nonlinear equality constraints may not be satisfied at an iterate, and even if they are the next iterate may fail to satisfy them. Nonlinear inequalities may define a nonconvex region, and if the constraint is "hard" (i.e., cannot be violated), then an algorithm may have to find a path that circumnavigates the forbidden region.

Many of the difficulties listed above imply that it is harder to obtain convergence for nonlinear problems than for (LP). For this reason, the primal barrier algorithm proposed here is a trust region method. The goal of this section is to use Algorithm 2.2.1 as a guide for deriving a general equality constrained subproblem with a trust region that can be solved by the Byrd-Omojokun algorithm.

As others have noted [42], [2], a trust region is implicitly present in the method of affine scaling for linear problems. This is apparent from the bound $\|\hat{d}\|_2 = 1$ that must be added to subproblem (2.2.2) in order to find the scaled search direction (2.2.3). Note that the bound could be replaced with an inequality constraint, because in linear programming it would always be active. The affine scaling method for solving (LP) uses the bound to find the search direction of a step, but not its length. However, in nonlinear programming the size of the trust region controls the step length directly and must be adjusted at each iteration in a manner that causes convergence to a solution.

The trust region constraint $\|\hat{d}\|_2 = 1$ is an unbiased hypersphere in the transformed coordinates so as not to disturb the effect of the scaling transformation. By (2.2.1), the constraint transforms to $\|D_k^{-1}d\|_2 = 1$ in the original coordinates of problem (LP). The equivalent equation $d^T D_k^{-2} d = 1$ defines a surface traced out by the variables $d$. This surface is a hyperellipsoid with center at $x_k$ and orthogonal axes that exactly touch each of the coordinate planes $x^i = 0$. Thus, imposing a spherical trust region in affinely scaled coordinates is equivalent to using an ellipsoidal trust region in the original coordinates.

Now we are ready to state a generalization of the affine scaling method for solving problem (BEP). The scaling transformation is still defined by a diagonal matrix $D_k$, with its $i$th component equal to the distance from $x_k^i$ to the nearest bound constraint; i.e.,

$$D_k^{ii} = \begin{cases} x_k^i - l^i & \mathsf{IF}\ l^i \text{ is finite and } u^i \text{ is not} \\ u^i - x_k^i & \mathsf{IF}\ u^i \text{ is finite and } l^i \text{ is not} \\ \min\{x_k^i - l^i, u^i - x_k^i\} & \mathsf{IF} \text{ both } l^i \text{ and } u^i \text{ are finite} \\ 1 & \mathsf{IF} \text{ neither } l^i \text{ or } u^i \text{ is finite.} \end{cases} \tag{2.2.8}$$

The last case of (2.2.8) leaves a variable with no bounds unchanged by the transformation.

A nonlinear optimization problem having only equality constraints $c(x) = \emptyset$ may be solved by a Newton method, with the step at $x_k$ found by solving the subproblem (see (1.2.28))

$$\min_{d \in \mathbf{R}^n} \ d^T g_k + \frac{1}{2} d^T W_k d \qquad \text{subject to } \ A_k^T d + c_k = \emptyset.$$

Recall that the matrix $W_k$ is the Hessian of the Lagrangian $\mathcal{L}(x, \lambda) = f(x) - \lambda^T c(x)$, and that all subscripted quantities are evaluated at the point $x_k$. Defining the transformations

$$\begin{aligned} \hat{d} &= D_k^{-1} d \\ \hat{g}_k &= D_k g_k \\ \hat{A}_k^T &= A_k^T D_k \\ \hat{W}_k &= D_k W_k D_k, \end{aligned} \tag{2.2.9}$$

an affinely scaled version of the Newton subproblem can be obtained analogous to (2.2.2) for problem (LP). To globalize Newton's method, a trust region constraint is added, giving the trust region subproblem

$$\begin{aligned} \min_{\hat{d} \in \mathbf{R}^n} \quad & \hat{d}^T \hat{g}_k + \frac{1}{2} \hat{d}^T \hat{W}_k \hat{d} \\ \text{subject to} \quad & \hat{A}_k^T \hat{d} + \hat{c}_k = \emptyset \\ & \|\hat{d}\|_2 \le \Delta_k. \end{aligned} \tag{2.2.10}$$

Remember that affine scaling allows the bound constraints of (BEP) to be dropped so long as the trust region size $\Delta_k$ is properly chosen. Note that the transformation (2.2.9) preserves the sparsity of matrices $A_k$ and $W_k$, and that its computational cost is small.

An important property of the trust region size in (2.2.10) is that when $\Delta_k < 1$ and $x_k$ is in the interior of the bound constraints, then every step from $x_k$ is in the interior. This can be seen by observing that the trust region constraint

$$\|D_k^{-1}d\|_2^2 = \sum_{i=1}^{n}\left(\frac{d^i}{D_k^{ii}}\right)^2 < 1$$

implies that

$$|d^i| < |D_k^{ii}| \quad \text{for all i.}$$

Suppose the closest bound to $x_k^i$ is $l^i$, so that $D_k^{ii} = x_k^i - l^i$. Then $|d^i| < x_k^i - l^i$, and no matter what the sign of $d^i$ is, the inequality $x_k^i + d^i > l^i$ holds; that is, the step does not violate this bound. The step also cannot violate the upper bound, because it is at least as far away from $x_k^i$ as $l^i$. A similar argument applies if the closest bound to $x_k^i$ is $u^i$. Thus, every component of $x_k^i + d^i$ lies within its bounded interval, and therefore the full step $x_k + d$ must be in the interior of the bound constraints.

The second idea borrowed from linear programming interior point methods is the addition of the logarithmic barrier term to move iterates away from the bounds. This is needed because the subproblem (2.2.10) becomes increasingly ill-conditioned as $x_k$ approaches a bound constraint (both $\hat{A}_k$ and $\hat{W}_k$ become singular). The logarithmic barrier term for (BEP) is similar to the term used in linear programming, the difference being that it accounts for both lower and upper bound constraints. To define the notation, $L_f$ denotes a subset of the indices $\{1, \ldots, n\}$ corresponding to variables having finite lower bounds (i.e., $i \in L_f$ if and only if $l^i > -\infty$), and $U_f$ denotes a similar subset of indices for the upper bounds. The logarithmic barrier for problem (BEP) is

$$\phi(x) \equiv \sum_{i \in L_f} \ln(x^i - l^i) + \sum_{i \in U_f} \ln(u^i - x^i).$$

The trust region subproblem (2.2.10) minimizes a quadratic model of the Lagrangian, so a quadratic model of $\phi$ is also appropriate. This is different from linear programming, which uses a linear model of the barrier term; hence, the primal barrier algorithm does not reduce to Algorithm 2.2.1 for linear programming problems. Note that the Hessian of $\phi$ is diagonal and inexpensive to compute. Adding the model barrier terms and affinely scaling gives the basic trust region subproblem for (BEP)

$$\begin{array}{ll} \min\limits_{\hat{d} \in \mathbf{R}^n} & \hat{d}^T \hat{g}_k + \frac{1}{2}\hat{d}^T \hat{W}_k \hat{d} - \rho_k(\hat{d}^T \nabla\hat{\phi}_k + \frac{1}{2}\hat{d}^T \nabla^2 \hat{\phi}_k \hat{d}) \\ \text{subject to} & \hat{A}^T \hat{d} + \hat{c}_k = \emptyset \\ & \|\hat{d}\|_2 \le \Delta_k, \end{array} \tag{2.2.11}$$

where $\nabla\hat{\phi}_k = D_k \nabla_x \phi(x_k)$ and $\nabla^2\hat{\phi}_k = D_k \nabla_x^2 \phi(x_k) D_k$.

The new primal barrier trust region algorithm solves (2.2.11) for $\hat{d}_k$ using particular values of $\Delta_k$ and $\rho_k$. The candidate step is evaluated by a merit function and, if found acceptable, produces the next iterate as $x_{k+1} = x_k + D_k \hat{d}_k$. Details are provided in the next section. Remember that in linear programming the parameters $\Delta_k$ and $\rho_k$ are adjusted according to special rules that ensure convergence at a specific linear rate. In the primal barrier algorithm $\Delta_k$ is chosen by trust region criteria, and $\rho_k$ is selected independently using heuristic guidelines given in §2.3.3.

Two essential ideas of the new algorithm are the use of a scaled ellipsoidal trust region constraint for better convergence, and a logarithmic barrier term to improve performance. Motivated from a completely different principle, Coleman, Li and Liu derived algorithms with similar properties for solving nonlinear problems with only bound constraints [52] and with bounds plus linear equality constraints [14]. Their algorithms use $D_k^{1/2}$ instead of $D_k$ for the ellipsoidal trust region, and they employ entirely different techniques to keep iterates away from the bound constraints. Thus, their algorithms differ in fundamental ways from the primal barrier method proposed here; furthermore, their method has no obvious technique for coping with nonlinear constraints.

## 2.3 Implementation Using the Byrd-Omojokun Algorithm

The primal barrier algorithm requires iterative solution of the trust region subproblem (2.2.11) and evaluation of trial steps using a merit function. The Byrd-Omojokun algorithm presented in §1.3.3 provides an efficient way of doing these tasks, and the ETR software implementation described in §1.5 is easily extended to accommodate (2.2.11). A general outline of the new algorithm is presented in Algorithm 2.3.1.

The main computational cost of Algorithm 2.3.1 is solving subproblem (2.2.11) for $\hat{d}_k$ by the method of Byrd and Omojokun. This method proceeds exactly as described in the first chapter, solving a vertical and horizontal subproblem at each iteration, but the performance and robustness of the method are affected by the scaling of (2.2.11). Recall that when second derivatives are used, the horizontal step is computed in ETR by conjugate gradient. Hence, if the subproblem becomes ill-conditioned, which certainly occurs as $x_k$ approaches bound constraints, then it takes a large number of inner CG iterations to compute $\hat{d}_k$. Scaling also means that the orthogonality of the vertical and horizontal components of $\hat{d}_k$ does not carry over to $d_k$ in the original coordinates. Very close to a bound the components could be almost parallel, conceivably cancelling each other out. Thus, the performance of the primal barrier algorithm is expected to degrade near bound constraints. It is more appropriately useful for reaching the local neighborhood of a solution to (BEP) and for guessing the active set of bound constraints. Then the exact solution to (BEP) should be found by some other technique, such as the active set extension of the Byrd-Omojokun algorithm mentioned at the beginning of this chapter.

A few parts of Algorithm 2.3.1 need to be elaborated more completely. The next sections address the issues of a proper stop test, backtracking along a step that leaves the interior, and choosing the barrier weight $\rho_k$.

**Algorithm 2.3.1** *General description of the primal barrier algorithm*

> Constants $\epsilon > 0$ and $\eta, \zeta, \gamma \in (0,1)$ are given
> Choose $\Delta_0 > 0$ and $\rho_0 \geq 0$
> Start with $x_0$ in the interior of all bound constraints
> LOOP, starting with $k = 0$
>   IF $x_k$ satisfies first-order conditions for (BEP) to within $\epsilon$ THEN STOP
>   Calculate the scaling matrix $D_k$ using (2.2.8)
>   Compute $c_k$, $g_k$, $A_k$ and $W_k$ in original problem coordinates
>   Transform using (2.2.9) to get $\hat{g}_k, \hat{A}_k$ and $\hat{W}_k$
>   Solve the trust region subproblem (2.2.11) for $\hat{d}_k$
>   using the Byrd-Omojokun method of vertical and horizontal subproblems
>   IF $x_k + D_k\hat{d}_k$ violates a bound constraint
>     THEN Find $\tau \in (0,1)$ such that $x_k + \tau D_k\hat{d}_k$ is in the interior
>       of all bound constraints, and set $\hat{d}_k \leftarrow \tau\hat{d}_k$
>   Compute the actual reduction in the merit function, a_red,
>   and the predicted reduction p_red at $x_k + D_k\hat{d}_k$
>   IF $\frac{\text{a\_red}}{\text{p\_red}} \geq \eta$
>     THEN Set $x_{k+1} = x_k + D_k\hat{d}_k$, $\Delta_{k+1} \geq \Delta_k$, and choose a new $\rho_{k+1}$
>     ELSE Set $x_{k+1} = x_k$, $\Delta_{k+1} \leq \gamma\|d_k\|_2$, and $\rho_{k+1} = \rho_k$
> CONTINUE, after incrementing $k$

### 2.3.1 Stopping Criteria

As mentioned, the primal barrier algorithm cannot hope to solve (BEP) exactly if bounds are active at the solution. Instead, it attempts to identify the active set of bound constraints and to satisfy the first-order KKT conditions to a finite tolerance. If $x_*$ is a local solution of (BEP) and $A_*$ is full rank, then $x_*$ must satisfy the first-order conditions (see [34] or [39])

$$\left.\begin{array}{r}(g_* - A_*\lambda)^i - \sigma_L^i + \sigma_U^i = 0 \\ \sigma_L^i(x_*^i - l^i) = 0 \quad \text{and} \quad \sigma_U^i(u^i - x_*^i) = 0 \\ \sigma_L^i \geq 0 \quad \text{and} \quad \sigma_U^i \geq 0\end{array}\right\} \quad \text{for } i = 1, \ldots, n$$

$$c_* = \emptyset,$$

where $\lambda \in \mathbf{R}^m, \sigma_L \in \mathbf{R}^n$ and $\sigma_U \in \mathbf{R}^n$ are Lagrange multiplier vectors, and where

$$(g_* - A_*\lambda)^i \equiv g_*^i - \sum_{j=1}^{m} \lambda^j \frac{\partial c_*^j}{\partial x^i}.$$

The active set $\mathcal{A}_*$ is defined as the subset of indices $\{1, \ldots, n\}$ corresponding to the components of $x_*$ that are at an active bound; i.e., $\mathcal{A}_* = \{i : x_*^i = l^i \text{ OR } x_*^i = u^i\}$. Notice that for $i \notin \mathcal{A}_*$, the multipliers $\sigma_L^i$ and $\sigma_U^i$ are zero.

¿From the KKT conditions we can derive formulas for the multipliers $\lambda$, $\sigma_L$ and $\sigma_U$ as follows. Reorder the components of $x_*$ so that those active at a lower bound are first, those active at an upper bound are last, and the components with no active bound are in between. Partition this vector and denote it by $x_* = [x_*^L \;\; x_*^I \;\; x_*^U]^T$, where the superscripts L , I and U stand for "lower", "interior" and "upper" bounded components, respectively. Now apply this same ordering and superscript convention to $g_*$, $A_*$, $\sigma_L$ and $\sigma_U$. The complementarity conditions imply that $\sigma_L^I = \sigma_L^U = \emptyset$ and $\sigma_U^L = \sigma_U^I = \emptyset$, and then the first KKT equation becomes

$$\begin{bmatrix} A_*^L \\ A_*^I \\ A_*^U \end{bmatrix} \lambda = \begin{bmatrix} g_*^L \\ g_*^I \\ g_*^U \end{bmatrix} - \begin{bmatrix} \sigma_L^L \\ \emptyset \\ \emptyset \end{bmatrix} + \begin{bmatrix} \emptyset \\ \emptyset \\ \sigma_U^U \end{bmatrix}.$$

Thus, assuming $A_*^I$ is full rank, the nonzero multipliers are given by

$$\lambda = [(A_*^I)^T A_*^I]^{-1} (A_*^I)^T g_*^I \tag{2.3.1}$$
$$\sigma_L^L = g_*^L - A_*^L \lambda \tag{2.3.2}$$
$$\sigma_U^U = -(g_*^U - A_*^U \lambda). \tag{2.3.3}$$

The stop test used in the primal barrier algorithm starts by guessing the active set of bound constraints as $\mathcal{A}_{k,0} = \{i : x_k^i - l^i < \epsilon \text{ OR } u^i - x_k^i < \epsilon\}$. This determines the index sets L , I and U that define $x_k = [x_k^L \;\; x_k^I \;\; x_k^U]^T$. Next, an estimate of $\lambda$ is computed using (2.3.1), and the vector $g_k - A_k\lambda$ is calculated. A check is made on all the components of this vector belonging to L and U to see if $\sigma_L^L$ and $\sigma_U^U$ have the correct sign in equations (2.3.2)-(2.3.3). If a sign error occurs, it means that one or more of the bounds in $\mathcal{A}_{k,0}$ do not belong there. The bound with the greatest sign violation is removed, giving a new active set guess $\mathcal{A}_{k,1}$. Now $\lambda$ is recomputed and the signs of $g_k - A_k\lambda$ checked again. This procedure repeats until an active set $\mathcal{A}_{k,j}$ is found that has no sign errors (or until the active set becomes empty). Finally, the stop test requires that $\|c_k\|_\infty < \epsilon$ and $\|g_k^I - A_k^I\lambda\|_\infty < \epsilon$ both hold.

This stop test is not entirely rigorous or foolproof. It avoids enforcing a tolerance on either of the KKT conditions $\sigma_L^i(x_k^i - l^i) = 0$ or $\sigma_U^i(u^i - x_k^i) = 0$. It does not try to add a previously dropped bound to $\mathcal{A}_{k,j}$ when $j > 1$, and so it has no guarantee of finding an active set with no sign errors. The present stop test is only preliminary, but it is effective in stopping the prototype primal barrier algorithm near a local solution with a good guess of the final active set $\mathcal{A}_*$.

### 2.3.2 Backtracking into the Interior

The trust region size is allowed to become larger than one in the primal barrier algorithm because longer steps can solve a problem faster; however, this allows a trial step to violate bound constraints. In this event, the algorithm backtracks along the step to a point in the interior of the bound constraints.

If the trial iterate is at $x_k + d_k$, then the smallest quantity among

$$\min_{i \in L_f,\, x_k^i + d_k^i \le l^i} \left\{ \frac{x_k^i - l^i}{|d_k^i|} \right\} \quad \text{and} \quad \min_{i \in U_f,\, x_k^i + d_k^i \ge u^i} \left\{ \frac{u^i - x_k^i}{|d_k^i|} \right\}$$

determines a number $\bar{\tau}$. If $x_k + d_k$ violates a bound constraint, then $\bar{\tau} < 1$ and $x_k + \bar{\tau} d_k$ is a point exactly on the most violated bound and in the interior of all other bounds. To give a trial point strictly in the interior, $\tau = 0.99\bar{\tau}$ is used. The trial step is modified to $\hat{d}_k \leftarrow \tau \hat{d}_k$ and tested as a candidate trust region subproblem solution. This operation superimposes the bound constraints of (BEP) onto the ellipsoidal trust region constraint of subproblem (2.2.11) in a manner that does not interfere with the global convergence properties of the trust region method. At most one backtrack per subproblem is made.

Not surprisingly, the backtracked step $x_k + \tau d_k$ is almost always accepted as a trust region subproblem solution, even when $\tau$ is fairly small. But it is not a good idea to allow trial steps to violate the bounds by a large amount, because the step is essentially ignoring the bound constraints. Hence, the rate at which the trust region may increase is limited to $1.2\,\|\hat{d}_k\|_2$ for subsequent iterations after a step is backtracked.

In [52] it is suggested that using $\tau = (1 - \bar{\tau}\|d_k\|)\bar{\tau}$ instead of $\tau = 0.99\bar{\tau}$ might increase the asymptotic convergence rate of the algorithm. This option has yet to be explored.

### 2.3.3 Heuristic for Choosing $\rho_k$

A reasonable principle for selecting the barrier term weight $\rho_k$ is to pick a value such that the solution to the modified barrier problem

$$\begin{array}{ll} \displaystyle\min_{x \in \mathbf{R}^n} & f(x) - \rho\phi(x) \\ \text{subject to} & c^i(x) = 0, \qquad \text{for } i = 1, \ldots, m \end{array} \tag{2.3.4}$$

is as close as possible to the current iterate $x_k$. For instance, $\rho_k$ could be chosen to minimize the $\ell_2$ norm of the gradient of the Lagrangian for this problem, thus attempting to satisfy the first-order necessary KKT conditions at $x_k$ [29]. But a simple counterexample due to Powell [65] reveals a weakness in this idea. Consider the one-dimensional problem

$$\min_x \; -\frac{1}{1+x^2} \qquad \text{subject to } \; x \geq 1, \tag{2.3.5}$$

which has the local solution $x_* = 1$. The modified barrier problem is

$$\min_x \; -\frac{1}{1+x^2} - \rho \ln(x-1), \tag{2.3.6}$$

which has a minimum at a point slightly larger than one, but is also unbounded from below as $x \to \infty$. The gradient of (2.3.6) is exactly zero when

$$\rho = \frac{2x}{(1+x^2)^2}(x-1). \tag{2.3.7}$$

Now if $x_k$ is the current iterate and $\rho_k$ is computed by (2.3.7), then (2.3.6) is immediately at a stationary point; however, this point is a local *maximum* of (2.3.6), not a minimum. This mistake occurs because the criteria for choosing $\rho_k$ uses only first-order information.

The first-order criteria suggested in [29] still has enough utility that it can be adapted for the primal barrier algorithm. Observe that if $\rho_k$ is chosen smaller than the value computed by (2.3.7), then the local maximum of (2.3.6) is pushed to the right of $x_k$. Iterative solution of (2.3.6) with small $\rho_k$ should therefore end at a point near the bound constraint $x \geq 1$. If $\rho_k$ is chosen larger, then the maximum is pushed to the left of $x_k$ and iterates head towards the unbounded minimizer at infinity. As a numerical illustration, consider the point $x_k = 5$. The choice of $\rho_k = 0.05917$ satisfies (2.3.7), and with this value of $\rho_k$ (2.3.6) has a local minimum at $x = 1.1367$ and a local maximum at $x = 5$. Choosing the smaller value $\rho_k = 0.01$ changes problem (2.3.6), moving the minimum to $x = 1.0204$ and the maximum to $x = 13.5358$.

The principle of adjusting $\rho_k$ to shift solutions of the modified barrier subproblem (2.3.4) is generalized for the primal barrier algorithm. Experiments show that it takes many iterations to find a point close to the solution of this subproblem, assuming $\rho_k$ is small enough that there is a solution. Therefore, a two-stage strategy is employed. At the start of the algorithm, if $x_k$ is close to or in violation of a bound constraint, then $\rho_k$ is chosen very large to push iterates away from the bounds. In terms of the example above, $\rho_k$ is chosen so that iterates head towards the unbounded minimizer at infinity. In the second stage, a fairly small value of $\rho_k$ is selected to favor a local solution near the active set of bounds.

The "large" and "small" values of the weight attempt to satisfy the largest and smallest components of the gradient of the Lagrangian for (2.3.4):

$$\rho_{sm} \equiv \min_i \begin{cases} (g_k - A_k\lambda)^i(x^i - l^i) & \text{IF } i \in L_f \text{ AND } (g_k - A_k\lambda)^i > 0 \\ -(g_k - A_k\lambda)^i(u^i - x^i) & \text{IF } i \in U_f \text{ AND } (g_k - A_k\lambda)^i < 0 \end{cases} \tag{2.3.8}$$

$$\rho_{lg} \equiv \max_i \begin{cases} (g_k - A_k\lambda)^i(x^i - l^i) & \text{IF } x^i - l^i \leq u^i - x^i \text{ AND } (g_k - A_k\lambda)^i > 0 \\ -(g_k - A_k\lambda)^i(u^i - x^i) & \text{IF } x^i - l^i > u^i - x^i \text{ AND } (g_k - A_k\lambda)^i < 0. \end{cases} \tag{2.3.9}$$

Only components that appear to have an active bound are considered in (2.3.8)-(2.3.9), and the multipliers $\lambda$ are computed from the least squares equation $g_k = A_k\lambda$. Algorithm 2.3.2 gives the heuristic rules for deciding whether to use the small or large value of the weight. This algorithm is called when Algorithm 2.3.1 computes $\rho_k$. It uses an external variable that is initialized in Algorithm 2.3.1.

Algorithm 2.3.2 is an ad-hoc collection of rules designed to move iterates away from the bounds (but not all the way to a "central path"), and then to allow convergence to a local solution. It differs greatly in philosophy and detail from the rules for choosing the barrier weight in linear programming. Notice that it allows $\rho_k$ to increase or decrease every iteration during the first stage, but quickly decreases it to zero in the second stage. The heuristic works well for the problems tested in §2.4, but no claim is made concerning its robustness or generality for other problems.

### 2.3.4 Finding a Start Point in the Interior

Many real problems do not have a start point that is in the interior of all bound constraints. Often the start point $x_0$ lies on a bound, and sometimes it is plainly in violation. In a manner similar to the "big M" method of linear programming [1], a modified version of problem (BEP) can be used by the primal barrier algorithm to find a point in the interior. This point may then be used as an acceptable start point for the original problem.

**Algorithm 2.3.2** *Heuristic for Computing $\rho_k$ in Algorithm 2.3.1*

(Initialize stage $= 1$ at the start of Algorithm 2.3.1)
IF closest bound $\geq 10^{-1}$ OR $\|c_k\|_\infty < 10^{-1}$ OR $\|\hat{g}_k - \hat{A}_k\hat{\lambda}_k\|_\infty < 10^{-2}$
  THEN
    stage $= 2$
    Compute $\rho_{sm}$ using (2.3.8)
    $\rho_k = \rho_{sm}/10$
    IF $\rho_k < 10^{-6}$ THEN $\rho_k = 0$
  ELSE IF stage $= 2$ THEN
    $\rho_k = \rho_{k-1}/10$
    IF $\rho_k < 10^{-6}$ THEN $\rho_k = 0$
  ELSE
    Compute $\rho_{lg}$ using (2.3.9)
    $\rho_k = \min\{100\rho_{lg}, 10^6\}$

The modification adds one artificial variable, called $s$. It is added to every violated bound constraint, then initialized to make the point $[x_0 \quad s_0]^T$ be in the interior of the modified problem. The primal barrier algorithm finds iterates for this modified problem using the logarithmic barrier with a large weight plus an additional penalty term for $s$. The extra terms together combine to drive iterates into the interior of the original bound constraints in an effective manner.

To describe the modified problem, two sets of indices need to be defined. Let $L_v \equiv \{i \in L_f : x_0^i - l^i < 10^{-3}\}$ be the components of the start point that are not in the interior of a lower bound constraint, and let $U_v \equiv \{i \in U_f : u^i - x_0^i < 10^{-3}\}$ be a similar set of indices not in the interior of an upper bound. Note the use of a small tolerance to make sure the start point is well into the interior. The modified form of (BEP) is

$$\begin{array}{llcll} \min\limits_{x,s} & f(x) + Ms & & & \\ \text{subject to} & c^i(x) & = & 0, & \text{for } i = 1,\ldots,m \\ & x^i - l^i + s & \geq & 0, & \text{for } i \in L_v \\ & x^i - l^i & \geq & 0, & \text{for } i \in L_f \text{ but } i \notin L_v \\ & u^i - x^i + s & \geq & 0, & \text{for } i \in U_v \\ & u^i - x^i & \geq & 0, & \text{for } i \in U_f \text{ but } i \notin U_v. \end{array}$$

Choosing $s_0$ sufficiently positive allows the bound constraints of the modified problem to be strictly satisfied at $[x_0 \;\; s_0]^T$. The precise value used is

$$s_0 = \max\left\{ 2\,|\text{largest violation}|,\ \min\{1, \frac{1}{2}(\text{smallest bound interval})\ \}\right\}.$$

Usually $s_0$ is set to twice the magnitude of the largest bound constraint violation at $x_0$ in the original problem. However, the largest violation could be zero if $x_0$ lies exactly on the bounds. In this case, $s_0$ is set arbitrarily to one, or to half the distance between the smallest interval $u^i - l^i$ taken over all bounds (an example with bounds interval less than one is test problem TORSION1).

The logarithmic barrier corresponding to the modified problem is

$$\begin{aligned}\phi(x,s) \equiv \sum_{i\in L_v} \ln(x^i - l^i + s) + \sum_{i\in L_f, i\notin L_v} \ln(x^i - l^i) \\ + \sum_{i\in U_v} \ln(u^i - x^i + s) + \sum_{i\in U_f, i\notin U_v} \ln(u^i - x^i).\end{aligned}$$

This is always included in the trust region subproblem (2.2.11) using the weight $\rho_{lg}$ computed by (2.3.9).

The penalty weight $M$ is chosen at each iteration to be just large enough to prevent $s$ from increasing. In linear programming $M$ is often chosen to reduce $s$, but experiments show this makes the modified nonlinear problem ill-conditioned. Instead, $M$ is used to keep $s$ approximately the same, and the logarithmic barrier term then pushes components of $x$ away from the modified problem bounds.

The solution of the modified problem for given $\rho_k$ and $M$ must satisfy the first-order KKT conditions, and the particular condition corresponding to the partial of the Lagrangian with respect to $s$ gives a formula for computing $M$. This condition is

$$\begin{aligned} 0 &= \frac{\partial}{\partial s}[f(x) - \rho_k \phi(x,s) + Ms - \lambda^T c(x)] \\ &= M - \rho_k \left[ \sum_{i\in L_v} \frac{1}{x^i - l^i + s} + \sum_{i\in L_f, i\notin L_v} \frac{1}{x^i - l^i} + \sum_{i\in U_v} \frac{1}{u^i - x^i + s} + \sum_{i\in U_f, i\notin U_v} \frac{1}{u^i - x^i} \right]. \end{aligned}$$

Given $\rho_k$ and the current iterates $x_k$ and $s_k$, a value for $M$ can be defined at which the problem is already solved with respect to this one KKT condition. The actual value used for $M_k$ is made 10% larger to keep $s$ from increasing and to prevent an exact solution of the modified problem from being found. This forces iterations to continue until an $x_k$ in the interior of the original problem is generated.

## 2.4 Test Results

The primal barrier algorithm was tested on a set of difficult test problems taken from the CUTE collection [3]. Most of the problems were large, and a wide range of equality constraint situations was represented (many constraints, few constraints, linear and nonlinear). The problem set is summarized in Table 2.1, which includes the number of bound constraints (*bnds*), the number of active bounds at the solution ($|\mathcal{A}_*|$), and whether the start point is in the interior or not (*int* $x_0$). The number of active bounds was determined by solving the problem with the LANCELOT [17] algorithm using a stop tolerance of $10^{-6}$.

All problems were tested with a uniform choice of algorithm parameters, most of which were the same as the values used in ETR to obtain the results of §1.6. A notable exception

| *problem name* | $n$ | $m$ | *nature of* $c(x)$ | *bnds* | $\lvert\mathcal{A}_*\rvert$ | *int* $x_0$ |
|---|---|---|---|---|---|---|
| HS2 | 2 | 0 | - | 1 | 1 | no |
| LINVERSE1 | 199 | 0 | - | 100 | 58 | no |
| LINVERSE | 999 | 0 | - | 500 | 316 | no |
| BQPGAUSS | 2003 | 0 | - | 4006 | 94 | no |
| TORSION2 | 484 | 84 | linear | 800 | 144 | yes |
| GOULDQP2 | 699 | 349 | linear | 1398 | 123 | no |
| HAGER4 | 1001 | 501 | linear | 500 | 251 | yes |
| OBSTCLAE | 1024 | 124 | linear | 1800 | 384 | no |
| TORSION1 | 1024 | 124 | linear | 1800 | 312 | no |
| HS99 | 7 | 2 | nonlinear | 14 | 0 | yes |
| DIXCHLNV | 100 | 50 | nonlinear | 100 | 0 | yes |
| LEAKNET | 156 | 153 | nonlinear | 82 | 3 | no |
| OPTCNTRL | 122 | 83 | nonlinear | 120 | 39 | no |
| READING1 | 202 | 101 | nonlinear | 402 | 177 | no |
| ORTHREGF | 305 | 100 | nonlinear | 2 | 0 | yes |

Table 2.1: Description of Test Problem Set

was the increase in the trust region size following an accepted step. The rules of §1.5.8 were used, but the maximum trust region size increase was only $5\|\hat{d}_k\|_2$ at the start of each problem, and was then reduced to $1.2\,\|\hat{d}_k\|_2$ if an iterate left the interior (see §2.3.2). All tests used second derivatives and the dogleg option for solving the vertical subproblem.

Every problem except READING1 was solved to the extent defined by the stop conditions of §2.3.1. The correct active set was not always identified, but it appears that the trust region method is robust. The lone exception, READING1, failed because an iterate was reached at which the constraint Jacobian $A_k$ was singular.

Problems with equality constraints tended to take fewer outer loops of the algorithm if the vertical step was allowed to be larger. In ETR the size is controlled by adjusting the parameter $\zeta$, since $\|v_k\|_2 \leq \zeta\Delta_k$. The test results reported in this section used $\zeta = 0.85$, compared to the value 0.8 used in the ETR tests of the previous chapter.

The "big M" method described in §2.3.4 worked well for all the problems with a bad start point $x_0$. It usually found a point in the interior of the bound constraints in two or three iterations. During some early testing, a linear model of the logarithmic barrier term was used instead of the quadratic model (making Algorithm 2.3.1 consistent with Algorithm 2.2.1), and it was discovered that an interior point could not be found for problem READING1 except for extremely large values of $\rho_k$. This is another argument for using the quadratic model in (2.2.11).

As anticipated, the computational work of solving the trust region subproblem (2.2.11) increased as bounds were approached. Table 2.2 shows the magnitude of this phenomenon for problem OBSTCLAE, which has 384 bounds active at its solution. For even-numbered iterations $k$, the table lists the distance from the current iterate $x_k$ to the nearest bound (*closest bnd*), the number of bounds that are within $10^{-3}$ of $x_k$ ($num < \epsilon$), and the number of CG inner iterations required to solve the horizontal subproblem (*CG iters*). A stop tolerance of $\epsilon = 10^{-3}$ was used to obtain these results. Recall that the horizontal subproblem is not preconditioned because it operates on the reduced Hessian $Z_k^T W_k Z_k$.

| $k$ | *closest bnd* | $num < \epsilon$ | *CG iters* |
|---|---|---|---|
| 2 | $4.3 \times 10^{-3}$ | 0 | 31 |
| 4 | $1.1 \times 10^{-4}$ | 2 | 15 |
| 6 | $1.1 \times 10^{-4}$ | 5 | 11 |
| 8 | $1.1 \times 10^{-4}$ | 12 | 12 |
| 10 | $1.0 \times 10^{-4}$ | 16 | 28 |
| 12 | $6.5 \times 10^{-5}$ | 94 | 34 |
| 14 | $2.7 \times 10^{-5}$ | 363 | 71 |
| 16 | $2.6 \times 10^{-5}$ | 398 | 116 |
| 18 | $3.2 \times 10^{-6}$ | 393 | 383 |
| 20 | $7.1 \times 10^{-7}$ | 390 | 550 |
| 22 | $2.8 \times 10^{-7}$ | 387 | 543 |
| 24 | $2.4 \times 10^{-8}$ | 387 | 1441 |

Table 2.2: Computational Cost of Approaching Bound Constraints

Comprehensive results for all test problems are recorded in Table 2.3. The number of active bound constraints at the exact solution ($|\mathcal{A}_*|$) are compared with the number of guessed active bounds ($|\mathcal{A}_\epsilon|$), and the number of outer iterations made by Algorithm 2.3.1 (*outer iters*) are listed. The stop tolerance was $\epsilon = 10^{-3}$. The number of function and gradient evaluations required to solve each problem is not reported, but both are roughly the same as the number of outer iterations.

Table 2.3 indicates that the active set of constraints was accurately determined in most cases with a tolerance of only $\epsilon = 10^{-3}$. For example, the guessed active set for problems BQPGAUSS and OPTCNTRL agreed exactly with $\mathcal{A}_*$, and for HAGER4 the 244 bounds guessed active were all part of $\mathcal{A}_*$. Since the stop conditions also require $|(g_k - A_k\lambda)^i| < \epsilon$ for every variable not at a bound, correct selection of the active set implies that the point found by Algorithm 2.3.1 was close to the exact solution. Hence, the algorithm was successful in approximately solving nearly all of the test problems.

Problems LINVERSE1 and LINVERSE (the same problem with different numbers of unknowns) were solved by the primal barrier algorithm, but only after an excessive number of iterations. The steps converged to a solution, but at a very slow linear rate. One novel feature of these problems is that more than half the unknowns are at their lower bound at the solution, while all variables start slightly infeasible with respect to their lower bound constraint; thus, moving into the interior of the problem also moves away from the solution. Nevertheless, the poor performance of the algorithm on these two problems is not fully

| *problem name* | $n$ | $m$ | $\|\mathcal{A}_*\|$ | $\|\mathcal{A}_\epsilon\|$ | *outer iters* |
|---|---|---|---|---|---|
| HS2 | 2 | 0 | 1 | 1 | 9 |
| LINVERSE1 | 199 | 0 | 58 | 60 | 95 |
| LINVERSE | 999 | 0 | 316 | 276 | 299 |
| BQPGAUSS | 2003 | 0 | 94 | 94 | 34 |
| TORSION2 | 484 | 84 | 144 | 152 | 12 |
| GOULDQP2 | 699 | 349 | 123 | 1 | 6 |
| HAGER4 | 1001 | 501 | 251 | 244 | 11 |
| OBSTCLAE | 1024 | 124 | 384 | 385 | 25 |
| TORSION1 | 1024 | 124 | 312 | 340 | 12 |
| HS99 | 7 | 2 | 0 | 0 | 4 |
| DIXCHLNV | 100 | 50 | 0 | 0 | 18 |
| LEAKNET | 156 | 153 | 3 | 1 | 6 |
| OPTCNTRL | 122 | 83 | 39 | 39 | 30 |
| READING1 | 202 | 101 | 177 | could not solve | |
| ORTHREGF | 305 | 100 | 0 | 0 | 28 |

Table 2.3: Performance of the Primal Barrier Algorithm

understood.

Problems TORSION2, GOULDQP2 and TORSION1 were solved with a small number of iterations, but relatively bad estimates of the number of active bounds were made. In all three of these problems most of the unknowns have both upper and lower bound constraints, and the difference $u^i - l^i$ is small compared to one. For TORSION1, the difference is as small as $6.45 \times 10^{-2}$, for TORSION2 it is $4.76 \times 10^{-2}$, and for GOULDQP2 the difference is only $4.02 \times 10^{-3}$. The small bounds gap implies that the stop tolerance of $\epsilon = 10^{-3}$ used in Table 2.3 is too large for these problems. Indeed, when the tolerance was reduced to $10^{-4}$, then the algorithm found the correct active set for problems TORSION1 and TORSION2. However, this tolerance was still not small enough for GOULDQP2, and reducing $\epsilon$ to $10^{-5}$ allowed the scaling transformation to become too ill-conditioned for robust performance. In short, the algorithm failed to solve problem GOULDQP2 to a reasonable degree of accuracy.

It is premature to compare the performance of the primal barrier algorithm to that of LANCELOT or other commercial software products. Much work remains to improve the algorithm, as outlined in the next section. But it is instructive to note that the number of outer iterations was significantly less for Algorithm 2.3.1 than for LANCELOT on problems having nonlinear equality constraints, especially problems HS99, DIXCHLNV, LEAKNET and OPTCNTRL. On the other hand, Algorithm 2.3.1 was much worse on problems with no equality constraints, particularly when there were many active bounds at the solution. Since LANCELOT uses an active set technique for dealing with bound constraints, these differences in performance are probably explained by the different approaches of the two algorithms.

## 2.5 Summary and Future Directions

This chapter presented a new algorithm for solving general nonlinear constrained problems. The algorithm is a trust region method that uses interior point techniques from linear programming to define a scaled trust region subproblem. Conceptually, the primal barrier algorithm is an extension of the Byrd-Omojokun method for equality constrained optimization that uses a changing ellipsoidal trust region constraint. A software implementation for large-scale problems was made using the ETR algorithm of the first chapter. Test results for various large problems show the algorithm to possess the robustness qualities of trust region methods, but to perform inefficiently near a solution at which many bound constraints are active. The algorithm seems uniquely promising for solving large-scale problems subject to nonlinear constraints. The work reported here is only preliminary, and further study is called for in the following areas:

- Asymptotic performance. Employing an active set method after probable active bounds are identified may relieve the ill-conditioning of subproblems near a solution. Eventually, a true active set extension of ETR should be used to precisely compute the final solution point. Implementation requires careful consideration of possible wrong guesses of active bounds by the primal barrier algorithm.

- Choice of the barrier weight $\rho_k$. Investigation of possible choices for nonlinear convex problems might lend insight and suggest a better heuristic than that of §2.3.3. It might be possible to base the choice on a convex model of the problem.

- Solving without second derivatives. Many problems do not have second derivative information available. ETR deals with this by using a limited memory BFGS approximation of the full Hessian $W_k$. Tests need to be conducted to evaluate this option in the primal barrier algorithm.

- Nonconvex inequality constraints. Suppose an inequality restricts entry through a closed bounded region, yet the region needs to be crossed in order to reach a solution. The shortest path might be directly through the region, but an interior point method must find a path that goes around it without coming too close to its boundary. This is an inherently difficult problem for interior methods that has not yet been addressed in the primal barrier algorithm.

# Appendix A

# Mathematical Fundamentals

## A.1 Properties of Orthogonal Complements

**Theorem A.1.1** *Let $A$ be an $n \times m$ matrix with $n > m > 0$. If $A$ is full rank, then the $m \times m$ matrix $A^T A$ is nonsingular and positive definite.*

*Proof.* For all nonzero $x \in \mathbf{R}^m$, $x^T A^T A x = \|Ax\|_2 \geq 0$. Since $A$ is full rank, the vector $Ax$ is nonzero and the inequality is strict. The relation $x^T A^T A x > 0$ is the very definition of a matrix being positive definite. □

**Definition A.1.2** *Let $A$ be an $n \times m$ matrix with $n > m > 0$, and let $Z$ be an $n \times (n-m)$ matrix. If $A$ and $Z$ are both full rank and satisfy $A^T Z = \emptyset$, then $A$ and $Z$ are* **orthogonal complements**.

**Theorem A.1.3** *If $A$ and $Z$ are orthogonal complements, then the $n \times n$ matrix $[A \;\; Z]$ is nonsingular.*

*Proof.* The proof is by contradiction. Assume the matrix is singular and some nonzero $x \in \mathbf{R}^n$ exists such that

$$[A \;\; Z]x = [A \;\; Z] \begin{bmatrix} x^1 \\ x^2 \end{bmatrix} = \emptyset.$$

Then $A^T(Ax^1 + Zx^2) = A^T A x^1 = \emptyset$. By Theorem A.1.1, $A^T A$ is nonsingular, so $x^1 = \emptyset$. Similarly, $Z^T(Ax^1 + Zx^2) = Z^T Z x^2 = \emptyset$, and we conclude that $x^2 = \emptyset$. Therefore, $x$ is zero, contradicting the assumption. □

**Theorem A.1.4** *If $A$ and $Z$ are orthogonal complements, then every vector $x$ has unique projections $x^a \in \mathbf{R}^m$ and $x^z \in \mathbf{R}^{n-m}$ such that*

$$x = Ax^a + Zx^z. \tag{A.1.1}$$

*Proof.* Multiply (A.1.1) by $A^T$ and then by $(A^TA)^{-1}$ to obtain a formula for $x^a$. Theorem A.1.1 establishes the existence of the inverse of $A^TA$, so this procedure is well-defined. A formula for $x^z$ is obtained similarly, and we have

$$\begin{aligned} x^a &= (A^TA)^{-1}A^Tx \\ x^z &= (Z^TZ)^{-1}Z^Tx. \end{aligned}$$

By construction, this proves that projections always exist. To show they are unique, suppose to the contrary that some $\tilde{x}^a \neq x^a$ exists such that $x = A\tilde{x}^a + Zx^z$ and $x = Ax^a + Zx^z$. Subtraction gives $A(\tilde{x}^a - x^a) = \emptyset$, and since $A$ is full rank, $\tilde{x}^a - x^a = \emptyset$. This contradicts the assumption and proves that $x^a$ is unique. A similar argument is made for $x^z$. □

**Theorem A.1.5** *If $A$ and $Z$ are orthogonal complements, then*

$$A(A^TA)^{-1}A^T + Z(Z^TZ)^{-1}Z^T = I.$$

*Proof.* From Theorem A.1.3 we know that the square matrix $[A \;\; Z]$ is nonsingular. A left inverse is easily found, satisfying the identity

$$\begin{bmatrix} (A^TA)^{-1}A^T \\ (Z^TZ)^{-1}Z^T \end{bmatrix} [A \;\; Z] = \begin{bmatrix} I & \emptyset \\ \emptyset & I \end{bmatrix}.$$

For square matrices, left and right inverses are identical; therefore,

$$[A \;\; Z] \begin{bmatrix} (A^TA)^{-1}A^T \\ (Z^TZ)^{-1}Z^T \end{bmatrix} = I,$$

and multiplying the matrices out gives the desired formula. □

## A.2 Properties of the $\ell_2$ Matrix Norm

Let $A$ be an $n \times m$ matrix with $n > m > 0$. Let $\lambda_1^{A^TA} \leq \cdots \leq \lambda_m^{A^TA}$ be the ordered eigenvalues of the $m \times m$ matrix $A^TA$. The $\ell_2$ norm of $A$ is given by the formula [41]

$$\|A\|_2 = \sqrt{\max\{\lambda_i^{A^TA}\}} = \sqrt{\lambda_m^{A^TA}}. \tag{A.2.1}$$

Computing the $\ell_2$ norm of $A^TA$ in the same way gives

$$\|A^TA\|_2 = \sqrt{\max\{\lambda_i^{(A^TA)^TA^TA}\}} = \sqrt{\max\{\lambda_i^{(A^TA)^2}\}} = \lambda_m^{A^TA}.$$

Therefore,

$$\|A^TA\|_2 = \|A\|_2^2. \tag{A.2.2}$$

If $A$ is full rank, then the $n \times n$ matrix $AA^T$ is of rank $m$ and $n-m$ of its eigenvalues are zero. The other $m$ eigenvalues are the same as the eigenvalues of $A^TA$, because the identity $A^TA\nu_i = \lambda_i^{A^TA}\nu_i$ implies that $AA^T(A\nu_i) = \lambda_i^{A^TA}(A\nu_i)$. Hence, $\max\{\lambda_i^{AA^T}\} = \max\{\lambda_i^{A^TA}\}$, and by formula (A.2.1),

$$\|AA^T\|_2 = \|A^TA\|_2. \tag{A.2.3}$$

## A.3 Vector Form of Taylor Series

Let $f(x)$ be a scalar function of the variables $x_1, \ldots, x_n$. If $f$ is continuous in every variable so that all its partial derivatives exist, then the fundamental theorem of calculus holds. The following three equations are equivalent ways of writing the fundamental theorem, for arbitrary vectors $x$ and $y$:

$$\begin{aligned} f(y) - f(x) &= \int_x^y df \\ &= \int_{x_1}^{y_1} \cdots \int_{x_n}^{y_n} \frac{\partial f}{\partial x_1} dx_1 + \ldots + \frac{\partial f}{\partial x_n} dx_n \\ &= \int_x^y \nabla f(z)^T dz, \quad \text{for} \quad z \in \mathbf{R}^n. \end{aligned}$$

In the last equation, let us parameterize the vector variable of integration in terms of a scalar variable $t$, so that $z(t) = x + t(y - x)$. Then the fundamental theorem becomes

$$f(y) - f(x) = \int_0^1 (y - x)^T \nabla f(x + t(y - x))\, dt. \tag{A.3.1}$$

This is the one-term Taylor series with remainder for the function $f(y)$ expanded about the point $x$.

Now let $g(x) = [g_1(x) \cdots g_n(x)]^T$, where each component $g_i(x)$ is a scalar function of the variables $x_1, \ldots, x_n$. If every $g_i(x)$ is continuous, then equation (A.3.1) generalizes to

$$g(y) - g(x) = \int_0^1 J(x + t(y - x))(y - x)\, dt, \tag{A.3.2}$$

where

$$J(z) \equiv \begin{bmatrix} \nabla g_1(z)^T \\ \vdots \\ \nabla g_n(z)^T \end{bmatrix}.$$

If $g$ is the gradient of a scalar function $f(x)$, then $J$ is its Hessian $\nabla^2 f$, and we can construct the two-term Taylor series with remainder for $f(y)$ expanded about the point $x$. Start with (A.3.1) in the form

$$f(y) - f(x) - (y - x)^T \nabla f(x) = \int_0^1 (y - x)^T [\nabla f(x + t(y - x)) - \nabla f(x)]\, dt,$$

and apply (A.3.2) to the quantity in square brackets to obtain

$$\begin{aligned} &f(y) - f(x) - (y - x)^T \nabla f(x) = \\ &\quad \int_0^1 t \int_0^1 (y - x)^T [\nabla^2 f(x + ut(y - x))](y - x)\, du\, dt. \end{aligned} \tag{A.3.3}$$

## A.4 Powell's Theorem for Trust Region Subproblems

A theorem first established by Powell [66] quantifies the progress made in minimizing a trust region subproblem by taking the Cauchy step. The proof is repeated here and then generalized in the next theorem for subproblems having an ellipsoidal trust region constraint. The notation of these two theorems matches that of §1.4.

**Theorem A.4.1** *(Powell's Theorem) Consider the trust region subproblem*

$$\min_{\tilde{d}} \ \tilde{\phi}(\tilde{d}) = \tilde{g}^T\tilde{d} + \frac{1}{2}\tilde{d}^T\tilde{B}\tilde{d} \quad \text{subject to} \quad \|\tilde{d}\|_2 \le \Delta,$$

*where $\tilde{B}$ is symmetric. The reduction in $\tilde{\phi}$ made by the step $\tilde{d}_{cp}$ to the Cauchy point satisfies*

$$\tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) \ \ge \ \frac{1}{2}\|\tilde{g}\|_2 \ \min\{\frac{\|\tilde{g}\|_2}{\|\tilde{B}\|_2}, \Delta\}. \tag{A.4.1}$$

*Proof.* The Cauchy point for this subproblem is given by (1.4.17)-(1.4.18):

$$\tilde{d}_{cp} = -\tilde{\alpha}\tilde{g}, \quad \text{where} \quad \tilde{\alpha} = \begin{cases} \dfrac{\tilde{g}^T\tilde{g}}{\tilde{g}^T\tilde{B}\tilde{g}}, & \text{if } \tilde{g}^T\tilde{B}\tilde{g} > 0 \quad \text{and} \quad \dfrac{(\tilde{g}^T\tilde{g})^{3/2}}{\tilde{g}^T\tilde{B}\tilde{g}} \le \Delta \\ \dfrac{\Delta}{\|\tilde{g}\|_2}, & \text{otherwise.} \end{cases}$$

Thus,

$$\tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) = \tilde{\alpha}\tilde{g}^T\tilde{g} - \frac{1}{2}\tilde{\alpha}^2\tilde{g}^T\tilde{B}\tilde{g},$$

and the inequality holds trivially when $\tilde{g} = \emptyset$. There are three cases to consider when $\tilde{g}$ is nonzero, corresponding to the rules for choosing $\tilde{\alpha}$. First, suppose that $\tilde{\alpha}$ is given by the top formula. Then,

$$\tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) = \frac{\tilde{g}^T\tilde{g}}{\tilde{g}^T\tilde{B}\tilde{g}}\tilde{g}^T\tilde{g} - \frac{1}{2}\left(\frac{\tilde{g}^T\tilde{g}}{\tilde{g}^T\tilde{B}\tilde{g}}\right)^2 \tilde{g}^T\tilde{B}\tilde{g} = \frac{1}{2}\|\tilde{g}\|_2^2\frac{\tilde{g}^T\tilde{g}}{\tilde{g}^T\tilde{B}\tilde{g}}.$$

The triangle inequality states that $\tilde{g}^T\tilde{B}\tilde{g} \le \|\tilde{g}\|_2^2\|\tilde{B}\|_2$, and so the reduction in $\tilde{\phi}$ for this choice of $\tilde{\alpha}$ can be expressed as

$$\tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) \ge \frac{1}{2}\frac{\|\tilde{g}\|_2^2}{\|\tilde{B}\|_2}. \tag{A.4.2}$$

Next, consider the case when $\tilde{g}^T\tilde{B}\tilde{g} > 0$ but $(\tilde{g}^T\tilde{g})^{3/2}/\tilde{g}^T\tilde{B}\tilde{g} > \Delta$. The bottom formula for $\tilde{\alpha}$ gives

$$\begin{aligned} \tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) &= \Delta\|\tilde{g}\|_2 - \frac{1}{2}\frac{\Delta^2}{\tilde{g}^T\tilde{g}}\tilde{g}^T\tilde{B}\tilde{g} \\ &> \Delta\|\tilde{g}\|_2 - \frac{1}{2}\Delta^2\frac{\|\tilde{g}\|_2}{\Delta}. \end{aligned}$$

Hence, for this choice of $\tilde{\alpha}$

$$\tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) > \frac{1}{2}\|\tilde{g}\|_2\Delta. \tag{A.4.3}$$

The remaining possibility is that $\tilde{g}^T\tilde{B}\tilde{g} \leq 0$. In this case $\tilde{\alpha}$ is again given by the bottom formula, so that

$$\tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) = \Delta\|\tilde{g}\|_2 - \frac{1}{2}\frac{\Delta^2}{\tilde{g}^T\tilde{g}}\tilde{g}^T\tilde{B}\tilde{g}.$$

Now the second term is nonnegative, which means that

$$\tilde{\phi}(\emptyset) - \tilde{\phi}(\tilde{d}_{cp}) \geq \frac{1}{2}\|\tilde{g}\|_2\Delta. \tag{A.4.4}$$

The three possible inequalities (A.4.2), (A.4.3) and (A.4.4) combine into (A.4.1). □

**Theorem A.4.2** *Consider the trust region subproblem*

$$\min_d \ \phi(d) = g^Td + \frac{1}{2}d^TBd \quad \text{subject to} \quad \|Md\|_2 \leq \Delta,$$

*where $B$ is symmetric and $M$ nonsingular. The reduction in $\phi$ made by the step $d_{cp}$ to the Cauchy point satisfies*

$$\phi(\emptyset) - \phi(d_{cp}) \ \geq \ \frac{1}{2}\sqrt{g^T(M^TM)^{-1}g}\ \min\{\frac{\sqrt{g^T(M^TM)^{-1}g}}{\|(M^TM)^{-1/2}B(M^TM)^{-1/2}\|_2}, \Delta\}. \tag{A.4.5}$$

*Proof.* It was shown in §1.4 that the subproblem of this theorem is equivalent to the subproblem of Theorem A.4.1 under the transformations (1.4.3) and (1.4.7):

$$\tilde{d} = (M^TM)^{1/2}d, \quad \tilde{g} = (M^TM)^{-1/2}g, \quad \text{and} \quad \tilde{B} = (M^TM)^{-1/2}B\,(M^TM)^{-1/2}.$$

Substitution of these quantities into (A.4.1) gives (A.4.5). □

# Appendix B

# Selected Test Problems

Recall that $n$ is the number of unknowns, $m$ the number of equality constraints, $nnz(A)$ denotes the number of nonzeroes in the sparse constraint Jacobian, and $nnz(W)$ is the number of nonzeroes in the sparse Hessian of the Lagrangian. In this appendix, subscripts on a vector are used to indicate vector components.

**HS99LNP** $(n = 7,\ m = 2,\ nnz(A) = 14,\ nnz(W) = 28)$
A simplification of problem 99 from the collection of Hock and Schittkowski [49]. The original problem had bound constraints on the variables, none of which were active at the solution. The bounds were dropped to define HS99LNP.

$$\begin{aligned}
\min_x \quad & -(9000\cos x_7 + 9000\cos x_6 + 3750\cos x_5 + 3750\cos x_4 \\
& \qquad +3750\cos x_3 + 1250\cos x_2 + 1250\cos x_1)^2 \\
\text{subject to} \quad & 25(50\sin x_1 - 32) + 25(50\sin x_2 - 32) \\
& +50(75\sin x_3 - 32) + 50(75\sin x_4 - 32) \\
& +50(75\sin x_5 - 32) + 90(100\sin x_6 - 32) \\
& +90(100\sin x_7 - 32) = 10^3 \\
& 9187.5(50\sin x_1 - 32) + 8562.5(50\sin x_2 - 32) \\
& +15250(75\sin x_3 - 32) + 12750(75\sin x_4 - 32) \\
& +10250(75\sin x_5 - 32) + 12150(100\sin x_6 - 32) \\
& +4050(100\sin x_7 - 32) = 10^5
\end{aligned}$$

**HS100LNP** $(n = 7,\ m = 2,\ nnz(A) = 10,\ nnz(W) = 9)$
A simplification of Hock and Schittkowski problem 100 [49]. The original problem had four inequality constraints, two active at the solution and two inactive. The former were made into equality constraints and the latter were dropped to define HS100LNP.

$$\begin{aligned}
\min_x \quad & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 \\
& \qquad +7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \\
\text{subject to} \quad & 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 = 127 \\
& 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 = 0
\end{aligned}$$

**HS104LNP** ($n = 8,\ m = 4,\ nnz(A) = 13,\ nnz(W) = 16$)

A modification of Hock and Schittkowski problem number 104 [49]. Like HS100, nonlinear inequality constraints were either made into equality constraints or were dropped. The original problem also had bound constraints not active at the solution that served to keep all variables positive. These were dropped, but two quadratic penalty terms had to be added to the objective to force $x_2$ and $x_8$ to remain positive. Even with this modification, the trust region size in ETR cannot be allowed to grow too quickly, or iterates are generated which become negative.

$$\begin{aligned}\min_x \quad & 10 - x_1 - x_2 + 0.4(x_1/x_7)^{0.67} + 0.4(x_2/x_8)^{0.67} \\ & +3(x_2 - x_2^*)^2 + 10(x_8 - x_8^*)^2\end{aligned}$$

$$\text{subject to} \quad \left.\begin{array}{rcl} 0.1x_1 + b_0 x_5 x_7 & = & 1 \\ 0.1x_1 + 0.1x_2 + b_0 x_6 x_8 & = & 1 \\ 4(x_3/x_5) + 2(x_3^{b1}/x_5) + b_0 x_3^{b2} x_7 & = & 1 \\ 4(x_4/x_6) + 2(x_4^{b1}/x_6) + b_0 x_4^{b2} x_8 & = & 1 \end{array}\right\} \text{ where } \begin{array}{rcl} b_0 & = & 0.0588 \\ b_1 & = & -0.71 \\ b_2 & = & -1.3 \end{array}$$

**HS111LNP** ($n = 10,\ m = 3,\ nnz(A) = 14,\ nnz(W) = 55$)

A version of Hock and Schittkowski problem 111 [49] from which all bound constraints have been removed. None were active at the solution.

**GENHS28** ($n = p,\ m = p - 2,\ nnz(A) = 3m,\ nnz(W) = 2n - 1$)

A multi-dimensional extension of problem 28 of Hock and Schittkowski [49].

$$\min_x \quad \sum_{i=1}^{p-1} (x_i + x_{i+1})^2$$

$$\text{subject to} \quad x_i + 2x_{i+1} + 3x_{i+2} = 1, \quad \text{for} \quad i = 1, \ldots, p - 2.$$

**HAGER2** ($n = 2p + 1,\ m = p + 1,\ nnz(A) = 3p + 1,\ nnz(W) = 3p + 1$)

A nonlinear optimal control problem by Hager [46]. The unknowns consist of a state vector $x$ and control vector $u$, each discretized into $p$ points.

$$\min_{x,u} \quad \sum_{i=1}^{p} \frac{1}{6p}(x_{i-1}^2 + x_{i-1}x_i + x_i^2) + \frac{u_i^2}{4p}$$

$$\begin{array}{lrcll} \text{subject to} & (p - \frac{1}{4})x_i - (p + \frac{1}{4})x_{i-1} - u_i & = & 0, & \text{for} \quad i = 1, \ldots, p \\ & x_0 & = & 1 & \end{array}$$

**HAGER3** ($n = 2p + 1,\ m = p + 1,\ nnz(A) = 3p + 1,\ nnz(W) = 5p + 1$)

The constraints are identical to HAGER2, and the objective is

$$\min_{x,u} \quad \sum_{i=1}^{p} \frac{1}{8p}[\frac{5}{8}(x_{i-1}^2 + x_{i-1}x_i + x_i^2) + (x_{i-1} + x_i)u_i] + \frac{u_i^2}{4p}$$

**ORTHREGA, ORTHREGC** ($n = 2m + 5$, $nnz(A) = 7m$, $nnz(W) = 9m$)

Two orthogonal regression (total least squares) problems by Gulliksson [45]. The object is to find the closest ellipse to a set of $m$ data points $\{(a_1, b_1), \ldots, (a_m, b_m)\}$ in the plane. An unknown $(x_i, y_i)$ corresponds to each data point, and these are constrained to lie on the unknown ellipse, which is specified by five additional variables. ORTHREGC is the same as ORTHREGA, differing only in the starting point.

$$\begin{array}{rl} \displaystyle\min_{x,y,z} & \displaystyle\sum_{i=1}^{m} (x_i - a_i)^2 + (y_i - b_i)^2 \\ \text{subject to} & z_1 x_i^2 + 2z_2 x_i y_i + z_3 y_i^2 - 2z_4 x_i - 2z_5 y_i = 1, \quad \text{for} \quad i = 1, \ldots, m \end{array}$$

**ORTHREGD** family ($n = 2m + 3$, $nnz(A) = 5m$, $nnz(W) = 9m + 6$)

Another orthogonal regression problem [45]. In this case, a circle is computed to fit the set of $m$ data points (comments in the CUTE SIF file indicate that Gulliksson may have intended the circle to be a cardioid). The center of the circle is also considered valid for fitting data points to. Any point $(x_i, y_i)$ that satisfies its constraint by being at the center of the circle $(z_1, z_2)$ will have the gradient of its constraint equal to zero; thus, $A_*$ can be singular.

$$\begin{array}{rl} \displaystyle\min_{x,y,z} & \displaystyle\sum_{i=1}^{m} (x_i - a_i)^2 + (y_i - b_i)^2 \\ \text{subject to} & \left[(x_i - z_1)^2 + (y_i - z_2)^2\right]^2 \left[1 - (z_3^2 + 1)^2\right] = 0, \quad \text{for} \quad i = 1, \ldots, m \end{array}$$

The difference between ORTHREGD, ORTHRGDM and ORTHRGDS is the set of data points $(a_i, b_i)$. In all cases, the starting point of the optimization problem is defined by setting $(x_i, y_i) = (a_i, b_i)$. ORTHREGD defines the data points as

$$\left.\begin{array}{rcl} a_i &=& \rho_i \cos\theta_i \\ b_i &=& \rho_i \sin\theta_i \end{array}\right\} \quad \text{where} \quad \begin{array}{rcl} \rho_i &=& (1.7^2 + 1 + \cos\theta_i)[\frac{1}{5}\cos(237.1531\,\theta_i) + 1] \\ \theta_i &=& 2\pi(i - 1)/m. \end{array}$$

This scatters $(a_i, b_i)$ in a ring-like pattern around the point $(1, 0)$, which is the center of the optimal circle. In ORTHRGDM the width of the ring is decreased by using

$$\rho_i = (1.7^2 + 1 + \cos\theta_i)[\frac{1}{10}\cos(237.1531\,\theta_i) + 1],$$

thus preventing any $(x_i, y_i)$ from moving to the center of the circle. In ORTHRGDS certain points $(a_i, b_i)$ are set to $(1.1, 0.1)$, forcing the corresponding variables $(x_i, y_i)$ to converge to the center of the circle.

**OPTCNTRL** family ($n = 3p + 2$, $m = 2p + 3$, $nnz(A) = 7p + 3$, $nnz(W) = 3p$)

A nonlinear optimal control problem by Murtagh and Saunders [59, problem 5.11]. The unknowns consist of state vectors $x$ and $y$, and a control vector $u$, each discretized into $p$ points.

$$\min_{x,y,u} \quad \frac{1}{2}\sum_{i=0}^{p} x_i^2$$

$$\begin{array}{rrcll} \text{subject to} & x_{i+1} & = & x_i + 0.2y_i, & \text{for} \quad i = 0, \ldots, p-1 \\ & y_{i+1} & = & y_i - 0.01y_i^2 - 0.004x_i + 0.2u_i, & \text{for} \quad i = 0, \ldots, p-1 \\ & x_0 & = & 10 & \\ & y_0 & = & 0 & \\ & y_p & = & 0 & \end{array}$$

Bound constraints are active at the solution for all but one component of $u$. In problems OPTCTRL3 and OPTCTRL6 all bounds are dropped, and a quadratic penalty term is added to the objective to attract $u$ to the original solution point. The modified objective is

$$\min_{x,y,u} \quad \frac{1}{2}\sum_{i=0}^{p} x_i^2 + \rho\sum_{i=1}^{p-1}(u_i - 0.2)^2,$$

with $\rho = 10^3$ in OPTCTRL3 and $\rho = 10^6$ in OPTCTRL6. Higher values of $\rho$ cause the Hessian of the modified objective to be more ill-conditioned.

# Bibliography


[1] Adler, I., M.G.C. Resende, and G. Veiga. "An implementation of Karmarkar's algorithm for linear programming." *Math. Prog. (Ser. A)* 44 (1989): 297–335.

[2] Barnes, E.R. "A variation on Karmarkar's algorithm for solving linear programming problems." *Math. Prog.* 36 (1986): 174–182.

[3] Bongartz, I., A.R. Conn, N.I.M. Gould, and Ph.L. Toint. "CUTE: constrained and unconstrained testing environment." Research Report. IBM T.J. Watson Research Center, Yorktown Heights, NY, 1993.

[4] Bunch, J.R. "Partial pivoting strategies for symmetric matrices." *SIAM J. Numer. Anal.* 11 (1974): 521–528.

[5] Bunch, J.R., and L. Kaufman. "Some stable methods for calculating inertia and solving symmetric linear systems." *Math. Comp.* 31 (1977): 163–179.

[6] Bunch, J.R., and B.N. Parlett. "Direct methods for solving symmetric indefinite systems." *SIAM J. Numer. Anal.* 8 (1971): 639–655.

[7] Buys, J.D., and P.C. Haarhoff. "A new method for the optimization of a nonlinear function subject to nonlinear constraints." *Computing Journal.* 13 (1970): 178–184.

[8] Byrd, R.H. "Robust trust region methods for constrained optimization." Third SIAM Conference on Optimization. Houston, 20 May 1987.

[9] Byrd, R.H., J. Nocedal, and R.B. Schnabel. "Representations of quasi-Newton matrices and their use in limited memory methods." *Math. Prog. (Ser. A)* 63 (1994): 129–156.

[10] Byrd, R.H., and R.B. Schnabel. "Approximate solution of the trust region problem by minimization over two-dimensional subspaces." *Math. Prog. (Ser. A)* 40 (1988): 247–263.

[11] Byrd, R.H., R.B. Schnabel, and G.A. Schultz. "A trust region algorithm for nonlinearly constrained optimization." *SIAM J. Numer. Anal.* 24 (1987): 1152–1170.

[12] Carter, R.G. "On the global convergence of trust region algorithms using inexact gradient information." *SIAM J. Numer. Anal.* 28 (1991): 251–265.

[13] Celis, M.R., J.E. Dennis, and R.A. Tapia. "A trust region strategy for nonlinear equality constrained optimization." *Numerical Optimization 1984.* Ed. Paul T. Boggs, Richard H. Byrd, and Robert B. Schnabel. Philadelphia: SIAM, 1985. 71–82.

[14] Coleman, T.F., and J. Liu. "An interior Newton method for quadratic programming." Technical Report 93-1388. Dept. of Computer Science, Cornell University, 1993.

[15] Conn, A.R., N.I.M. Gould, and Ph. L. Toint. "Global convergence of a class of trust region algorithms for optimization with simple bounds." *SIAM J. Numer. Anal.* 25 (1988): 433–460.

[16] Conn, A.R., N.I.M. Gould, and Ph. L. Toint. "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds." *SIAM J. Numer. Anal.* 28 (1991): 545–572.

[17] Conn, A.R., N.I.M. Gould, and Ph.L. Toint. *LANCELOT : A Fortran Package for Large-Scale Nonlinear Optimization (Release A).* Springer Series in Computational Mathematics 17. Berlin: Springer-Verlag, 1992.

[18] Dembo, R.S., S.C. Eisenstat, and T. Steihaug. "Inexact Newton methods." *SIAM J. Numer. Anal.* 19 (1982): 400–408.

[19] Dennis, J.E., Jr., M.M. El-Alem, and M.C. Maciel. "A global convergence theory for general trust-region-based algorithms for equality constrained optimization." Technical Report. Dept. of Mathematical Sciences, Rice University, 1993.

[20] Dennis, J.E., Jr., and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall Series in Computational Mathematics. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[21] Dikin, I.I. "Iterative solution of problems of linear and quadratic programming." *Soviet Math. Dokl.* 8 (1967): 674–675.

[22] Duff, I.S., A.M. Erisman, and J.K. Reid. *Direct Methods for Sparse Matrices.* Monographs in Numerical Analysis. Oxford, UK: Clarendon, 1986.

[23] Duff, I.S., N.I.M. Gould, J.K. Reid, J.A. Scott, and K. Turner. "The factorization of sparse symmetric indefinite matrices." *IMA J. Numer. Anal.* 11 (1991): 181–204.

[24] Duff, I.S., and J.K. Reid. "A comparison of some methods for the solution of sparse overdetermined systems of linear equations." *J. Inst. Maths. Applics.* 17 (1976): 267–280.

[25] Duff, I.S., and J.K. Reid. "Some design features of a sparse matrix code." *ACM Trans. Math. Software.* 5 (1979): 18–35.

[26] Duff, I.S., J.K. Reid, N. Munksgaard, and H.B. Nielsen. "Direct solution of sets of linear equations whose matrix is sparse, symmetric and indefinite." *J. Inst. Maths. Applics.* 23 (1979): 235–250.

[27] El-Alem, M.M. "A global convergence theory for the Celis-Dennis-Tapia trust region algorithm for constrained optimization." *SIAM J. Numer. Anal.* 28 (1991) : 266–290.

[28] El-Alem, M.M. "A robust trust region algorithm with a non-monotonic penalty parameter scheme for constrained optimization." Technical Report. Dept. of Mathematical Sciences, Rice University, 1993.

[29] Fiacco, A.V., and G.P. McCormick. "Computational algorithm for the sequential unconstrained minimization technique for nonlinear programming." *Management Sci.* 10 (1964) : 601–617.

[30] Fiacco, Anthony V., and Garth P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* New York: Wiley & Sons, 1968.

[31] Fletcher, R. "Conjugate gradient methods for indefinite systems." *Numerical Analysis Dundee 1975.* Ed. G.A. Watson. Lecture Notes in Mathematics 506. Berlin: Springer-Verlag, 1976. 73–89.

[32] Fletcher, R. "Numerical experiements with an exact $\ell_1$ penalty function method." *Nonlinear Programming 4.* Ed. Olvi L. Mangasarian, Robert R. Meyer, and Stephen M. Robinson. New York: Academic Press - Harcourt, 1981. 99–129.

[33] Fletcher, R. "An $\ell_1$ penalty method for nonlinear constraints." *Numerical Optimization 1984.* Ed. Paul T. Boggs, Richard H. Byrd, and Robert B. Schnabel. Philadelphia: SIAM, 1985. 26–40.

[34] Fletcher, R. *Practical Methods of Optimization.* 2nd ed. Chichester, UK: Wiley & Sons, 1987.

[35] Fourer, R., and S. Mehrotra. "Solving symmetric indefinite systems in an interior-point method for linear programming." *Math. Prog. (Ser. B)* 62 (1993): 15–39.

[36] Freund, R.W., and N.M. Nachtigal. "QMR: a quasi-minimal residual method for non-Hermitian linear systems." *Numer. Math.* 60 (1991): 315–339.

[37] Frisch, K.R. "The logarithmic potential method of convex programming." Memorandum. University Institute of Economics. Oslo, 13 May 1955.

[38] Gilbert, J. Ch., and C. Lemaréchal. "Some numerical experiments with variable-storage quasi-Newton algorithms." *Math. Prog. (Ser. B)* 45 (1989): 407–435.

[39] Gill, Philip E., Walter Murray, and Margaret H. Wright. *Practical Optimization.* London: Academic Press - Harcourt, 1981.

[40] Goldfeld, S.M., R.E. Quandt, and H.F. Trotter. "Maximization by quadratic hill-climbing." *Econometrica* 34 (1966): 541–551.

[41] Golub, Gene H., and Charles F. Van Loan. *Matrix Computations.* 2nd ed. Baltimore: Johns Hopkins UP, 1989.

[42] Gonzaga, C.C. "Path-following methods for linear programming." *SIAM Review* 34 (1992): 167–224.

[43] Goodman, J. "Newton's method for constrained optimization." *Math. Prog.* 33 (1985): 162–171.

[44] Gould, N.I.M. "On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem." *Math. Prog.* 32 (1985): 90–99.

[45] Gulliksson, M. "Algorithms for nonlinear least-squares with applications to orthogonal regression." Technical Report UMINF-178.90. University of Umea, Sweden, 1990.

[46] Hager, W.W. "Multiplier methods for nonlinear optimal control." *SIAM J. Numer. Anal.* 27 (1990) : 1061–1080.

[47] Han, S.P. "A globally convergent method for nonlinear programming." *J. Opt. Th. and Appl.* 22 (1977): 297–309.

[48] *Harwell Subroutine Library, Release 10.* Advanced Computing Department, AEA Industrial Technology, Harwell Laboratory. Oxfordshire, UK, 1990.

[49] Hock, W., and K. Schittkowski. *Test Examples for Nonlinear Programming Codes.* Lecture Notes in Economics and Mathematical Systems 187. Heidelberg: Springer-Verlag, 1981.

[50] Lalee, M. "Algorithms for nonlinear optimization." Diss. Dept. of Industrial Engineering, Northwestern University, 1992.

[51] Levenberg, K. "A method for the solution of certain nonlinear problems in least squares." *Quart. Appl. Math.* 2 (1944): 164–168.

[52] Li, Y. "Centering, trust region, reflective techniques for nonlinear minimization subject to bounds." Technical Report 93-1385. Dept. of Computer Science, Cornell University, 1993.

[53] Liu, D.C., and J. Nocedal. "On the limited memory BFGS method for large scale optimization." *Math. Prog. (Ser. B)* 45 (1989): 503–528.

[54] Luenberger, David G. *Linear and Nonlinear Programming.* 2nd ed. Reading, MA: Addison-Wesley, 1984.

[55] Maratos, N. "Exact penalty function algorithms for finite dimensional and control optimization problems." Diss. University of London, 1978.

[56] Marquardt, D.W. "An algorithm for least-squares estimation of nonlinear parameters." *SIAM J. Appl. Math.* 11 (1963): 431–441.

[57] Moré, J.J. "Recent developments in algorithms and software for trust region methods." *Mathematical Programming: The State of the Art.* Ed. A. Bachem, M. Grotschel, and B. Korte. New York: Springer-Verlag, 1983. 258–287.

[58] Moré, J.J., and D.C. Sorensen. "Computing a trust region step." *SIAM J. Sci. Stat. Comput.* 4 (1983): 553–572.

[59] Murtagh, B.A., and M.A. Saunders. "A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints." *Math. Prog. Study* 16 (1982): 84–117.

[60] Nocedal, J., and M.L. Overton. "Projective Hessian updating algorithms for nonlinearly constrained optimization." *SIAM J. Numer. Anal.* 22 (1985): 821–850.

[61] Omojokun, E.O. "Trust region algorithms for optimization with nonlinear equality and inequality constraints." Diss. Dept. of Computer Science, University of Colorado, 1991.

[62] Ortega, J.M., and W.C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Computer Science and Applied Mathematics Series. New York: Academic Press - Harcourt, 1970.

[63] Paige, C.C., and M.A. Saunders. "Solution of sparse indefinite systems of linear equations." *SIAM J. Numer. Anal.* 12 (1975): 617–629.

[64] Powell, M.J.D. "A hybrid method for nonlinear equations." *Numerical Methods for Nonlinear Algebraic Equations.* Ed. P. Rabinowitz. London: Gordon and Breach, 1970. 87–114.

[65] Powell, M.J.D. "Problems related to unconstrained optimization." *Numerical Methods for Unconstrained Optimization.* Ed. W. Murray. London: Academic Press, 1972. 29–55.

[66] Powell, M.J.D. "Convergence properties of a class of minimization algorithms." *Nonlinear Programming 2.* Ed. Olvi L. Mangasarian, Robert R. Meyer, and Stephen M. Robinson. New York: Academic Press - Harcourt, 1975. 1–27.

[67] Powell, M.J.D. "The convergence of variable metric methods for nonlinearly constrained optimization calculations." *Nonlinear Programming 3.* Ed. Olvi L. Mangasarian, Robert R. Meyer, and Stephen M. Robinson. New York: Academic Press - Harcourt, 1978. 27–63.

[68] Powell, M.J.D., and Y. Yuan. "A recursive quadratic programming algorithm that uses differentiable exact penalty functions." *Math. Prog.* 35 (1986): 265–278.

[69] Powell, M.J.D., and Y. Yuan. "A trust region algorithm for equality constrained optimization." *Math. Prog. (Ser. A)* 49 (1991): 189–211.

[70] Saad, Y., and M.H. Schultz. "GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems." *SIAM J. Sci. Stat. Comput.* 7 (1986): 856–869.

[71] Schultz, G.A., R.B. Schnabel, and R.H. Byrd. "A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties." *SIAM J. Numer. Anal.* 22 (1985): 47–67.

[72] Sonneveld, P. "CGS, a fast Lanczos-type solver for nonsymmetric linear systems." *SIAM J. Sci. Stat. Comput.* 10 (1989): 36–52.

[73] Sorensen, D.C. "Newton's method with a model trust region modification." *SIAM J. Numer. Anal.* 19 (1982): 409–426.

[74] Steihaug, T. "The conjugate gradient method and trust regions in large scale optimization." *SIAM J. Numer. Anal.* 20 (1983): 626–637.

[75] Strang, Gilbert. *Linear Algebra and its Applications.* 3rd ed. San Diego: Harcourt, 1988.

[76] Toint, Ph.L. "Towards an efficient sparsity exploiting Newton method for minimization." *Sparse Matrices and Their Uses.* Ed. I.S. Duff. New York: Academic Press - Harcourt, 1981. 57–87.

[77] Vardi, A. "A trust region algorithm for equality constrained minimization: convergence properties and implementation." *SIAM J. Numer. Anal.* 22 (1985): 575–591.

[78] Williamson, K.A. "A robust trust region algorithm for nonlinear programming." Diss. Dept. of Mathematical Sciences, Rice University, 1991.

[79] Wilson, R.B. "A simplicial algorithm for concave programming." Diss. Graduate School of Business Administration, Harvard University, 1963.

[80] Wright, M.H. "Interior methods for constrained optimization." *Acta Numerica 1992.* Ed. A. Iserles. N.p.: Cambridge UP, 1991. 341–407.

[81] Yuan, Y. "On a subproblem of trust region algorithms for constrained optimization." *Math. Prog. (Ser. A)* 47 (1990): 53–63.

[82] Zhang, J.Z., and D.T. Zhu. "Projected quasi-Newton algorithm with trust region for constrained optimization." *J. Opt. Th. and Appl.* 67 (1990): 369–393.

[83] Zhang, Y. "Computing a Celis-Dennis-Tapia trust region step for equality constrained optimization." *Math. Prog. (Ser. A)* 55 (1992): 109–124.