A BARRIER ALGORITHM FOR LARGE NONLINEAR OPTIMIZATION PROBLEMS

A DISSERTATION
SUBMITTED TO THE SCIENTIFIC COMPUTING
AND COMPUTATIONAL MATHEMATICS PROGRAM
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
SCIENTIFIC COMPUTING AND COMPUTATIONAL MATHEMATICS

Maureen Doyle
December 2003

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Walter Murray
(Principal Advisor)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Michael A. Saunders

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Gene H. Golub

Approved for the University Committee on Graduate Studies:

Abstract

The problem of large-scale constrained optimization is addressed. A barrier function is used to transform the problem into a sequence of subproblems with nonlinear equality constraints. Barrier methods differ primarily in how such subproblems are solved. The method used here for each subproblem is similar to what the second-derivative method of Murray and Prieto (MP) reduces to when applied to equality-constrained problems. Second derivatives are used to generate a search direction and a direction of negative curvature for a curvilinear search. The MP method would be based on the KKT equations for the subproblem, but here we use the primal-dual transformation of those equations.

A key feature of our method is that only a single system of linear equations is solved at each subproblem iteration (as in barrier methods for linear programming). In contrast, a trust-region method applied to the subproblem may require the solution of several linear systems, and sequential quadratic programming (SQP) methods applied directly to the original problem require the solution of a quadratic program at each iteration. For small problems this is of little consequence, but for large problems, especially when iterative methods are necessary to solve the linear systems, it may prove a vital advantage. The advantage would be lost if the number of iterations were significantly more than for alternative methods. Since analysis cannot predict iteration counts for general nonlinear problems, an experimental MATLAB code named MELBA has been implemented to investigate this aspect of performance. MELBA is compared primarily to another second-derivative barrier method, KNITRO, whose subproblems are solved by a trust-region method. We also give results for SNOPT, which uses a quasi-Newton SQP method. For the common set of problems solved the results are encouraging because there is little to distinguish between the three sets of results, even though there is much room for refining the new method. It is shown that the method converges globally to a point satisfying the second-order necessary conditions, without an assumption that the iterates lie in a compact set. Moreover, the iterates converge at a quadratic rate. A feature of the method is that the implementation is close to that of the theoretical algorithm.

Acknowledgments

First and foremost, I would like to thank my advisor Professor Walter Murray. Professor Murray's upbeat attitude and depth of knowledge of optimization was the backbone that made this thesis possible. I appreciate the time he spent with me to discuss research and the help and guidance he gave me when I felt that I had hit an impasse. He also provided me with a wonderful work environment and vital financial support.

I also would like to thank Professor Michael Saunders for his significant contributions to the software and his thorough review of this thesis. Professor Saunders allowed me to speak freely, and for that I am so grateful. I value both his professionalism and patience.

I appreciate Professor Gene Golub's participation in my thesis review committee. As the director of SCCM for part of my tenure as a Stanford Ph.D. candidate, Professor Golub's passion for SCCM helped create a wonderful learning community for me. I also thank him for the financial support which supplemented my TA funding during my third year.

I wish to thank Dr. Noe Lozano, for selecting me, and Professor Walter Murray, for nominating me, for the Dean's Doctoral Fellowship that supported me during my first eight semesters at Stanford.

I also want to acknowledge my colleagues who were always available to discuss academic, research and other pertinent issues. My office mates at Stanford included Melissa Aczon, Alexis Collomb, Anthony Diaco, Michael Friedlander, Sep Kamvar, Byunggyoo Kim, Brad Mallison, Neil Molino, Kien Ming Ng, Vinayak "Uday" Shanbhag, Herser Sirgurgeirsson, Che-Lin Su and James Warren.

I could not have acclimated to or enjoyed life at Stanford if it was not for the wonderful and supportive Stanford staff. In particular, I would like to thank Evelyn Boughton and Arden King from SCCM; Lorrie Pappadakis and Landon Phan from Management Science and Engineering; Dana Halpin from Computer Science; Chris Griffith from the Graduate Life Office; Stephanie Eberle from the Career Development Center; and both Rabbi Patricia Karlin-Neumann and Reverend Joanne Saunders from the Religious Life Office.

I am grateful to my peers and superiors from BBN Technologies. I want to thank them for their support of my decision to return to school, their encouragement and interest in my studies. I am also thankful for my summer employment at BBN, during part of my tenure at Stanford. Mark Berman, Jeff Berliner, Ed Campbell, Therese Cwikla Maguire, Tom Mitchell and Carol Sabia were especially helpful and kind. I am

indebted to you all.

Thank you to my new employer, Morehead State University. I appreciate the support of my peers, the Department Chair of Mathematics and Computer Science, Dr. Rodger Hammons, and the Dean of Science and Technology, Dr. Gerald DeMoss. And, I appreciate their patience while I served double-duty as a professor at Morehead and student at Stanford.

I am thankful for my friends from Stanford and California, including Kanaka Szabo, Debbie Ballard, Steve Bryson, Selena Burns, Maggie Davidson, Joel Franklin, Holly (Lutze) Guerin, Paula Razquin, Paul and Joanne Romberg and Amanda White. Their support and sharing of their lives kept me grounded and helped me grow.

I am indebted to my family for the love and support during my five years at Stanford. I would like to thank my 'California' family: My Godparents, Uncle Ted and Aunt Eleanore Doyle, and my cousins who all opened their hearts and home to me. I appreciate their friendship and love and cherish the time we were neighbors. I am also grateful for the support and easy laughs I enjoyed with my cousins Kirsten Davis and Leslie Doyle Fitzpatrick. Thanks to my 'East Bay' cousins Tim and Barbara Daniels, who always provided a wonderful respite from Stanford life.

A special thank you and much gratitude to my Mom, siblings Robert, Barbara, Timothy, Shawn and Patty, sibling-in-law, Geraldine Halpin-Doyle, and my nieces, Alicia Halpin and Danielle Doyle. I am forever grateful for the phone calls, visits and gentle handling from them all. The unwavering support, love and faith from Mom, Barb and Patty, both motivated and anchored me. I could not have accomplished this thesis without having them in my corner.

Lastly, to all of those acknowledged here and the unnamed family, friends, teachers and co-workers, thank you for being you and letting me be a part of your lives.

Contents

Algorithms, Figures, and Tables

Algorithms

Figures

Tables

Chapter 1

Introduction

We present an interior-point method named MELBA (MATLAB Experimental Linesearch Barrier Algorithm) for solving optimization problems with nonlinear constraints. The general form of the problem is

NLP

$$\begin{array}{lll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) & \\ \text{subject to} & c_i(x) = 0 & i = 1, 2, \ldots, m \\ & x_j \geq 0 & j = 1, 2, \ldots, n, \end{array} \tag{1.1}$$

where $f(x)$ is a function with known gradient $g(x)$ and Hessian $H(x)$, $c(x) \in \mathbb{R}^m$ is a vector of the $c_i(x)$ functions with a known Jacobian $J(x) \in \mathbb{R}^{m \times n}$, and $H_i(x)$ is the known Hessian for $c_i(x)$. The format of NLP is suitable for general equality or inequality constraints. Our method is intended for optimization problems with smooth functions and allows some of the functions $f(x)$ and/or $c_i(x)$ to be nonconvex.

The term *interior-point method* implies that the solution process maintains strict inequality for constraints that are expressed as inequalities. For problem (1.1), this means that all approximate solutions $\breve{x}$ will satisfy $\breve{x} > 0$, even though some components $\breve{x}_j$ may be converging towards zero.

Interior-point methods were initially proposed by Frisch in 1955 [Fri55]. Fiacco and McCormick [FM68] proved global convergence for general interior-point methods for problem (1.1) by reformulating this problem as an unconstrained optimization problem. Classical log-barrier methods, one type of interior-point algorithm, were used extensively in the 1960's and 1970's. It was proven in the 1970's [Mur71] that the Hessians for barrier methods are ill-conditioned at the solution, so despite reasonable results, other methods became primary topics for research.

In 1984, Karmarkar presented an algorithm that solved linear optimization problems in polynomial time [Kar84]. This was a significant improvement over current algorithms (notably the Simplex method), which solved worst-case problems in exponential time. It was soon shown that Karmarkar's algorithm was equivalent to the log-barrier method

[GMSTW86] and interest in interior-point methods resurged.

Megiddo first proposed an interior-point method that solved the primal and dual variables simultaneously. In [Meg89], Megiddo describes many of the properties of the primal and dual central paths for linear programming. A primal-dual barrier method for linear problems was then implemented by [KMY89]. Methods of this type were shown to solve linear programming problems faster than general primal barrier methods. Barrier methods are now commonly used for solving linear programming problems as described in [Wri97] and for quadratic programming [Pon90]. Current large-scale implementations include CPLEX [ICPLEX], OSL [Hun98], and MOSEK [MOSEK].

As a result of the success of barrier methods for both linear and quadratic programming, research into their use for nonlinear optimization continues today. Current research areas and results are discussed in later chapters. This thesis discusses one barrier approach for solving nonlinear optimization problems. We reformulate the problem into a sequence of equality-constrained optimization subproblems. Each subproblem is altered using the primal-dual approach. The altered subproblem is then solved using the reduced Hessian of the Lagrangian and negative curvature for determining the sequence of search directions. The step size is computed using a linesearch method to reduce an augmented Lagrangian function. MELBA is tailored for large, sparse problems. The termination criteria for each subproblem, and the sequence of subproblems, are designed to ensure convergence to a point satisfying the necessary conditions for second-order optimality. The necessary conditions for optimality are defined in section 1.3.

The rest of this introduction provides the foundation and background needed for describing MELBA.

1.1 Nomenclature and definitions

This section contains general nomenclature and definitions used throughout.

1.1.1 Nomenclature

- *C*apital letters, both Greek and English, are used to represent matrices.
- *l*owercase letters, both Greek and English, are used to represent vectors.
- $K \succ 0$ defines K to be positive definite.
- $K \succeq 0$ defines K to be positive semidefinite.
- $\lambda_{\max}(K)$ and $\lambda_{\min}(K)$ denote the maximum and minimum eigenvalues of K.
- e identifies a vector of all $1's$.

- e_i identifies a vector of all $0's$, except $e_i(i) = 1$.
- diag(x) and $\mathbf{X}$ are both the matrix having the vector x on the diagonal and zero elsewhere.

1.1.2 Definitions

- $f_B(x,\mu) = f(x) - \mu \sum_{j=1}^{n} \ln x_j$ is the objective function with the barrier terms.
- $g(x)$ is the gradient of $f(x)$. $g(x) \equiv \nabla f(x)$, $g(x) \in I\!R^n$.
- $g_B(x,\mu)$ is the gradient of $f_B(x,\mu)$. $g_B(x,\mu) \in I\!R^n$.
- $J \equiv \nabla c(x)$ is the Jacobian for $c(x)$. $J_{ij} = \frac{\partial c_i(x)}{\partial x_j}$, $J \in I\!R^{m\times n}$.
- $H(x)$ is the Hessian of $f(x)$. $H(x) \equiv \nabla^2 f(x)$, $H(x) \in I\!R^{n\times n}$.
- $H_i(x)$ is the Hessian of $c_i(x)$. $H_i(x) \equiv \nabla^2 c_i(x)$, $H_i \in I\!R^{n\times n}$.
- I_n is the identity matrix, $I_n \in I\!R^{n\times n}$.
- y is a vector of Lagrange multipliers for the constraints $c(x) = 0$, $y \in I\!R^m$.
- z is a vector of Lagrange multipliers for the constraints $x \geq 0$, $z \in I\!R^n$.
- $L(x,y) \equiv f(x) - y^T c(x)$ is the Lagrangian for problem (1.1), $L \in I\!R$.
- $g_L(x,y) \equiv g(x) - J(x)^T y$ is the gradient of the Lagrangian, $g_L \in I\!R^n$.
- $H_L(x,y) \equiv H(x) - \sum_{i=1}^{m} y_i H_i(x)$ is the Hessian of the Lagrangian, $H_L \in I\!R^{n\times n}$.
- $W_B(x,y,z) \equiv H_L(x,y) + \mu \mathbf{X}^{-2}$ is the Hessian of the Lagrangian for the barrier subproblem (2.7), $W_B \in I\!R^{n\times n}$.
- $W(x,y,z) \equiv H_L(x,y) + \text{diag}(\mathbf{X}^{-1}z)$ is the Hessian of the Lagrangian for the primal-dual reformulation of the barrier subproblems (2.7), $W \in I\!R^{n\times n}$.
- $Z(x)$ spans the null space of the Jacobian of the equality constraints $c_i(x) = 0$, $Z(x) \in I\!R^{n\times(n-m)}$. $Z(x)$ satisfies $JZ(x) = 0$ and exists when $n > m$.
- $Z(x)^T W(x,y,z) Z(x)$ is the reduced Hessian of the Lagrangian.
- $\overline{Z(x)^T W(x,y,z) Z(x)}$ is positive definite and is called the modified reduced Hessian of the Lagrangian. It is computed by altering $Z^T W Z$ to obtain a positive definite symmetric matrix.
- $L_A(x,y,\rho) \equiv f(x) - y^T c(x) + \frac{1}{2} c(x)^T R c(x)$ is the augmented Lagrangian for problem (1.1), $L_A(x,y,\rho) \in I\!R$.

- ρ is the vector of penalty parameters for the augmented Lagrangian function. $\rho \in \mathbb{R}^m$, $\rho_i > 0$.
- $R = \text{diag}(\rho)$ is the matrix of penalty parameters for the augmented Lagrangian function.
- d is a direction of negative curvature for the reduced Hessian of the Lagrangian for the barrier subproblem, such that $d^T Z^T W Z d < 0$. $d \in \mathbb{R}^{n-m}$.
- dn is a direction of negative curvature for the full Hessian of the Lagrangian for the barrier subproblem, W, such that $dn^T W dn < 0$. $dn \in \mathbb{R}^n$.
- $\mathbf{X} = \text{diag}(x)$, where x is a vector of primal variables.
- $\mathbf{Z} = \text{diag}(z)$, where z is a vector of dual variables.
- $\tilde{x}$ is the next iterate of x when a line search is used. Given a search direction Δx, an initial x and a step size α, $\tilde{x} = x + \alpha \Delta x$.
- $\hat{x}$ is the next iterate of x when a curvilinear search is used to determine the step size. Given a search direction, Δx, a direction of negative curvature, dn, a step size α and initial x, $\hat{x} = x + \alpha^2 \Delta x + \alpha dn$.
- $\breve{x}$ indicates the next iterate of x, regardless of step size algorithm.
- **funObj(x)** is a user-supplied function that calculates the objective function $f(x)$ and (optionally) its gradient for a specified vector x.
- **funCon(x,y)** is a user-supplied function that calculates the vector of constraint functions $c(x)$ and (optionally) its Jacobian for specified vectors x and the constraint Lagrange multpliers, y.
- **funHess(x,y,v,gotH)** is a user-supplied function that calculates $H_L(x, y)$ multiplied by a vector, v. The Hessian of the Lagrangian is defined for the specified vectors x and y. When $gotH$ is TRUE, the last saved $H_L(x, y)$ is used; otherwise a new matrix is calculated.

1.2 Statement of problem

The standard form for nonlinear programming problems is

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & l \leq \begin{pmatrix} c_i(x) \\ x_j \end{pmatrix} \leq u, \end{array} \tag{1.2}$$

where $f(x)$ is a function with known gradient $g(x)$ and Hessian $H(x)$, $c(x) \in \mathbb{R}^m$ is a vector of the $c_i(x)$ functions with a known Jacobian $J \in \mathbb{R}^{m \times n}$, and $H_i(x)$ is the known Hessian for $c_i(x)$. In order to obtain the general form (1.1), system (1.2) is reformulated. Slack variables, s, are added to the nonlinear constraints:

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) - s = 0 \\ & l \leq \begin{pmatrix} s_i \\ x_j \end{pmatrix} \leq u. \end{array} \tag{1.3}$$

Removing the bounds from the variables in the nonlinear equations reduces to the form

$$\begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) - s = 0 \\ & \begin{pmatrix} s_i \\ x_j \end{pmatrix} + t_1 = u, \qquad \begin{pmatrix} s_i \\ x_j \end{pmatrix} - t_2 = l \\ & t_1 \geq 0,\ t_2 \geq 0, \end{array} \tag{1.4}$$

which is in the general form of problem (1.1). In subsequent sections this problem is simplified using fewer variables when the algorithm or function being discussed is not affected by the inclusion of the variable. For example, $c(x)$ is used instead of $c(x) - s$ when the inclusion of s does not alter the major steps in the algorithm.

Problem (1.4) can be reduced further by substituting $x = (x, t, s)$ and $c(x) = c(x) - s$ to obtain

$$\begin{array}{lllll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) & & & \\ \text{subject to} & c_i(x) = 0 & i = 1, 2, \ldots, m & : y \\ & x_j \geq 0 & j = 1, 2, \ldots, n & : z, \end{array} \tag{1.5}$$

where y, z are called the Lagrange multipliers or dual variables. The term dual variables comes from linear programming. For any linear programming problem with primal variables x and dual variables y, there is a companion maximization problem with primal variables y and dual variables x. The term Lagrange multipliers is used here to indicate the variables y. The term dual variables defines either z or (y, z), depending on context. When solving (1.1), we solve for the primal (x) and the dual (y, z) variables.

1.3 Optimality conditions

Consider NLP (1.1), where $f(x)$ and $c_i(x)$ are twice-continuously differentiable. A point x^* is a weak local minimizer of (1.1) if $x^* \geq 0$, $c(x^*) = 0$ and there exists a $\delta > 0$ such

that $f(x) \geq f(x^*)$ for all x satisfying

$$\|x - x^*\| \leq \delta,\ x \geq 0,\ c(x) = 0.$$

Determining that x^* satisfies these constraints is usually not practical, so most optimization methods determine that x^* satisfies equivalent, verifiable optimality conditions. Assuming the constraint qualification that J^* has full row rank, then the following are the optimality conditions for NLP.

Definition 1.1. *Necessary optimality conditions*

If J^* has full row rank, then x^* is a local minimizer of NLP when there exist multipliers y^* and z^* such that (x^*, y^*, z^*) satisfy the Karush-Kuhn-Tucker (KKT) second-order conditions:

$$\begin{aligned} c(x^*) &= 0, \\ \nabla f(x^*) &= J^{*T} y^* + z^*, \\ x^* &\geq 0, \\ z^* &\geq 0, \\ Z^{*T} H_L^* Z^* &\succeq 0, \\ x_j^* > 0 \rightarrow z_j^* &= 0, \\ z_j^* &> 0,\ \forall\ x_j^* = 0, \\ x_j^* &> 0,\ \forall\ z_j^* = 0, \end{aligned}$$

where Z^* is a basis for the nullspace of J^*. $H_L^* \equiv H(x^*) - \sum_{i=1}^{m} y_i^* H_i(x^*)$ is the Hessian of the Lagrangian for (1.1). The first-order condition is $\nabla f(x^*) = J^{*T} y^* + z^*$ and the second-order condition is $Z^{*T} W^* Z^* \succeq 0$.

Definition 1.2. *Sufficient optimality conditions*

If J^* has full row rank, then for (x^*, y^*, z^*) to be a local minimizer of NLP the necessary optimality conditions hold and $Z^{*T} W^* Z^* \succ 0$.

1.4 Thesis contents

The following chapter discusses interior-point methods in-depth and presents the barrier algorithm, MELBA. Chapter 3 specifies and outlines the algorithms implemented for processing the large matrices that may occur in nonlinear optimization problems. Chapters 4 and 5 discuss the algorithm in detail. Chapter 6 contains the convergence theory and Chapter 7 the computational results.

For simplicity, vectors and matrices that are functions of x (e.g., $H(x)$) will initially be defined and written with (x); however, this term will be dropped in subsequent references (e.g. H is used instead of $H(x)$).

Chapter 2

Interior-Point Methods

In this chapter we define interior-point methods (IPMs), discuss current research using IPMs in nonlinear optimization, summarize the barrier approach and primal-dual formulation, and presents our barrier algorithm.

2.1 Interior-point methods for nonlinear optimization

The term *interior-point method* was originally used by Fiacco and McCormick in [FM68] to describe any algorithm that computes a local minimum of a nonlinear optimization problem (NLP) by solving a specified sequence of *unconstrained* minimization problems. This definition has evolved to include any method that solves a set of optimization problems involving a decreasing multiplier μ, searching for local solutions within the interior of the feasible region of the inequality constraints for the NLP.

2.1.1 Sequential unconstrained minimization

Consider the inequality constrained problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & c_i(x) \geq 0, \quad i = 1, \ldots, m, \end{array} \tag{2.1}$$

where $f(x)$ and $c_i(x)$ are twice-continuously differentiable functions. To describe an interior unconstrained minimization function, Fiacco and McCormick first define two scalar-valued functions $I(x)$ and $s(r)$ for vector x and scalar r [FM68].

Definition 2.1. *$I(x)$ is any function $I(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ with the following properties:*

Property 1 $I(x)$ is continuous in the region $R^0 = \{x \mid c_i(x) > 0, i = 1, \ldots, m\}$.

Property 2 If $\{x_k\}$ is any finite sequence of points in R^0 converging to x such that $c_i(x) = 0$ for at least one i, then $\lim_{k \rightarrow \infty} I(x_k) = +\infty$.

Definition 2.2. *$s(r)$ is any function $s(r) : \mathbb{R} \to \mathbb{R}$ with the following properties:*

Property 1 If $r_1 > r_2 > 0$, then $s(r_1) > s(r_2) > 0$.

Property 2 If $\{r_k\}$ is an infinite sequence of points such that $\lim_{k\to\infty} r_k = 0$, then $\lim_{k\to\infty} s(r_k) = 0$.

Definition 2.3. *Sequential Unconstrained Minimization Function*

Given two scalar-valued functions $I(x)$ and $s(r)$ as in Definitions 2.1 and 2.2, define the function

$$U(x, r_k) \equiv f(x) + s(r_k)I(x) \tag{2.2}$$

as an interior unconstrained minimization function [FM68].

The minimization subproblem solved is

$$\min_x U(x, r_k). \tag{2.3}$$

The solution to this subproblem is $x(r_k)$. Fiacco and McCormick, under general assumptions, prove convergence of the subsequence $\{x(r_k)\}$ to a local minimizer, $x^*(r_k)$, of problem (2.3). They also prove that the sequence of local minimizers $\{x^*(r_k)\}$ converges to a local minimizer of the original problem (2.1). Thus, interior-point methods for problems of the form (2.1) converge to a solution by solving a specified sequence of altered subproblems.

Theorem 2.4. *For problem (2.1), assume the functions $f, c_1, \ldots, c_m$, are continuous, $U(x, r_k) = f(x) + s(r_k)I(x)$ is an interior unconstrained minimization function with $s(r)$ and $I(x)$ as in Definitions 2.1 and 2.2, the problem has at least one local minimum in the closure of R^0, and $\{r_k\}$ is a strictly decreasing null sequence.*

Given these assumptions, if x^* is an isolated local minimizer of problem (2.1) and $x(r_k)$ is the minimizer for each $U(x, r_k)$, then the sequence $\{x(r_k)\}$ converges to x^*.

Proof. See [FM68]. ∎

The sequence $\{x(r_k)\}$ of Theorem 2.4 forms a barrier trajectory of solutions that converge to x^*.

Two common examples of $U(x, r_k)$ are the inverse barrier and logarithmic barrier functions. First, consider $I(x) = \frac{1}{c(x)}$, along with a decreasing sequence $\{\mu_k\} = \{(\frac{1}{2})^k\}$ and $s(\mu_k) = \mu_k$. The two functions and sequence satisfy the above properties, and therefore the function

$$U(x, r_k) = f(x) + \mu_k \sum_{i=1}^{m} \frac{1}{c_i(x)}$$

is an interior unconstrained minimization function. Of course, it is no longer a continuous function in view of the singularity when $c_i(x) = 0$. This particular formulation is called the inverse barrier function and was introduced by Carroll [Car59, Car61].

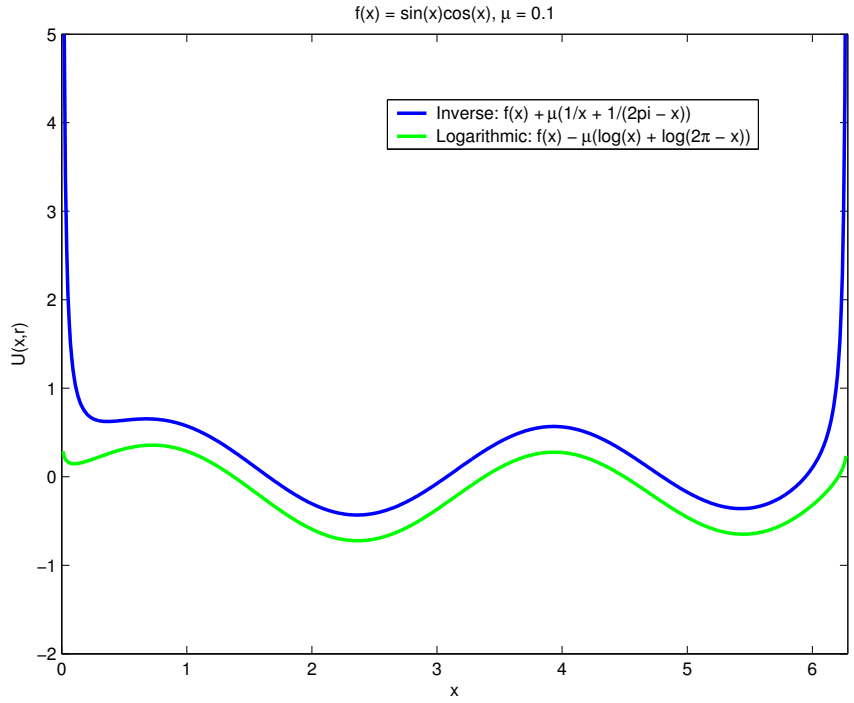

FIGURE 2.1 *Interior-Point examples*

Second, consider the function $I(x) = -\ln(c(x))$, along with the sequence μ_k defined above. These functions similarly satisfy the definitions for $I(x)$ and $s(r)$, and therefore

$$U(x, r_k) = f(x) - \mu_k \sum_{i=1}^{m} \ln(c_i(x))$$

is also an interior unconstrained minimization function. This particular function is called the logarithmic barrier function, or barrier for short.

Figure 2.1 is a plot of the barriers formed by the inverse and logarithmic barriers for $\mu = 0.1$ and the nonlinear problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sin(x)\cos(x) \\ \text{subject to} \quad & 0 \leq x \leq 2\pi. \end{aligned}$$

We now present a specific problem, used in [GMW81], to illustrate how the logarithmic barrier function converges to a solution. Consider the problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & xy^2 \\ \text{subject to} \quad & 2 - x^2 - y^2 \geq 0. \end{aligned} \tag{2.4}$$

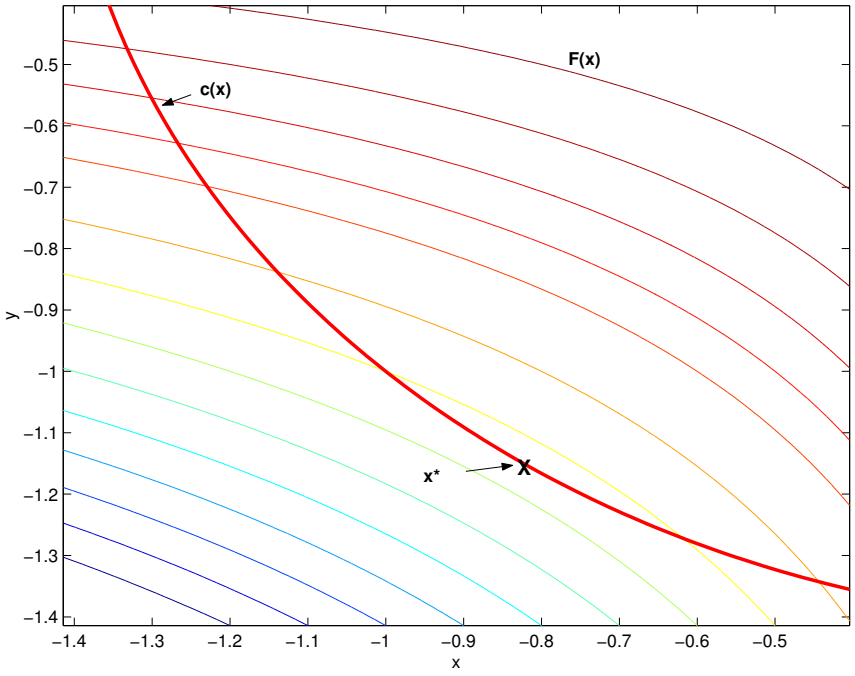

FIGURE 2.2 *Constrained nonlinear problem*

Figure 2.2 shows the values of xy^2 as contours. The red line is the equality constraint $2 - x^2 - y^2 = 0$. The solution is identified as $x^* = (-.81650, -1.1547)$.

Figure 2.3 shows the contours for the barrier reformulation when $\mu = 0.1$ where its minimum is visibly 'close' to the solution x^*.

Figure 2.4 illustrates the barrier trajectory for problem (2.4). The solutions to the subproblems are plotted for $\mu = (1, 0.5, 0.1)$. The barrier trajectory is the curve connecting the solution of each subproblem. Note how the solution for subproblems for larger μ's are well behaved and also as μ decreases, the solution approaches the true solution of problem (2.4).

2.1.2 Using interior-point methods

When second derivatives do not exist, IPMs are not very useful. If they exist but are not available, the Hessian of the Lagrangian $H_L(x, y)$ must be estimated. This creates a difficulty for IPMs. Either the barrier term is combined into H_L or the barrier term will be explicitly computed and the H_L will be estimated. In the first case, the H_L will change significantly with small changes in x as a result of the barrier terms. Therefore, the estimate for the Hessian of the Lagrangian never converges to the true Hessian and it can not be proven that the IPM will converge even to the first-order optimality conditions, as defined in section 1.3. In the latter case, the Hessian is a combination of

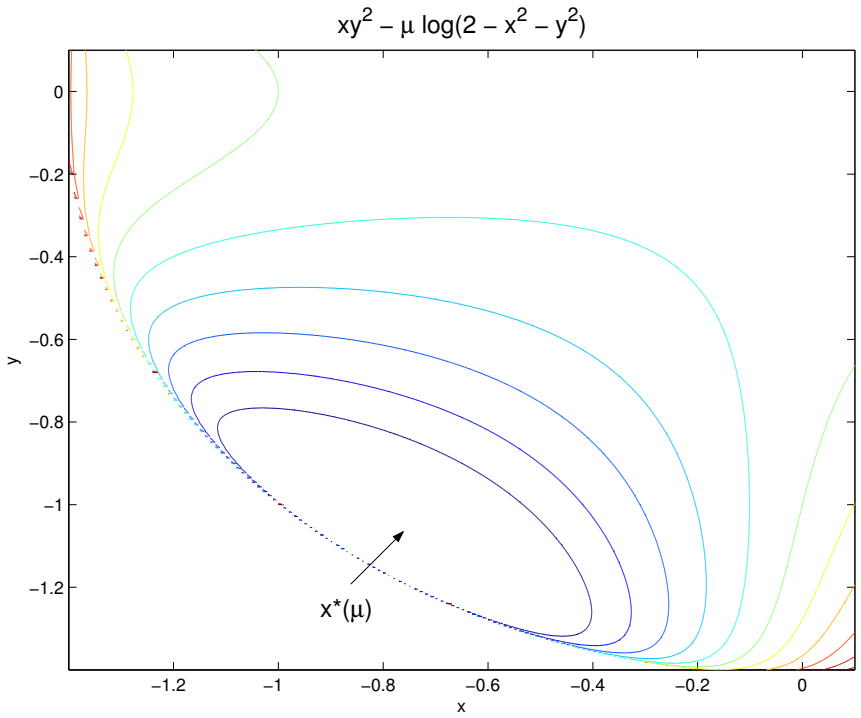

FIGURE 2.3 *Barrier formulation of constrained nonlinear problem*

explicit terms and estimated terms. Handling this combination has been problematic in other applications and is not recommended here. Therefore, when second derivatives are not on hand, other methods should be used.

SQP methods constitute one of the more successful approaches to solving (1.1) when only first derivatives are available. They do not require second derivatives and, in fact, most current SQP methods estimate the Hessian of the Lagrangian. However, there is still the cost of computing the approximate reduced Hessian for each change to the active set of constraints. This expensive computation is largely avoided with interior-point methods because all inequality constraints are eliminated from the problem with the function $I(x)$, so there is only a single reduced Hessian to be dealt with.If the reduced Hessian is very large then the conjugate gradient method may be more efficient, as it economizes on storage but at the expense of an unknown number of iterations. This option is less attractive with an active set method since there are more reduced Hessian systems to be solved [though each system will in general be smaller].

When second derivatives are available, using a barrier method allows convergence to a second-order minimizer, as we show in Chapter 6.

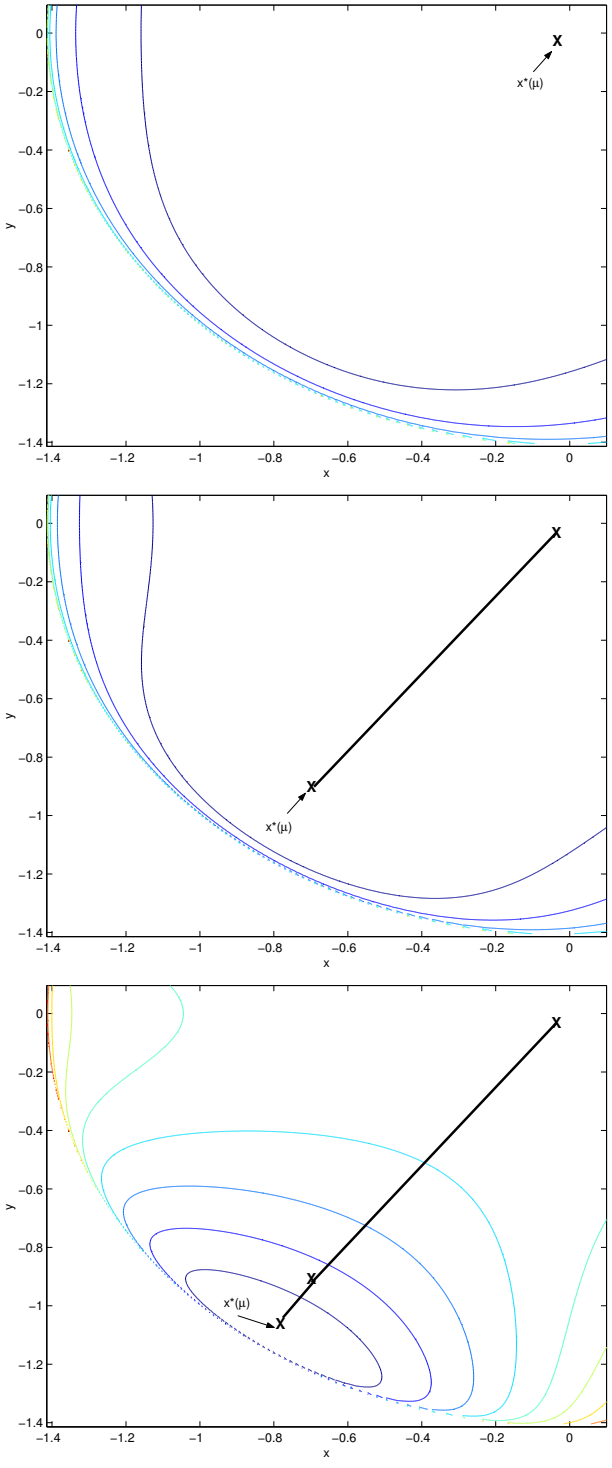

FIGURE 2.4 *Barrier trajectory example*

2.1.3 Other research

There has been significant research into the use of interior-point methods for nonlinear optimization, and in this section we compare MELBA to other IPM projects.

Many of the available implementations are designed for large-scale problems but do not have convergence theory or proven convergence rates. For example, Byrd et al. developed KNITRO (Nonlinear Interior-Point Trust Region Optimization) [BHN99], which applies SQP techniques to a sequence of barrier subproblems. In [BGN96], Byrd et al. prove global convergence for KNITRO, but do not prove convergence to second-order optimality conditions. Nor is the rate of convergence determined.

Sargent and Ding [SD00] developed SQPIPM, which solves a sequence of QP subproblems by an interior-point method using an inexact Newton approach. They prove finite termination of the algorithm and Q-quadratic convergence.

Shanno and Vanderbei [SV00] developed LOQO (Linear Optimization/Quadratic Optimization) using Newton's method applied to the first-order optimality conditions for a primal-dual barrier formulation. They report good performance but do not provide proofs for global convergence or convergence to second-order optimality conditions, nor the convergence rate of the algorithm.

Gertz and Gill implemented a trust-region barrier algorithm, IOTR (Interior-Point Optimization, Trust-Region) [GG02]. They prove the desired convergence properties: global convergence; convergence to second-order optimality conditions; and superlinear convergence rate. They use a trust-region approach, in contrast, to the linesearch approach used in MELBA. Moguerza and Prieto [MP03] implemented a linesearch algorithm and have also proven the desired convergence properties, but their implementation was not designed for large problems (sparse-matrix methods are not used).

2.2 The log-barrier function

As mentioned in section 2.1.1, barrier methods are a class of IPMs for solving the NLP (1.1) when the feasible region has a nonempty interior. The barrier method reformulates inequality constraints into a logarithmic barrier term in the objective. Specifically, problem (1.1) is reformulated into the equality constrained problem

$$\begin{array}{ll} \underset{x>0,\, x\in\mathbb{R}^n}{\text{minimize}} & f(x) - \mu \sum_{j=1}^{n} \ln(x_j) \\ \text{subject to} & c_i(x) = 0 \qquad\qquad i = 1, 2, \ldots, m, \end{array} \tag{2.5}$$

where $\mu > 0$ is the barrier parameter.

Fiacco and McCormick proved convergence for the inequality constrained problem (2.1). However, equation (1.1) has additional equality constraints. Initially, equality constraints were reformulated using penalty functions, giving the following uncon-

strained optimization problem:

$$\underset{x>0,\ x\in\mathbb{R}^n}{\text{minimize}} \quad f(x) - \mu \sum_{j=1}^{n} \ln(x_j) + \rho c(x)^T c(x), \tag{2.6}$$

Methods for unconstrained optimization were found to converge slowly on such problems because of the inherent ill-conditioning of the Hessian at the solution for small values of μ and large values of ρ. When interest in barrier methods was rekindled in the late 1980's, the problem of slow convergence was still an open issue.

As a result of ideas presented in [Meg89] for linear programming, [KMY89] first demonstrated improved convergence rates for linear programs by solving for the primal (x) and dual (y, z) variables simultaneously with a judicious reformulation of the KKT equations. This approach is called the primal-dual method. It was also proven in [KMY89] that for linear programming the barrier approach using a primal-dual method converges in polynomial time. This is the approach used for our nonlinear barrier algorithm, as further explained in the next section.

2.3 Primal-dual equations

In this section we derive the equations for the primal-dual method for the NLP (1.1). Let us first introduce the dual variables as multipliers for our constraints. The vector y is the multiplier for the equality constraints and z is the multiplier for the inequality constraints:

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) & \\ \text{subject to} & c_i(x) = 0 \quad i = 1, 2, \ldots, m & : y \\ & x_j \geq 0 \quad\ \ j = 1, 2, \ldots, n & : z. \end{array}$$

Replacing the non-negativity constraints by a log-barrier function and introducing the barrier parameter μ ($\mu > 0$) we obtain:

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) - \mu \sum_j \ln(x_j) \\ \text{subject to} & c_i(x) = 0 \qquad i = 1, 2, \ldots, m. \end{array} \tag{2.7}$$

The first-order KKT conditions for the barrier subproblem involve the *primal* equations of NLP(μ), along with *dual* equations stating that the gradient of the subproblem objective should be a linear combination of the gradients of the primal constraints:

$$\begin{array}{rl} c(x) = & 0 \\ J^T y = & g(x) - \mu \mathbf{X}^{-1} e, \end{array} \tag{2.8}$$

where $\mathbf{X} = \text{diag}(x)$. The primal-dual method introduces a positive vector z defined by the equation $\mathbf{X}z = \mu e$. The primal-dual conditions for (2.7) then become

$$\begin{aligned} c(x) &= 0 \\ J^T y + z &= g(x) \\ \mathbf{X}z &= \mu e. \end{aligned} \tag{2.9}$$

If Newton's method is applied to (2.9) the resulting equation for the Newton step is

$$\begin{pmatrix} -H_L & I & J^T \\ \mathbf{Z} & \mathbf{X} & 0 \\ J & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta z \\ \Delta y \end{pmatrix} = \begin{pmatrix} g(x) - J^T y - z \\ \mu e - \mathbf{X}z \\ -c(x) \end{pmatrix}, \tag{2.10}$$

where $H_L(x, y) \equiv H(x) - \sum_{i=1}^{m} y_i H_i(x)$ is the Hessian of the Lagrangian and $\mathbf{Z} = \text{diag}\,(z)$. Eliminating Δz using $\mathbf{X}\Delta z = (\mu - \mathbf{X}z) - \mathbf{Z}\Delta x$ gives

$$\begin{pmatrix} -W & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} g - \mu \mathbf{X}^{-1} e - J^T y \\ -c(x) \end{pmatrix}, \tag{2.11}$$

where $W = H_L + \mathbf{X}^{-1}\mathbf{Z}$.

The equivalent equations for the system (2.8) are identical to (2.11) except the matrix W is replaced by $H_L(x, y) + \mu \mathbf{X}^{-2}$.

2.4 Overview of our algorithm

The basic idea is to solve (2.7) for a sequence of decreasing μ. Since convergence to the solution of (2.7) is in the limit it is necessary to terminate prior to finding the exact solution. Barrier methods differ in the choice of algorithm to solve (2.7), how μ is adjusted, and the choice of termination conditions. The basic method we adopt to solve (2.7) for each μ is the second-derivative SQP algorithm of [MP95] (MP), except the search direction and direction of negative curvature are derived from the system (2.11) rather than having $W = H_L(x, y) + \mu \mathbf{X}^{-2}$. The properties of these directions are identical to those obtained in [MP95]. Indeed, in the limit the two definitions of W converge. Note that since (2.7) has no inequality constraints there is no need to solve a QP at each iteration. The merit function used in the MP method is

$$M(x, y, \rho) = f(x) - y^T c(x) + \tfrac{1}{2}\rho \|c(x)\|_2^2,$$

which is the same as that used in SNOPT [GMS98] and NPSOL. In MP $f(x)$ is the objective function since we apply this method to the barrier subproblem the merit

function used in MELBA replaces $f(x)$ with the barrier objective $f_B(x, \mu)$.

It seems particularly appropriate to use a merit function that is defined in terms of primal and dual variables when a search direction in these variables has been determined. A common merit function used in SQP methods is

$$M(x) = f(x) + \rho\|c(x)\|_1.$$

However, this merit function is not differentiable and is therefore unsuitable for use with directions of negative curvature. The use of such directions is necessary for an algorithm to generate iterates converging to second-order KKT points.

The termination conditions used are a relaxation of the transformed KKT conditions (2.9). Specifically,

$$\begin{aligned} \|c(x)\|_\infty &\leq \tau_1\mu \\ \|J^T y + z - g(x)\|_\infty &\leq \tau_2\mu \\ \|\mathbf{X}z - \mu e\|_\infty &\leq \tau_3\mu \\ \lambda_{\min}(Z^T W Z) &> -\tau_4\mu. \end{aligned} \tag{2.12}$$

The values of $\tau_i \geq 0$ and typically about 10. Note that these termination conditions differ from those typically used in algorithms to solve equality constrained problems. Typically the reduced gradient is used but since in the case of barrier functions the gradient in the neighborhood of the solutions is unbounded it is not a good measure of proximity of an iterate to the solution.

Algorithm 2.1 specifies a prototype barrier algorithm for solving the sequence of nonlinear equality optimization problems.

Data : x_0, y_0, z_0, $\{\mu_1, \ldots \mu_S\}$
Result : x^*, y^*, z^*

Initialization

$k = 0$

for $s = 1, \ldots, S$ **do**

  $\mu \leftarrow \mu_s$

  Call **Factorize(x_k, y_k, μ, funObj, funCon, funHess)**

  while *termination criteria not satisfied for* μ **do**

    Call **Primal-Dual** to compute $\Delta x, \Delta y, \Delta z, dn$

    Call **Linesearch** to calculate steplength, α_x

    Call **DualLinesearch** to calculate steplength, α_{yz}

    $x_{k+1} \leftarrow x_k + \alpha_x^2 \Delta x + \alpha_x dn$

    $y_{k+1} \leftarrow y_k + \alpha_{yz} \Delta y$

    $z_{k+1} \leftarrow z_k + \alpha_{yz} \Delta z$

    $k = k + 1$

    Call **Factorize(x_k, y_k, μ, funObj, funCon, funHess)**

Algorithm 2.1: Prototype algorithm

In Algorithm 2.1 dn is a direction of negative curvature for the matrix W, satisfying $dn^T W dn < 0$. Note the direction of negative curvature applies only in the primal space. Also for small α this direction dominates.

All routines or functions are discussed in subsequent chapters. The routine **Factorize** performs the major linear algebra operations and is discussed in chapter 3. The search direction $(\Delta x, \Delta y, \Delta z)$ and direction of negative curvature (dn) are computed in the function **Primal-Dual** and discussed in chapter 4. Chapter 5 discusses both the **Linesearch** and **DualLinesearch** functions, which compute the step length for each new search direction.

For clarity, error checking and error handling, although part of the design and implementation, are not included in the prototype algorithm.

Chapter 3

Barrier Method Factorization

This chapter discusses the function **Factorize**, as initially defined in Algorithm 2.1. **Factorize** is called when μ is changed, x_k, y_k, z_k is changed, or both.

Factorize is the backbone of our barrier method, yet it is independent of the barrier function or the primal-dual approach. This routine enables us to solve large-scale systems by performing factorization and matrix storage based upon proven methods used in other large-scale solvers. The solvers and references are included in the detailed discussion of each function called in **Factorize**.

The term factorize applies to the factorization of two matrices: a basis for the sparse Jacobian of the constraints, $B(x) \in \mathbb{R}^{m \times m}, B \equiv LU$, and the dense modified reduced Hessian of the Lagrangian function for the reformulated barrier problem, $\overline{Z(x)^T W(x,y,z) Z(x)} \equiv LDL^T = Z(x)^T W(x,y,z) Z(x) + E$. For our implementation, the Jacobian matrix is sparse but the reduced Hessian of the Lagrangian is a dense matrix. As a result, our implementation is best suited to problems where $n - m$ is not too large.

Factorize includes a partitioning of the Jacobian J and computation of Z, a basis for the null space. The Jacobian is partitioned into two parts such that $JP = \begin{pmatrix} B & S \end{pmatrix}$, where P is a permutation, $S \in \mathbb{R}^{m \times n-m}$, and B is nonsingular. The null space Z is stored implicitly as a function of B and S. The **Factorize** algorithm is outlined in figure 3.1.

Data : x, y, **funCon, funHess**
Result : $L, U, S, P_{col}, P_{row}, L_{chol}, D, E, d$

1 Partition J, factorize B and define Z
Call $[P_{row}, P_{col}, L, U, S]$ = **ComputeNullSpace(**x**,** y**, funCon)**

2 Compute and factorize the reduced Hessian of the Lagrangian
Compute $Z^T W Z$
Factorize $Z^T W Z + E = L_{chol} D L_{chol}^T$, also returns d

Algorithm 3.1: Factorize algorithm

The two main components of this function are discussed in the following sections. The first section discusses the computation of B, S and Z as well as the factorization of B. The second section explains how $\overline{Z^T W Z}$ is computed and factorized.

3.1 Computing the null space

The function **ComputeNullSpace** permutes the Jacobian J into two parts, $B(x)$ and $S(x)$, where $JP_{col} = \begin{pmatrix} B & S \end{pmatrix}$. It also computes the LU factors of B and uses this factorization to give implicit access to $Z(x)$, the nullspace of the Jacobian.

ComputeNullSpace is based on the approach used in MINOS [MS78] and SNOPT [GMS02]. The choice of B must take into account the term $\text{diag}(\mathbf{X}^{-1}z)$ in $W(x, y, z)$. We favor columns of J associated with large values of x. We define a sequence of threshold values aiming to determine a basis for the range space of J without values of x close to 0. For example, let $bigX$ = [1 0.1 0.01 0.001]. The first time through our loop, we select only the columns of J for which $x_j > 1$. The resulting matrix is J_r and we define $P_r \in \mathbb{R}^{n \times n}$ such that $JP_r = \begin{pmatrix} J_r & K \end{pmatrix}$, where K contains the remaining columns of J. Next, we estimate whether J_r has full rank and if not, reset J_r to contain the columns of J for which $x_j > 0.1$. This continues until J_r has full rank or is the full J.

Given $J_r \in \mathbb{R}^{m \times r}$ and $m \leq r$, we perform an LU factorization on its transpose to find a basis of the columns of J_r. Since the Jacobian is sparse, we use MATLAB's sparse **lu** solver on J_r^T. This sparse solver combines a column pre-ordering strategy with a right-looking unsymmetric-pattern multifrontal numerical factorization [Dav02]. The routine returns four matrices, $\hat{L}, \hat{U}, \hat{P}, \hat{Q}$ such that $\hat{P} J_r^T \hat{Q} = \hat{L}\hat{U}$, where $\hat{L}$ is nonsingular (and well-conditioned in general) and $\hat{U} = \begin{pmatrix} \tilde{U} & 0 \end{pmatrix}^T$. We next create the permutation

matrix

$$P_1 = \begin{pmatrix} \hat{P} & 0 \\ 0 & I \end{pmatrix} \in \mathbb{R}^{n \times n}$$

and define $P_{col} = P_r P_1^T$. This gives the partition $JP_{col} = \begin{pmatrix} B & S \end{pmatrix}$ with B nonsingular. We can now define the nullspace of $J \in \mathbb{R}^{m \times n}, m < n$, as the following $Z \in \mathbb{R}^{n \times (n-m)}$:

$$Z = P_{col} \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix}.$$

This Z is a basis for the nullspace of J because it has full column rank and

$$JZ = JP_{col} \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix} = \begin{pmatrix} B & S \end{pmatrix} \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix} = 0.$$

The final step is to factorize B, again using MATLAB's **lu**. This returns matrices P_{row}, Q, L, U such that $P_{row}BQ = LU$. All operations using these matrices are optimized in utility functions (e.g., **computeZg**, **solveZg**).

The routine is summarized in Algorithm 3.2. In the test for the rank of J_r, δ is a small value, such as 10^{-8}.

Data : x, y, μ, $bigX$, **funCon**
Result : $L, U, S, P_{row}, P_{col}$

1 Get user-defined J, the Jacobian of the constraints
  Call $J =$ **funCon(**x**,** y**)**

2 Get column permutation of J to give us a full rank basis, B
  for $minX$ *in* $bigX$ **do**

3 Select columns of J associated with large x_j
  Set $J_r = J(:, \text{find}(x > minX))$
  Set P_r, such that $JP_r = \begin{pmatrix} J_r & K \end{pmatrix}$
  Use sparse LU solver
    $[\hat{L}, \hat{U}, \hat{P}, \hat{Q}] =$ **lu(**J_r^T**)**
  If tentative basis B has full rank, then we are done
    if $\min(\text{abs}(\text{diag}(\hat{U}))) > \delta$ **then**
      break
  Set $\hat{U} = \hat{U} * \hat{Q}^T$

4 From S explicitly and factorize B
  Set $P_{col} = P_r \begin{pmatrix} \hat{P} & 0 \\ 0 & I \end{pmatrix}^T$
  Set $sortJ = JP_{col}$
  Set $[L, U, P_{row}, Q] =$ **lu(**$sortJ(:, 1:m)$**)**
  Set $U \leftarrow U * Q^T$
  Set $S = sortJ(:, m+1:n)$

Algorithm 3.2: Compute null space

3.2 Computing the reduced Hessian of the Lagrangian

The reduced Hessian of the Lagrangian $Z(x)^T W(x,y) Z(x)$ is required to compute $\Delta x, \Delta y$ and Δz. The approach is based on the reduced-gradient method used in MINOS and SQOPT, as described in [MS78] and [GMS02].

The first step computes the matrix $Z^T W Z \equiv Z^T (H_L + \mathbf{X}^{-1}\mathbf{Z}) Z$.

The second step calls the function **modchol**. **Modchol** returns a factorization $LDL^T = Z^T W Z + E$, where the diagonal matrix E is added to the original matrix to obtain a positive definite matrix, $\overline{Z^T W Z}$. Using a positive definite matrix ensures that $(\Delta x, \Delta y, \Delta z)$ is a descent direction for the merit function $M(x, y, \rho)$ that is used

to determine the length of a step and is discussed in detail in Chapter 5.

The routine **modchol** is based on the *Modified Cholesky Factorization* algorithm defined in [GMW81]. The standard Cholesky factorization usually applies to a symmetric, positive definite matrix. For a dense matrix, it is the most efficient factorization. The modified Cholesky factorization is used for symmetric matrices that are *almost* symmetric positive definite. The diagonal modification E provides us with a factorization of a symmetric, positive definite matrix that is close to the original matrix. The intermediate matrix $\overline{Z^T W Z}$ is not retained, only its factorization.

Algorithm 3.3 is the algorithm for factorizing the reduced Hessian of the Lagrangian for the reformulated primal-dual problem.

Data : $L, U, S, P_{row}, P_{col}, x, y$
Result : $P_{chol}, L_{chol}, D, E, d$

1 Compute $Z^T H Z$
begin
  Compute $Z = P_{col} \begin{pmatrix} -B^{-1}S \\ I \end{pmatrix}$ from the factorization of B
  Solve $Z1 = -U \backslash (L \backslash (P_{row}^T S))$
  Rearrange
  $Z \leftarrow P_{col} \begin{pmatrix} Z1 \\ I_{n-m} \end{pmatrix}$
  Perform the matrix-vector multiplications to obtain HZ
  for $j = 1, 2, \ldots, n-m$ **do**
    $HZ(:, j) =$ **funHess(**$x, y, Z(:, j)$**), FALSE)**
2 Compute Hessian of Barrier Term * Z and compute $Z^T W Z$.
  begin
    $Barrier \leftarrow (\mathbf{X}^{-1} z) .* Z$
    $Z^T W Z = Z^T (HZ + Barrier)$
  end
end
3 Compute $L_{chol} D L_{chol}^T = P_{chol}^T (Z^T W Z) P_{chol} + E$, and d
begin
  $[P_{chol}, L_{chol}, D, E, d] =$ **modchol(**$Z^T W Z$**)**
end

Algorithm 3.3: Factorize the reduced Hessian of the Lagrangian

Chapter 4

Search Direction

In this chapter, we discuss the computation of the search direction for the primal (Δx) and dual ($\Delta y, \Delta z$) variables. We also define and explain the computation of the direction of negative curvature (d) and discuss the computation of the minimum eigenvalue of the reduced Hessian of the Lagrangian ($\lambda_{\min}$).

So far, we have considered a standard format for NLP (1.1) that has been adequate for explaining concepts and the **factorize** function. However, for search direction and step length, it is now necessary to include the use of upper and lower bounds. The format used in most large-scale solvers is

NLP1	$\underset{x}{\text{minimize}}$	$f(x)$
	subject to	$\ell \leq \begin{pmatrix} x \\ c(x) \end{pmatrix} \leq u.$

A variable vector $s \in \mathbb{R}^m$, termed the *slacks*, is used to reformulate the problem to

NLP2	$\underset{x}{\text{minimize}}$	$f(x)$
	subject to	$c(x) - s = 0, \quad \ell \leq \begin{pmatrix} x \\ s \end{pmatrix} \leq u.$

Since the slack variables are treated just like x in the primal-dual method, we take the liberty of implicitly implying the existence of s and use the problem definition

NLP3	$\underset{x}{\text{minimize}}$	$f(x)$
	subject to	$c(x) = 0, \quad \ell \leq x \leq u,$

where $f(x)$ and $c_i(x)$ are as defined for (1.1). For unbounded variables, the upper and lower bounds are set to a maximum or minimum bound.

4.1 Computing $\Delta x, \Delta y, \Delta z$

The search directions for the primal and dual variables are computed using modifications to the KKT system. The derivations are defined in section 4.1.1. The implementation is then discussed in section 4.1.2.

4.1.1 Derivations

BLA assumes all variables are non-negative. To convert the bounds to non-negativity constraints, we define additional slack variables $t_1, t_2 \in I\!R^n$:

$$\begin{array}{lll} \underset{x,t_1,t_2}{\text{minimize}} & f(x) & \\ & c(x) = 0 & \\ \text{subject to} & x - t_1 = \ell & \\ & x + t_2 = u & \\ & t_1, t_2 \geq 0. & \end{array}$$

Then, we replace the non-negativity constraints by the log barrier function, obtaining a sequence of equality-constrained subproblems with decreasing values of μ ($\mu > 0$):

$$\begin{array}{lll} \text{NLP}(\mu) \quad \underset{x,t_1,t_2}{\text{minimize}} & f(x) - \mu \sum_j \ln([t_1]_j [t_2]_j) & \\ & c(x) = 0 & : y \\ \text{subject to} & x - t_1 = \ell & : z_1 \\ & -x - t_2 = -u, & : z_2 \end{array}$$

where y, z_1, and z_2 denote dual variables for the associated constraints. With $\mu > 0$, the variables t_1, t_2, z_1, z_2 are strictly positive.

The KKT conditions for the barrier subproblem involve the three *primal* equations of NLP(μ), along with three *dual* equations stating that the gradient of the subproblem objective should be a linear combination of the gradients of the primal constraints:

$$\begin{array}{rll} c = & 0 & \\ x - t_1 = & \ell & \\ -x - t_2 = & -u & \\ J^T y + z_1 - z_2 = & g & : x \\ \mathbf{T_1} z_1 = & \mu e & : t_1 \\ \mathbf{T_2} z_2 = & \mu e & : t_2, \end{array}$$

where $\mathbf{T_1} = \text{diag}(t_1)$, $\mathbf{T_2} = \text{diag}(t_2)$, and similarly for $\mathbf{Z_1}$, $\mathbf{Z_2}$ later. In fact, the last

two equations arise in a different form. The dual equation for t_1 is actually

$$-z_1 = \nabla(-\mu \ln(t_1)) = -\mu \mathbf{T_1}^{-1} e,$$

where e is a vector of 1's. Thus, $t_1 > 0$ implies $z_1 > 0$, and multiplying by $-\mathbf{T_1}$ gives the equivalent equation $\mathbf{T_1} z_1 = \mu e$ as stated. In this form, the last two equations are commonly called (perturbed) *complementarity* conditions.

Newton's method

We now apply Newton's method, defining new primal directions as $\Delta x, \Delta t_1$ and Δt_2, which may later be combined with a direction of negative curvature, dn. The search directions for the dual variables are defined to be $\Delta y, \Delta z_1$ and Δz_2. The resulting system is

$$\begin{aligned}
J\Delta x &= -c \\
(x + \Delta x) - (t_1 + \Delta t_1) &= \ell \\
-(x + \Delta x) - (t_2 + \Delta t_2) &= -u \\
J^T(y + \Delta y) + (z_1 + \Delta z_1) - (z_2 + \Delta z_2) &= g + H_L \Delta x \\
\mathbf{T_1} z_1 + \mathbf{T_1} \Delta z_1 + \mathbf{Z_1} \Delta t_1 &= \mu e \\
\mathbf{T_2} z_2 + \mathbf{T_2} \Delta z_2 + \mathbf{Z_2} \Delta t_2 &= \mu e,
\end{aligned}$$

where c is the current constraint vector, g is the objective gradient and H_L is the Hessian of the Lagrangian. To solve this Newton system, we work with three sets of residuals:

$$\begin{pmatrix} \Delta x - \Delta t_1 \\ -\Delta x - \Delta t_2 \end{pmatrix} = \begin{pmatrix} r_\ell \\ r_u \end{pmatrix} \equiv \begin{pmatrix} \ell - x + t_1 \\ -u + x + t_2 \end{pmatrix} \tag{4.1}$$

$$\begin{pmatrix} \mathbf{T_1} \Delta z_1 + \mathbf{Z_1} \Delta t_1 \\ \mathbf{T_2} \Delta z_2 + \mathbf{Z_2} \Delta t_2 \end{pmatrix} = \begin{pmatrix} c_\ell \\ c_u \end{pmatrix} \equiv \begin{pmatrix} \mu e - \mathbf{T_1} z_1 \\ \mu e - \mathbf{T_2} z_2 \end{pmatrix} \tag{4.2}$$

$$\begin{pmatrix} J\Delta x \\ -H_L \Delta x + J^T \Delta y + \Delta z_1 - \Delta z_2 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \equiv \begin{pmatrix} -c \\ g - J^T y - z_1 + z_2 \end{pmatrix}. \tag{4.3}$$

We use (4.1) and (4.2) to replace two set of vectors in (4.3) with

$$\begin{pmatrix} \Delta t_1 \\ \Delta t_2 \end{pmatrix} = \begin{pmatrix} -r_\ell + \Delta x \\ -r_u - \Delta x \end{pmatrix}, \quad \begin{pmatrix} \Delta z_1 \\ \Delta z_2 \end{pmatrix} = \begin{pmatrix} \mathbf{T_1}^{-1}(c_\ell - \mathbf{Z_1} \Delta t_1) \\ \mathbf{T_2}^{-1}(c_u - \mathbf{Z_2} \Delta t_2) \end{pmatrix}. \tag{4.4}$$

Defining

$$\begin{aligned} W &\equiv H_L + \mathbf{T_1}^{-1}\mathbf{Z_1} + \mathbf{T_2}^{-1}\mathbf{Z_2} && (4.5)\\ w &\equiv r_2 - \mathbf{T_1}^{-1}(c_\ell + \mathbf{Z_1} r_\ell) + \mathbf{T_2}^{-1}(c_u + \mathbf{Z_2} r_u), && (4.6) \end{aligned}$$

we find that Δx and Δy satisfy the KKT-type system

$$\begin{pmatrix} -W & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} w \\ r_1 \end{pmatrix}. \qquad (4.7)$$

This is a sparse linear system that may be solved by many possible direct or iterative methods. The efficiency of the chosen method largely determines the efficiency of the complete barrier method.

In MELBA, we have chosen to use a reduced Hessian method, based on a sparse representation of the null space of J. This allows us to transform equation (4.7) to block triangular form.

Given a permutation matrix P such that

$$JP_{col} = \begin{pmatrix} B & S \end{pmatrix},$$

where $B \in I\!R^{m\times m}$, $S \in I\!R^{m\times n-m}$ and $\mathrm{rank}(B) = m$, define matrices $Y \in I\!R^{n\times m}$ and $Z \in I\!R^{n\times(n-m)}$, as well as the vectors $\Delta x_Y \in I\!R^m$ and $\Delta x_Z \in I\!R^{n-m}$ such that

$$\begin{aligned} JY &= B \\ JZ &= 0 \\ Y &= P_{col}\begin{pmatrix} I \\ 0 \end{pmatrix} \\ Z &= P_{col}\begin{pmatrix} -B^{-1}S \\ I_{n-m} \end{pmatrix} \\ \Delta x &= \begin{pmatrix} Y & Z \end{pmatrix}\begin{pmatrix} \Delta x_Y \\ \Delta x_Z \end{pmatrix}. \end{aligned}$$

Finally, define the matrix

$$T = \begin{pmatrix} Y & Z & \\ & & I \end{pmatrix}.$$

Premultiplying (4.7) by T^T and substituting the above definitions, we obtain the block

triangular system

$$\begin{pmatrix} -Y^TWY & -Y^TWZ & B^T \\ -Z^TWY & -Z^TWZ & \\ B & & \end{pmatrix} \begin{pmatrix} \Delta x_Y \\ \Delta x_Z \\ \Delta y \end{pmatrix} = \begin{pmatrix} Y^Tw \\ Z^Tw \\ r_1 \end{pmatrix}.$$

We can now obtain Δx_Z and Δx_Y by solving

$$B\Delta x_Y = r_1 \tag{4.8}$$

$$Z^TWZ\Delta x_Z = -Z^T(w + WY\Delta x_Y). \tag{4.9}$$

We can then form $\Delta x = Y\Delta x_Y + Z\Delta x_Z$. Finally, we can solve for Δy by solving the system

$$B^T\Delta y = Y^T(w + W\Delta x). \tag{4.10}$$

4.1.2 Primal and dual search direction implementation

The top-level algorithm for determining the search directions is given in Algorithm 4.1,

Data : $L, U, S, P_{col}, P_{row}, P_{chol}, d$, **funHess**, **funObj**, x, t, y, z, μ
Result : $\Delta x, \Delta y, \Delta z_1, \Delta z_2, \Delta t_1, \Delta t_2, dn, \lambda_{\min}$

1 Compute w (4.6), W (4.5) and r_u, r_l (4.1), c_u, c_l (4.2), r_1, r_2 (4.3)

2 Call $[\Delta x, \Delta t_1, \Delta t_2]$ = **ComputePrimalDirection**

3 Call $[\Delta y, \Delta z]$ = **ComputeDualDirection**

4 Compute direction of negative curvature dn

Algorithm 4.1: Compute search direction

The primal directions

The computation of the primal direction is a straightforward implementation of the derivations in Section 4.1.1. However, there are two things to note. First, we do not show the use of the factorize routines (e.g., $B = P_{row}^T LU$) in these derivations. Also, and much more interesting, is the computation of the matrix W. We have a positive definite reduced Hessian from **Factorize**, $\overline{Z^TWZ} = L_{chol}DL_{chol}^T$. In order to guarantee that the same Hessian (or altered Hessian) used in all computations, we compute the matrix $\overline{W}$ where

$$\overline{Z^TWZ} = Z^TWZ + E.$$

Given the definitions of the nullspace of the Jacobian, this $\overline{W}$ is easily computed as shown in the following theorem.

Theorem 4.1. *For $W \in I\!R^{n\times n}$, $Z \in I\!R^{n\times(n-m)}$, where Z is defined for a Jacobian matrix of the constraints $J \in I\!R^{m\times n}$ with $JP_{col} = \begin{pmatrix} B & S \end{pmatrix}$, P_{col} a permutation, B nonsingular, $JZ = 0$ and $Z = P_{col}\begin{pmatrix} -B^{-1}S \\ I \end{pmatrix}$, then given the modified Cholesky factorization $Z^TWZ + E = L_{chol}DL_{chol}^T$, the matrix $\overline{W}$ such that $\overline{Z^TWZ} = L_{chol}DL_{chol}^T$ is $\overline{W} = W + P_{col}\begin{pmatrix} 0 & 0 \\ 0 & E \end{pmatrix}P_{col}^T$.*

Proof. For the specified $\overline{W}$,

$$\begin{aligned} Z^T\overline{W}Z &= Z^TWZ + Z^TP_{chol}\begin{pmatrix} 0 & 0 \\ 0 & E \end{pmatrix}P_{chol}^TZ \\ &= Z^TWZ + \begin{pmatrix} (-B^{-1}S)^T & I \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 0 & E \end{pmatrix}\begin{pmatrix} -B^{-1}S \\ I \end{pmatrix} \\ &= Z^TWZ + E \\ &= L_{chol}DL_{chol}^T. \end{aligned}$$

∎

This theorem shows that by choosing the permutation P_{col} such that B is nonsingular we are able to obtain the modification to W that gives us the modified reduced matrix. A related result is that choosing the permutation P_{col} in a more careful way shows that the reduced Hessian is ill-conditioned only by having an ill-conditioned diagonal matrix as shown in the following theorem.

Theorem 4.2. *Consider $Z \in I\!R^{n\times(n-m)}$, where Z is defined as a nullspace for the Jacobian J, and $JP_{col} = \begin{pmatrix} B & S \end{pmatrix}$. When Z is defined as*

$$Z = P_{col}\begin{pmatrix} -B^{-1}S \\ I \end{pmatrix},$$

the number of i's where $x_i^* = 0$ is less than or equal to $n - m$, and the rank of $B = m$, then the ill-conditioning due to the barrier terms is kept to the diagonal of the reduced Hessian of the Lagrangian.

Proof. For simplicity we revert to problem (2.7) with bounds $x \geq 0$. The reduced Hessian of the Lagrangian is

$$Z^T(H_L + \mathbf{X}^{-1}\mathbf{Z})Z = Z^TH_LZ + Z^T\mathbf{X}^{-1}\mathbf{Z}Z,$$

where ill-conditioning due to the barrier term occurs only in the last term. Using our definition of Z,

$$Z^T\mathbf{X}^{-1}\mathbf{Z}Z \;=\; \begin{pmatrix} -S^TB^{-T} & I \end{pmatrix}\mathbf{X}^{-1}\mathbf{Z}\begin{pmatrix} -B^{-1}S \\ I \end{pmatrix}.$$

Let $Y = -B^{-1}S$ and $M = P_{col}^T\mathbf{X}^{-1}\mathbf{Z}P_{col}$. Then M is a diagonal matrix in which small x values are in the lower right of the matrix. Let

$$M = \begin{pmatrix} M_{11} & 0 \\ 0 & M_{22} \end{pmatrix},$$

where the columns of M_{22} correspond to all x_j's where $x_j^* = 0$. We now have

$$\begin{pmatrix} Y^T & I \end{pmatrix} M \begin{pmatrix} Y \\ I \end{pmatrix} \;=\; Y^TM_{11}Y + M_{22},$$

where M_{22} is a diagonal matrix containing the ill-conditioned x values. ∎

As a result of this theorem, when there are only a few $x^* = 0$ at the solution, we can compute an accurate solution regardless of the ill-conditioning.

Algorithm 4.2 details the algorithm for computing the primal search directions $\Delta x, \Delta t_1, \Delta t_2$.

Data : $L, U, P_{col}, P_{row}, P_{chol}, L_{chol}, D, r_\ell, r_u, w, \overline{W}$
Result : $\Delta x, \Delta t_1, \Delta t_2$

1 Compute Δx
begin
  Solve $B\Delta x_Y = -c(x)$ for Δx_Y, see (4.8)
  Solve $\overline{Z^TWZ}\Delta x_Z = -Z^Tw - Z^T\overline{W}Y\Delta x_Y$ for Δx_Z, see (4.9)
  Set $\Delta x = Y\Delta x_Y + Z\Delta x_Z$
end
2 Compute $\Delta t_1, \Delta t_2$
begin
  Set $\Delta t_1 = -r_\ell + \Delta x$, see (4.4)
  Set $\Delta t_2 = -r_u - \Delta x$
end

Algorithm 4.2: Compute primal directions

The dual directions

The algorithm for the computation of the dual direction is given in Algorithm 4.3, which is a direct implementation of the derivations in section 4.1.1.

Data : $L, U, S, P_{col}, P_{row}, \overline{W},\ y, z_1, z_2, t_1, t_2, c_\ell, c_u, \Delta t_1, \Delta t_2,\ \Delta x, w$
Result : $\Delta y, \Delta z_1, \Delta z_2$

1 Compute $\Delta z_1, \Delta z_2$
begin
  Solve $\mathbf{T_1}\Delta z_1 = c_\ell - \mathbf{Z_1}\Delta t_1$ for Δz_1, see (4.4)
  Solve $\mathbf{T_2}\Delta z_2 = c_u - \mathbf{Z_2}\Delta t_2$ for Δz_2
end
2 Compute Δy
begin
  Solve $B^T\Delta y = Y^T(w + \overline{W}\Delta x)$ for Δy, see (4.10)
end

Algorithm 4.3: Compute dual directions

4.2 Direction of negative curvature

In this section we discuss computation of the directions of negative curvature dn and α for Z^TWZ and W, and also the minimum eigenvalue of Z^TWZ ($\lambda_{\min}$). The eigenvalue is not part of the search direction algorithm, but is included here because it is computed as one of the steps for refining d.

4.2.1 Why a direction of negative curvature is needed

A direction of negative curvature for Z^TWZ is any vector $d \in \mathbb{R}^{n-m}$ that satisfies $d^T(Z^TWZ)d < 0$. In order to guarantee convergence to a second-order point it is essential that a direction of negative curvature be used. For example, consider the problem

$$\begin{aligned} &\min && x^3 \\ &\text{subject to} && x \geq -2. \end{aligned}$$

If the current iterate is $x = 0$ the first-order optimality conditions are satisfied, but the second-order optimality conditions are not. Generating the next search direction from the KKT system at $x = 0$ gives $\Delta x = 0$ and no progress can be made. Without d, the algorithm can not converge to $x^* = -2$.

Using d may also improve efficiency even when d is not essential for convergence. Assume x represents all primal variables and y are the dual variables. After a descent direction is computed, a typical Newton method performs a linesearch to find a step length α_k that reduces some merit function $M(x, y)$ at the point

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} + \alpha_k \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}.$$

Murray and Prieto [MP95] propose an alternative computation for x_{k+1}, y_{k+1} using dn. They define a curvilinear search as a procedure that computes a step length α_k such that

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} + \alpha_k \begin{pmatrix} dn \\ 0 \end{pmatrix} + \alpha_k^2 \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

reduces the merit function. They show that fewer iterations may be needed to satisfy the convergence criteria using directions of negative curvature rather than descent directions alone. This method, and its theory, has been successfully applied to SQP methods.

Moguerza and Prieto [MP03] analyzed the use of negative curvature for a primal-dual method using a log-barrier function. In 20% of their test cases, the total iterations decreased when negative curvature was combined with descent directions to determine the next iterates. In no test case did the total iterations increase.

The benefits of using negative curvature can also be seen in the example

$$\begin{array}{ll} \min_{x_1,x_2} & -x_1^2 + x_2^2 \\ \text{subject to} & 0 \le x_1 \le 1. \end{array}$$

This problem illustrates how using negative curvature results in a larger reduction in the objective function value than using the direction of descent alone. There is one solution $(1, 0)$ with Lagrange multiplier $\lambda = -2$ satisfying the second-order KKT conditions for inequality constrained problems.

Consider a starting point of $(0.5,\ 0.5)$. The Newton descent direction is found by solving $H_L p = -g$, where $p = (-2\Delta x_1,\ 2\Delta x_2)$. The direction of negative curvature is obtained from the eigenvector corresponding to the smallest eigenvalue of H_L. Figure 4.1 shows the contour for the objective function values, along with the direction of negative curvature, descent direction and the combined direction vectors. We see that the combined direction has a lower objective function value at the boundary and converges to the solution in one step.

The routine **modchol** computes the modified Cholesky factorization of the reduced Hessian $Z^T W Z$ and gives us a direction of negative curvature for $Z^T W Z$ with minimal computation costs. This direction of negative curvature, dn, is an estimate for the

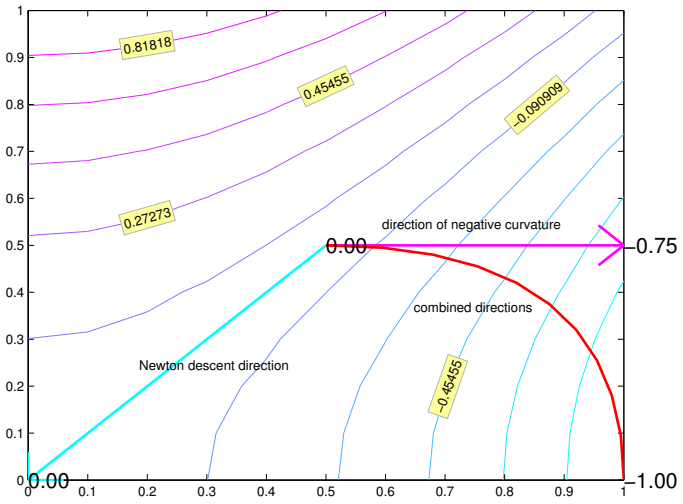

FIGURE 4.1 *Direction of negative curvature example*

eigenvector corresponding to the smallest eigenvalue of $Z^T W Z$.

Since the cost of computing d for $Z^T W Z$ is negligible and the convergence rate is generally improved, negative curvature is combined with the computed descent direction.

There are three parts to computing a direction of negative curvature. First, we update the direction d returned from the modified Cholesky algorithm. In section 4.2.2 we discuss this and the computation of $\lambda_{\min}$. After we solve for the primal direction, we extend d to $\mathbb{R}^n$ to obtain dn, and scale it to the same size as the primal direction using the 2-norm. Finally, we verify that dn is a direction of negative curvature for $H_L(x,y)+\mu \mathbf{X}^{-2}$ and if not, we discard it. This functionality is discussed in section 4.2.3.

4.2.2 Estimating the minimum eigenvalue and refining dn

The minimum eigenvalue of $Z^T W Z$ is necessary to demonstrate convergence of our sequence to second-order KKT points and to decide whether or not to use a direction of negative curvature d when it exists. When the minimum eigenvalue $\lambda_{min} > 0$ then $Z^T W Z \succ 0$, which is a condition for convergence to second-order KKT points (see section 1.3).

Given a symmetric matrix A, the modified Cholesky algorithm produces the factorization

$$A + E = L_{chol} D L_{chol}^T,$$

where $E \succeq 0$ is diagonal. When A is positive definite, $E = 0$ and there is no direction of negative curvature. When $E \neq 0$ we find w such that $A_{ww} = \min_j A_{jj}$. With e_w the

w^{th} unit vector, we then solve

$$L^T d = e_w.$$

The resulting d is often a direction of negative curvature. It approximates the eigenvector corresponding to $\lambda_{\min}$, the smallest eigenvalue of A [GMW81], and its Rayleigh quotient estimates λ_{min} itself [GV83].

In MELBA, A is $Z^T W Z$ and the estimate of $\lambda_{\min}$ is used to determine whether to use a direction of negative curvature. The estimate can be somewhat imprecise, indicating only whether or not we have a positive definite matrix. We consider negative curvature when our estimate satisfies $\lambda_{\min} < -\delta$ for some $\delta > 0$.

In estimating $\lambda_{\min}$ we have two issues to consider. First, how accurate does this eigenvalue need to be? Then, given the accuracy required, what is the most efficient way to obtain this value?

Moré and Sorensen [MS79] prove that the modified Cholesky algorithm has difficulty estimating directions of negative curvature when the minimum eigenvalue is close to 0. This is not an issue here because we only consider the direction of negative curvature when it is less than a small negative value ($\lambda_{\min} < -\delta$). If $\lambda_{\min}$ is negative, but close to 0, then the direction of negative curvature is discarded. This feature is a benefit as we get closer to the solution because $Z^T W Z$ may fluctuate between positive definiteness and indefiniteness because of roundoff errors in computing the products with Z and Z^T. If we do not eliminate small directions of negative curvature then there may be convergence rate difficulties. Using an upper bound on the minimum eigenvalue is a reasonable approach since the reduced Hessian $Z(x^*)^T H_L(x^*, y^*) Z(x^*)$ must be positive definite at the solution. By continuity, as $(x_k, y_k) \rightarrow (x^*, y^*)$ the reduced Hessian of the Lagrangian should also be positive definite and any fluctuation is due to rounding errors. Ignoring erroneous and small negative eigenvalues allows for quadratic convergence near the solution.

Given our direction of negative curvature (d computed in the modified Cholesky function), we estimate the corresponding minimum eigenvalue using the Rayleigh quotient [GV83]. Let $A \equiv \overline{Z^T W Z}$, with approximate eigenvector d. To find the corresponding eigenvalue $\lambda_{\min}$, we minimize $\|(A - \lambda_{\min} I)d\|_2$ with respect to $\lambda_{\min}$. The solution to this problem is called the Rayleigh quotient:

$$\lambda_{\min} \quad = \quad \frac{d^T A d}{d^T d}.$$

Using this equation we can improve both d and $\lambda_{\min}$ at a cost of $n - m$ operations. We perform a univariate search along each component of d and determine if a small change reduces λ_{min}. If it does, $\lambda_{\min}$ is set to this value and d_k is updated to $d + \alpha e_k$.

For each k, the univariate search solves the minimization problem

$$\min_{\alpha} r(\alpha) = \frac{(d + \alpha e_k)^T A (d + \alpha e_k)^T}{\|d + \alpha e_k\|^2}.$$

This requires us to find the roots of a quadratic and test both solutions to see if either reduces the eigenvalue estimate. If neither decreases $\lambda_{\min}$, we simply discard the change [Bom99].

The algorithm for estimating the minimum eigenvalue and improving d is detailed in Algorithm 4.4.

Data : d, A
Result : $\lambda_{\min}, d$

Univariate Search to improve eigenvalue estimate
if *A is not positive definite* **then**
  Obtain estimate for minimum eigenvalue
  Set $\lambda_{\min} = \frac{d^T A d}{\|d\|_2^2}$
  Perform univariate search to improve $\lambda_{\min}$ and d estimate.
  for $k = 1, 2, \ldots, n$ **do**
    Compute $\lambda_{temp} = \min_{\alpha} \frac{(d+\alpha e_k)^T A (d+\alpha e_k)}{\|(d+\alpha e_k)\|_2^2}$ and $\alpha_{\min}$ = optimal α
  if $\lambda_{temp} < \lambda_{\min}$ **then**
    $d(k) \leftarrow d(k) + \alpha_{\min}$
    $\lambda_{\min} \leftarrow \lambda_{temp}$

Algorithm 4.4: Compute minimum eigenvalue and negative curvature

4.2.3 Extending d to $\mathbb{R}^n$, scaling and validation

In the linesearch routines, the search direction is combined with the direction of negative curvature. The search direction is in $\mathbb{R}^n$ and $d \in \mathbb{R}^{n-m}$. Therefore, d is extended to $\mathbb{R}^n$ as follows.

The direction of negative curvature d satisfies $d^T Z^T W Z d < 0$. Defining $dn = Zd$, we see that

$$dn^T W dn = (d^T Z^T) W (Zd) < 0$$

and, hence, dn is a direction of negative curvature for W as required.

To prevent difficulties when combining dn with Δx, the direction of descent, we scale dn so that $\|dn\|_2 = \|\Delta x\|_2$ [MP03].

The last step is to check that dn is a direction of negative curvature for the original

barrier subproblem. Note that dn has been computed for the matrix

$$W = H_L + \mathbf{T_1}^{-1}\mathbf{Z_1} + \mathbf{T_2}^{-1}\mathbf{Z_2},$$

which is derived from the primal-dual equations. The Hessian of the Lagrangian for the barrier subproblem in section 4.1.1 is actually

$$W_B = H_L + \mu\left(\mathbf{T_1}^{-2} + \mathbf{T_2}^{-2}\right).$$

We therefore test if $dn^T W_B dn < 0$, which is equivalent to testing if $d^T Z^T W_B Z d < 0$. If not, then dn is not a valid direction of negative curvature and it is discarded.

Algorithm 4.5 defines the computation of the direction of negative curvature. The implementation is spread out in different routines, but presented here as one algorithm for clarity.

Data : d, $\Delta x, \Delta t_1, \Delta t_2, \lambda_{\min}$
Result : dn

1 If min eigenvalue is small enough, disregard direction of negative curvature
if $\lambda_{\min} >$ MINEIG **then**
  $dn = 0$
  return

2 Verify that d is a direction of negative curvature for the
reduced Hessian of the Lagrangian for the original barrier subproblem
Set $A = Z^T\left(H_L + \mu\left(\mathbf{T_1}^{-2} + \mathbf{T_2}^{-2}\right)\right)Z$
if $d^T A d < -\epsilon$ **then**
3 Extend d to $\mathbb{R}^n$: $dn \leftarrow Z \times d$
4 Scale dn to be the same size in norm as Δx
    $dn \leftarrow \frac{dn}{\|dn\|}\|\Delta x\|$
else
  Discard d
  Set $dn \leftarrow 0$

Algorithm 4.5: Compute the direction of negative curvature algorithm

Chapter 5

Step Size

In this chapter we discuss the computation of the step size to be taken along the linear and curvilinear search directions. It is well known that one of these needs to be exercised when performing a linesearch for a barrier function [GMW81]. We return to the problem formulation (1.1):

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) & \\ \text{subject to} & c_i(x) = 0 & i = 1, 2, \ldots, m \\ & x_j \geq 0 & j = 1, 2, \ldots, n, \end{array}$$

where $f(x)$ is a function with known gradient $g(x)$ and Hessian $H(x)$, and $c(x) \in \mathbb{R}^m$ is the vector of $c_i(x)$ functions with known Jacobian $J(x) \in \mathbb{R}^{m\times n}$ and Hessians $H_i(x)$. The barrier subproblem formulation is

$$\begin{array}{ll} \min_x & f(x) - \mu \sum_{j=1}^{n} \ln x_j \\ \text{subject to} & c(x) = \quad 0. \end{array} \tag{5.1}$$

A linesearch method determines an $\alpha > 0$ (usually $\alpha \in (0, 1]$) that reduces a predefined merit function $M(x + \alpha\Delta x, y + \alpha\Delta y)$ for search directions $(\Delta x, \Delta y)$. Close to the solution, we expect the full Newton step ($\alpha = 1$) to be returned by the linesearch routine.

This chapter is in four sections. The first section discusses the merit function used and gives an overview of the algorithm. In subsequent sections we discuss the different step size algorithms that have been implemented.

5.1 Merit function

The term *Merit Function* is used to denote a scalar function $M(x, y, \rho)$ that enables iterates to be ranked. For linearly constrained minimization, the merit function is usually the objective function $f(x)$ itself because sequences $\{x_k\}_{k=0}^{\infty}$ can be generated with all points x_k feasible.

For nonlinearly constrained problems, feasibility for all x_k can not be guaranteed and deciding if one point is 'better' than another is problematic. Figure 5.1 is a plot of problem (2.4) for three points X, Y and Z. As the plot shows, it is not clear which

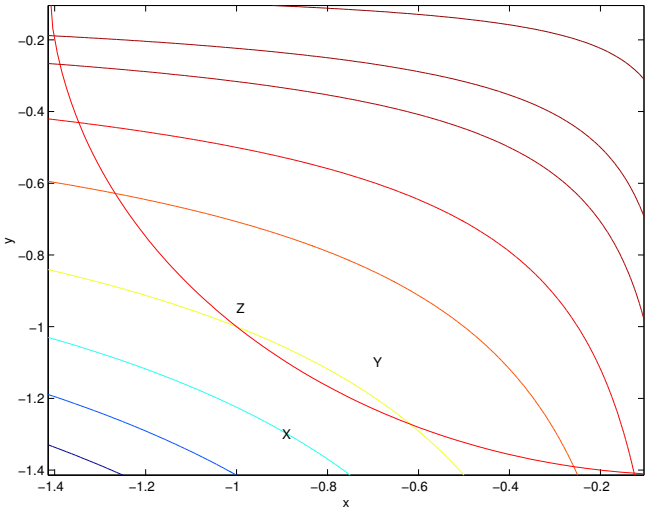

FIGURE 5.1 *Deciding which value X, Y, Z is 'better' is difficult*

value is 'better'. A merit function provides us with a numerical value that enables us to rank X, Y and Z.

We use a modification to the merit function defined by Eldersveld [Eld91] with one penalty parameter per constraint:

$$M(x, y, \rho) = f(x) - y^T c + \tfrac{1}{2} c^T R c,$$

where $R = \text{diag}(\rho_i),\ i = 1, \ldots, m$. The augmented Lagrangian function for problem (5.1) is a bit more complicated because it must include barrier terms:

$$M(x, y, \mu, \rho) = f_B(x, \mu) - y^T c + \tfrac{1}{2} c^T R c, \tag{5.2}$$

where ρ is the vector of ρ_i's and $f_B(x, \mu) = f(x) - \mu \sum_{j=1}^{n} \ln x_j$. When there is a direction of negative curvature, a curvilinear search is performed along the arc $(\alpha^2 \Delta x + \alpha \Delta x, \alpha^2 \Delta y)$ and step length α is determined that solves the problem

$$\min_{\alpha} M(x + \alpha^2 \Delta x + \alpha dn, y + \alpha \Delta y, \mu, \rho).$$

Otherwise, a linesearch is performed and α satisfies

$$\min_{\alpha} M(x + \alpha \Delta x, y + \alpha \Delta y, \mu, \rho).$$

The algorithm for computing the step size is outlined in Algorithm 5.1. The functionality for this procedure is discussed in Sections 5.2 and 5.3. Finally, a separate linesearch is done for the dual variables and the procedure for this is discussed in Sec-

tion 5.4.

Data : $E, L, U, S, P_{chol}, L_{chol}, E$, **funHess, funCon, funObj,** $x, y, z, \Delta x, \Delta y, \Delta z$
Result : α, α_{yz}

$\alpha = 0$
if *There exists a sufficient direction of negative curvature* **then**
  Call $\alpha =$ **Curvilinear search**

if $\alpha = 0$ **then**
  Call $\alpha =$ **Safeguarded linear search**

Call $\alpha_{yz} =$ **Dual linesearch**

Algorithm 5.1: Compute step size

The safeguarded linesearch is presented first because some of its functionality is used in the curvilinear search.

5.2 Safeguarded linear search

The safeguarded linesearch is based on the linear search described in Lin's thesis [Lin02]. This linesearch method is called when there is not a sufficient direction of negative curvature. As discussed earlier, the goal is to find an α that minimizes the augmented Lagrangian merit function (5.2). Determining α is a minimization problem in its own right and we want an approach that is efficient computationally, while still being accurate.

One approach is the Armijo or backtracking linesearch. We start with $\alpha = 1$ and test if it satisfies our linesearch termination criterion. If it does, we return. If not, α is reduced by a fixed factor ω. Then we compute the new objective and constraint function values and calculate the new merit function value. We continue until we satisfy the termination criterion. The backtracking linesearch can be quite expensive because each trial α requires calls to the user-defined objective and constraint functions. This approach uses very little of the problem information, using $\psi(\alpha)$ only to compare the two values for α.

A more efficient procedure for a linesearch, using more information from the function, is to approximate the univariate function $\psi(\alpha)$ using a simple function whose minimum is easily computed. Since such a procedure may diverge if the approximation is inaccurate, an interval of uncertainty $[l, h]$ is maintained as a safeguard. The most common interpolation functions are polynomials. Murray and Wright [MW94] discuss the problems of using polynomial interpolation for estimating a barrier function and

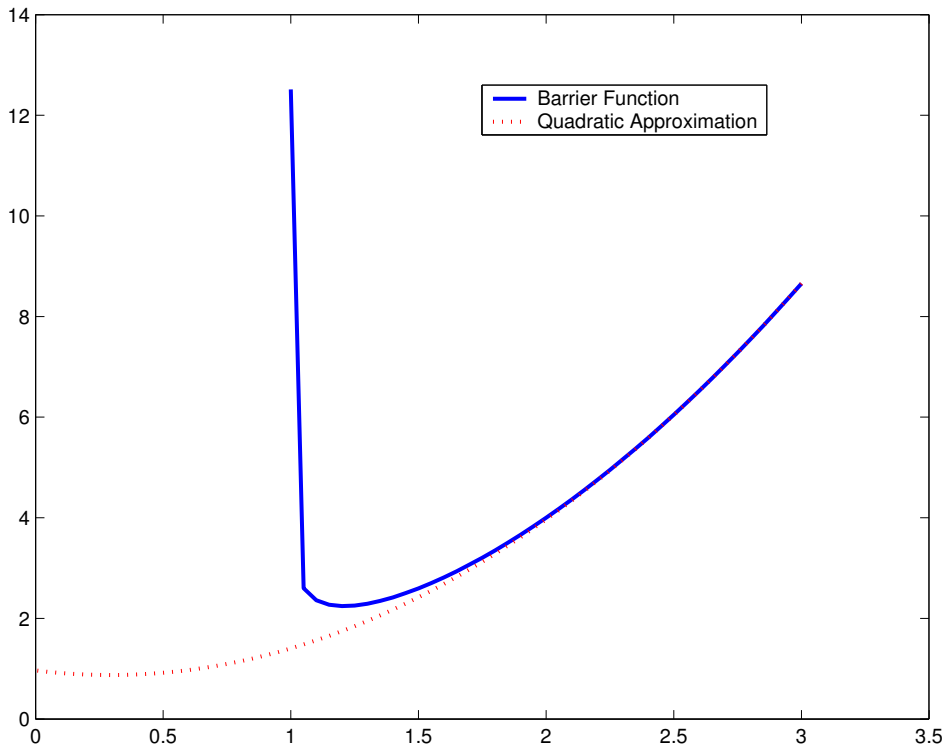

FIGURE 5.2 *Quadratic approximation example*

show that although low-order polynomials are typically used to estimate $\psi(\alpha)$, the estimating functions can not reflect the singularity at the barrier. Figure 5.2 illustrates this problem, plotting both a barrier function and a quadratic interpolation function based on the Taylor-series expansion of the barrier objective function:

$$\hat{q}(x) = q(x_k) + q'(x_k)(x - x_k) + \tfrac{1}{2}q''(x_k)(x - x_k)^2.$$

Notice how the true minimum of the function can not be detected. In view of this difficulty we base our linesearch method on an algorithm developed by Lin using linear constraints [Lin02]. Lin's method is based on [MW94].

5.2.1 Approach

Given a twice-differentiable objective function $f(x)$ and constraint functions $c_i(x), i \in 1, \ldots, m$, the merit function $M(x, y)$ and its gradient $M'(x, y)$ are defined as

$$\begin{aligned} M(x,y) &= f(x) - \mu \textstyle\sum_{j=1}^{n} \ln x_j - y^T c + \tfrac{1}{2} c^T R c \\ M'(x,y) &= \begin{pmatrix} \nabla f(x) - \mu \mathbf{X}^{-1} e - J^T y + J^T R c \\ -c \end{pmatrix}. \end{aligned}$$

With the search directions $(\Delta x, \Delta y)$, the safeguarded linesearch procedure determines the step size α that approximately minimizes the univariate function

$$\psi(\alpha) = M(x + \alpha\Delta x, y + \alpha\Delta y).$$

Lemma 5.1. *When* $\psi(\alpha) = M(x + \alpha\Delta x, y + \alpha\Delta y)$ *then*

$$\begin{aligned} \psi(0) &= M(x, y) \\ \psi'(0) &= M'(x, y)^T \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ \psi'(\alpha) &= M'(x + \alpha\Delta x, y + \alpha\Delta y)^T \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}. \end{aligned}$$

Proof. Let

$$\begin{pmatrix} \tilde{x} & \tilde{y} \end{pmatrix} = \begin{pmatrix} x + \alpha\Delta x & y + \alpha\Delta y \end{pmatrix}.$$

By definition,

$$\psi(\alpha) = M(\tilde{x}, \tilde{y}) = f(\tilde{x}) - \mu \sum_{j=1}^{n} \ln(\tilde{x}_j) - \tilde{y}^T c(\tilde{x}) + \tfrac{1}{2} c(\tilde{x})^T R c(\tilde{x}),$$

and therefore $\psi(0) = M(x, y)$. Next, consider

$$\begin{aligned} \psi'(\alpha) &= (\nabla f(\tilde{x}) - \mu \tilde{\mathbf{X}}^{-1} e)^T \Delta x - \tilde{y}^T J(\tilde{x}) \Delta x - \Delta y^T c(\tilde{x}) + c(\tilde{x})^T R J(\tilde{x}) \Delta x \\ &= \begin{pmatrix} g(\tilde{x}) - \mu \tilde{\mathbf{X}}^{-1} e - J(\tilde{x})^T \tilde{y} + J(\tilde{x})^T R c(\tilde{x}) \\ -c(\tilde{x}) \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ &= M'(\tilde{x}, \tilde{y})^T \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}. \end{aligned} \tag{5.3}$$

Consequently,

$$\begin{aligned} \psi'(0) &= \begin{pmatrix} g(x) - \mu \mathbf{X}^{-1} e - J^T y + J^T R c \\ -c(x) \end{pmatrix}^T \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \\ &= M'(x, y)^T \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}. \end{aligned}$$

∎

5.2.2 Linesearch algorithm

The linesearch algorithm first verifies that there is sufficient descent for the merit function. This is guaranteed when

$$\psi'(0) \leq -\tfrac{1}{2}\Delta x^T W \Delta x,$$

where W is the Hessian for problem (5.1). If this inequality does not hold, then the routine **compute_rho** is called to increase individual ρ_i's, thereby guaranteeing sufficient descent.

Given sufficient descent, the first routine called is **initializeAlpha**. This routine estimates an initial α using both linear and quadratic interpolation functions for $f(x)$. This routine also returns an interval of uncertainty $[0, \alpha_{\max}]$ for α and guarantees feasibility (i.e., $x + \alpha_{\max}\Delta x > 0$).

If the initial α satisfies the inequality

$$\psi(\alpha) \leq \psi(0) + \gamma_2 \alpha \psi'(0),$$

then α is returned and the linesearch is complete. Otherwise we iterate, updating our estimate of α until the following Gamma conditions are satisfied:

$$|\psi'(\alpha_k)| \leq -\gamma_1 \psi'(0) \;\text{ and }\; \psi(\alpha_k) \leq \psi(0) + \gamma_2 \alpha_k \psi'(0),$$

where $0 \leq \gamma_1 < 1$ and $0 \leq \gamma_2 < \frac{1}{2}$. If γ_1 is small, we have an 'accurate linesearch'. If $\gamma_1 = 0$, we have an 'exact linesearch', and the procedure terminates at a stationary point of $\psi(\alpha)$. Usually γ_2 is chosen to be small, increasing the probability that an approximate minimizer will satisfy the condition.

When $0 < \gamma_2 \leq \gamma_1 < 1$, then $\Gamma(\gamma_1, \gamma_2)$ contains a nontrivial interval [GMW81].

The iteration loop for refining α has three steps. First, the interval of uncertainty $[\alpha_{lo}, \alpha_{hi}]$ is updated for the current estimate of α in **updateUncertainty**. Then **computeAlpha** is called to update the estimate for α using a quadratic interpolating function based upon α_{lo}. Finally, **safeguard** is called to verify, and if necessary change, α to ensure it is a reasonable step size.

The algorithm for the safeguarded linesearch is given in Algorithm 5.2. The routines in bold are detailed in the following subsections.

Data : $x, y, \gamma_1, \gamma_2, \Delta x, \Delta y$
Result : α
Verify that there is sufficient descent
if $\psi'(0) > -\frac{1}{2}\Delta x^T W \Delta x$ **then**
  Call **compute_rho** to update the penalty parameters

1 Obtain initial estimate for α
$\alpha \leftarrow$ **initializeAlpha**
2 **if** $(\psi(\alpha) > \psi(0) + \gamma_2 \alpha \psi'(0))$ **then**
  $[\alpha_{lo}, \alpha_{hi}] \leftarrow$ **updateUncertainty**
  while $(|\psi'(\alpha)| > -\gamma_1 \psi'(0))$ ***or*** $(\psi(\alpha) > \psi(0) + \gamma_2 \alpha \psi'(0))$ **do**
    $\alpha_{temp} \leftarrow$ **computeAlpha**
    $\alpha \leftarrow$ **safeguard**
    $[\alpha_{lo}, \alpha_{hi}] \leftarrow$ **updateUncertainty**

Algorithm 5.2: Linesearch algorithm

5.2.3 initializeAlpha

In this section we describe an algorithm for computing the initial estimate for α. The goal is to compute an α efficiently using interpolating functions.

The first step is to compute a maximum allowable step $\alpha_{\max}$ such that $x + \alpha_{\max}\Delta x \geq 0$ according to

$$\alpha_{\max} = \min_{\Delta x_j < 0} -\frac{x_j}{\Delta x_j}.$$

The starting interval of uncertainty is set to $[0, \alpha_{\max}]$.

Next, we use a linear interpolant to refine the interval of uncertainty $[\alpha_{lo}, \alpha_{hi}]$ with $\psi'(\alpha_{lo}) < 0$ and $\psi'(\alpha_{hi}) > 0$ and use this interval for the initial estimate of α.

Linear Interpolant

Computing the merit function for each possible α is often computationally expensive because of the cost of the objective and constraint function calls. As a result, the merit function values for potential α values are interpolated. In order to do linear interpolation for $\psi(\alpha) = M(x + \alpha\Delta x, y + \alpha\Delta y)$, we first define the following univariate functions:

$$\begin{aligned} \varphi(\alpha) &= \textstyle\sum_{j=1}^{n} \ln(x_j + \alpha \Delta x_j) \\ \phi(\alpha) &= \psi(\alpha) - \mu\varphi(\alpha), \end{aligned}$$

where B is the log-barrier function. The interpolating function $\overline{\psi}$ is defined as

$$\overline{\psi}(\alpha) = \overline{\phi}(\alpha) - \mu\varphi,$$

where $\overline{\phi}(\alpha) = a + b\alpha$ is the linear interpolation for $\phi(\alpha)$; that is, $\overline{\phi}'(0) = b = \phi'(0)$.

Lin proves that a sufficient condition for a unique minimizer $\alpha_{\overline{\psi}}$ to exist in $[0, \alpha_{\max}]$ is that $\phi'(0) < 0$ [Lin02].

Initially, we need to set up an interval of uncertainty $[\alpha_{lo}, \alpha_{hi}]$ with $\psi'(\alpha_{lo}) < 0$ and $\psi'(\alpha_{hi}) > 0$. Since $\psi'(0) < 0$, we estimate α_{lo} given only the values $\psi(0), \psi'(0)$ and $\phi'(0)$. We use the linear interpolation function

$$\begin{aligned} \Theta_1(\alpha) &= \overline{\phi}(\alpha) - \mu\overline{\varphi}(\alpha) \\ &= a + b\alpha - \mu(\ln(\alpha_{\max} - \alpha) + c\alpha + f). \end{aligned}$$

As shown in [Lin02], this function's derivatives are

$$\begin{aligned} \Theta_1'(\alpha) &= (b - \mu c) + \tfrac{\mu}{\alpha_{\max} - \alpha}, \\ \Theta_1''(\alpha) &= \tfrac{\mu}{(\alpha_{\max} - \alpha)^2}. \end{aligned}$$

The second derivative is always positive, so Θ_1 is strictly convex with a minimum at α_{Θ_1} satisfying

$$\Theta_1'(\alpha_{\Theta_1}) = (b - \mu c) + \frac{\mu}{\alpha_{\max} - \alpha} = 0.$$

Solving for α_{Θ_1}, we obtain the analytic solution

$$\begin{aligned} \alpha_{\Theta_1} &= \alpha_{\max} + \tfrac{u}{b - \mu c}, \\ &= \tfrac{\psi'(0)}{\psi'(0) - \mu/\alpha_{\max}}\alpha_{\max}. \end{aligned}$$

Given α_{lo}, we find the smallest α_{hi} such that $\psi'(\alpha_{hi}) > 0$. Given this interval of uncertainty, we now perform a quadratic interpolation within the interval, reducing the interval until $\psi'(\alpha) \approx 0$.

Quadratic Interpolant

Given an interval $[\alpha_{lo}, \alpha_{hi}]$ for $\overline{\psi}(\alpha)$ with $\overline{\psi}'(\alpha_{lo}) < 0$ and $\overline{\psi}'(\alpha_{hi}) > 0$, we use the interpolation function

$$\begin{aligned} \Theta_2(\alpha) &= \overline{\phi}(\alpha) - \mu\overline{\phi}(\alpha) \\ &= a + b\alpha - \mu(d\ln(\alpha_{\max} - \alpha) + c\alpha + f). \end{aligned}$$

Its derivatives are

$$\begin{aligned} \Theta_2'(\alpha) &= (b - \mu c) + \tfrac{\mu d}{\alpha_{\max} - \alpha}, \\ \Theta_2''(\alpha) &= \tfrac{\mu d}{(\alpha_{\max} - \alpha)^2}. \end{aligned}$$

Lin fitted $\Theta_2(\alpha)$ to values of $\overline{\psi}'(\alpha)$ at α_{lo} and α_{hi}, showing it is convex and obtaining the following estimate for the minimum:

$$\alpha_{\Theta_2} = \alpha_{\max} + \frac{(\alpha_{\max} - \alpha_{hi})(\alpha_{\max} - \alpha_{lo})\left[\overline{\psi}'(\alpha_{hi}) - \overline{\psi}'(\alpha_{lo})\right]}{(\alpha_{\max} - \alpha_{lo})\overline{\psi}'(\alpha_{lo}) - (\alpha_{\max} - \alpha_{hi})\overline{\psi}'(\alpha_{hi})}.$$

The **initializeAlpha** routine is outlined in Algorithm 5.3.

Data : $x, y, \Delta x, \Delta y, \gamma, \gamma_1, \gamma_2, \eta$
Result : $\alpha, \alpha_{lo}, \alpha_{hi}$

Compute the maximum step size we can take and still be feasible
$\alpha_{\max} \leftarrow \min(1, -\frac{x_j}{|\Delta x_j|}, \Delta x_j < 0)$

Compute $\alpha_{\min}$, with $\psi(0), \psi'(0)$ and $\phi'(0)$
$\alpha_{\min} = \frac{\psi'(0)\alpha_{\max}}{\psi'(0) - \mu\alpha_{\max}}$
$x_{test} = x + \alpha_{\min}\Delta x$
Invert each value of $(x_{test})_j = \frac{1}{(x_{test})_j}$
$g_\psi = b - \mu(x_{test}^T \Delta x)$

Given $\alpha_{\min}$, get the next largest α where the derivative changes sign
while $g_{\min} < -\varepsilon$ **do**
  $\alpha_{lo} = \alpha_{\min}$
  $\alpha_{\min} = \zeta\alpha_{\max} + (1-\zeta)\alpha_{\min}$
  $x_{test} = x + \alpha_{\min}\Delta x$
  Invert each value of $(x_{test})_j = \frac{1}{(x_{test})_j}$
  $g_{\min} = b - \mu(x_{test}^T \Delta x)$
$\alpha_{hi} \leftarrow \alpha_{\min}$

Compute α where $\Theta'(\alpha) \approx 0$ in $[\alpha_{lo}, \alpha_{hi}]$
while $|\Theta'(\alpha)| > \varepsilon$ **do**
  $\alpha_{\min} = \alpha_{\max} + \frac{(\alpha_{\max} - \alpha_{hi})(\alpha_{\max} - \alpha_{lo})\left[\overline{\psi}'(\alpha_{hi}) - \overline{\psi}'(\alpha_{lo})\right]}{(\alpha_{\max} - \alpha_{lo})\overline{\psi}'(\alpha_{lo}) - (\alpha_{\max} - \alpha_{hi})\overline{\psi}'(\alpha_{hi})}$
  $\alpha_{trial} \leftarrow$ **safeguard**
  $x_{test} = x + \alpha_{\min}\Delta x$
  Invert each value of $(x_{test})_j = \frac{1}{(x_{test})_j}$
  $g \leftarrow b - \mu(x_{test}^T \Delta x)$
  if $\Theta'(\alpha) > 0$ **then**
    $\alpha_{hi} = \alpha_{trial}$
  else
    $\alpha_{lo} = \alpha_{trial}$
$\alpha = \alpha_{\min}$

Algorithm 5.3: Initialize alpha

5.2.4 computeAlpha

When the initial estimate for α does not pass the linesearch termination criterion $\psi(\alpha^0) - \psi(0) > \gamma_3\alpha^0\psi'(0)$, the estimate for α is updated in **computeAlpha**. We use a quadratic

interpolating function for ϕ:

$$\tilde{\psi}(\alpha) = \tilde{\phi}(\alpha) - \mu\varphi(\alpha),$$

where we interpolate at $\alpha = \alpha_{lo}$ with $\psi'(\alpha_{lo}) < 0$:

$$\tilde{\phi}(\alpha) = a + b(\alpha - \alpha_{lo}) + \frac{c}{2}(\alpha - \alpha_{lo})^2$$

and

$$\begin{aligned} a &= -\mu \\ b &= \phi'(\alpha_{lo}) \\ c &= \tfrac{1}{\alpha_s - \alpha_{lo}}(\phi'(\alpha_s) - \phi'(\alpha_{lo})) \end{aligned}$$

and we use interval reduction to find the linesearch minimizer. Algorithm 5.4 outlines this routine.

Data : $x, y, \alpha_{fit}, \alpha_{\max}, \alpha_{lo}, \alpha_{hi}, df_{lo}, df_{hi}, interval\_length, \mu$
Result : $\alpha, \alpha_{lo}, \alpha_{hi}$

1 Set initial guess to α_{hi} when it is the maximum value
if $\alpha_{hi} == \alpha_{\max}$ **then**
  $\alpha_s = \alpha_{fit}$
  $df_s = df_{fit}$
else
  $\alpha_s = \alpha_{hi}$
  $df_s = df_{high}$
Set $(x_{inv})_j = \frac{1}{x_j}$
Set $\psi \leftarrow df_{fit} + \mu({x_{inv}}^T \Delta x)$
Set $b1 \leftarrow df_{low} + \mu({x_{inv}}^T \Delta x)$
Set $c1 \leftarrow (\psi - b1)/(\alpha_s - \alpha_{lo})$
Set $length\_iu \leftarrow \alpha_{hi} - \alpha_{lo}$
while $(|df_s| > 0)$ ***and*** $(length\_iu > interval\_length)$ **do**

2 Perform the interpolation to update guess for α
  Set $\alpha_\delta \leftarrow \alpha_{\max} - \alpha_{lo}$
  Set $b \leftarrow df_{low} - \frac{\mu}{\alpha_\delta}$
  Set $c \leftarrow \frac{1}{\alpha_s - \alpha_{lo}}(df_s - b - \frac{\mu}{\alpha_{\max} - \alpha_s})$
  If c is 0 then use only α and b to update α_{trial} **if** $c = 0$ **then**
    Set $\alpha_{trial} \leftarrow \alpha_{lo} + \frac{\mu + b\alpha_\delta}{b}$
  else
    Set $\alpha_{trial} \leftarrow \alpha_{lo} + c\frac{\alpha_\delta - b - \sqrt{(b + c\alpha_\delta)^2 + 4c\mu}}{2c}$
  Set $\alpha_{\min} = \textbf{safeguard}(\alpha_{trial})$
3 Estimate new value
  $df_s = b1 + c1(\alpha_{\min} - \alpha_{lo}) - \mu {x_{inv}}^T \Delta x;$
  $[\alpha_{lo}, \alpha_{hi}] \leftarrow$ **updateUncertainty**
  Set $length\_iu = \alpha_{hi} - \alpha_{lo}$
  end;
Set $\alpha \leftarrow \alpha_{\min}$

Algorithm 5.4: Compute alpha

5.2.5 updateUncertainty

The interval of uncertainty is defined as $[\alpha_{lo}, \alpha_{hi}]$ with $\psi'(\alpha_{lo}) < 0$ and $\psi'(\alpha_{hi}) > 0$. Within this interval, there exists a minimizer of the merit function.

UpdateUncertainty updates this interval when a new estimate of α has been computed. The value of $\psi'(\alpha)$ replaces the lower or higher bound based on whether it is negative or positive.

The variable hit_ic is also updated in this routine. This variable is used in the function **safeguard** to prevent slow convergence to an endpoint of this interval.

Data : hit_ic, α_{lo}, α_{hi}, α
Result : hit_ic, α_{lo}, α_{hi}

1 **if** $\big(\psi(\alpha^0) > \psi(0)\big)$ ***or*** $(\psi'(\alpha) > 0)$ **then**
  $\alpha_{hi} \leftarrow \alpha$
  hit_ic(2) $\leftarrow$ hit_ic(2) + 1
  hit_ic(1) $\leftarrow 0$
else
  $\alpha_{lo} \leftarrow \alpha$
  hit_ic(1) $\leftarrow$ hit_ic(1) + 1
  hit_ic(1) $\leftarrow 0$

Algorithm 5.5: Update interval of uncertainty

5.2.6 Safeguard

The function **safeguard** is provided to prevent slow convergence to one of the end points of the interval of uncertainty. Given three points, $\alpha_{trial}, \alpha_{lo}, \alpha_{hi}$, we safeguard α by setting it to

$$\alpha = \begin{cases} \alpha_{lo} + \eta_1(\alpha_{hi} - \alpha_{lo}) & \text{if } \alpha_{trial} < \alpha_{lo} + \eta_1(\alpha_{hi} - \alpha_{lo}) \\ \alpha_{hi} - \eta_1(\alpha_{hi} - \alpha_{lo}) & \text{if } \alpha_{trial} > \alpha_{hi} - \eta_1(\alpha_{hi} - \alpha_{lo}) \\ \alpha_{trial} & \text{otherwise,} \end{cases}$$

where $0 \leq \eta_1 \leq 1$. The algorithm for this routine is outlined in Algorithm 5.6.

Data : hit_sg, hit_ic, α_{lo}, α_{hi}, α_{trial}, η_1
Result : α, hit_sg

1 Set $LB \leftarrow \eta_1\alpha_{hi} + (1-\eta_1)\alpha_{lo}$
Set $UB \leftarrow (1-\eta_1)\alpha_{hi} + \eta_1\alpha_{lo}$
if $\alpha < LB$ **then**
  if *(hit_ic(1) > 2)* ***and*** *(hit_sg(1) > 2)* **then**
    $\alpha \leftarrow UB$
  else if *(hit_ic(2) > 2)* ***and*** *(hit_sg(1) > 2)* **then**
    $\alpha \leftarrow \eta_2\alpha_{hi} + (1-\eta_2)\alpha_{lo}$
  else
    $\alpha \leftarrow LB$
  hit_sg(1) = hit_sg(1) + 1
  hit_sg(2) = 0
else if $\alpha > UB$ **then**
  if *(hit_ic(2) > 2)* ***and*** *(hit_sg(2) > 2)* **then**
    $\alpha \leftarrow LB$
  else if *(hit_ic(1) > 2)* ***and*** *(hit_sg(2) > 2)* **then**
    $\alpha \leftarrow \eta_2\alpha_{lo} + (1-\eta_2)\alpha_{hi}$
  else
    $\alpha \leftarrow HB$
  hit_sg(2) = hit_sg(2) + 1
  hit_sg(1) = 0
else
  hit_sg(1) $\leftarrow$ 0
  hit_sg(2) $\leftarrow$ 0

Algorithm 5.6: Safeguard algorithm

5.2.7 Compute rho

The function **compute_rho** is used by the safeguarded linesearch and the curvilinear search. It is called when we do not have a sufficient descent direction or significant direction of negative curvature.

In order to achieve a sufficient reduction in the merit function for the safeguarded linesearch we require

$$\psi'(0) \leq -\tfrac{1}{2}\Delta x^T W \Delta x \tag{5.4}$$

[GMW81]. When this inequality is not satisfied, ρ is modified to obtain a sufficient descent direction. The minimum ρ that guarantees satisfying equation (5.4) is

$$\rho_{\min} = \frac{g_B^T \Delta x + (y - \Delta y)^T c + \frac{1}{2}\Delta x^T W \Delta x}{\|c\|^2},$$

where g_B is the gradient of the objective of (5.1) and W is the Hessian of the Lagrangian for the barrier subproblem. It is shown in [Eld91] that the minimum-Euclidean-norm choice of the vector of penalty parameters for the augmented Lagrangian merit function is given by $\rho^* = \lambda r$, for

$$\lambda = \frac{g_B^T \Delta x + (y - \Delta y)^T c + \frac{1}{2}\Delta x^T W \Delta x}{r^T r},$$

and $r_i = c_i^2, i = 1, \ldots, m$.

This value $\rho_{\min}$ ensures that (5.4) is satisfied and we have a direction of descent for the linear merit function. However, in order to ensure that Δx is a descent direction for the curvilinear merit function, 0.5 must be added to $\rho_{\min}$. Since $\rho > \rho_{\min}$, we use the curvilinear computation for $\rho_{\min}$. The equivalence of the two ρ values is proven in the curvilinear search section 5.3.

It is also necessary to keep ρ from getting too large, and therefore when an individual $\rho_i > \rho_i^*$ it can be reduced and still satisfy equation (5.4). Eldersveld shows that in this situation the individual value can be reset to the geometric mean of ρ_i and ρ_i^*:

$$\rho_t = \sqrt{\rho_i(\delta_k + \rho_i^*)}, \tag{5.5}$$

where $\delta_k \geq 1$. The new ρ_i is

$$\rho_i = \begin{cases} \rho_t & \text{if } \rho_t \leq \frac{1}{2}\rho_i \\ \rho_i & \text{otherwise.} \end{cases}$$

The parameter δ_k is increased by a factor of two whenever a ρ_i is decreased to prevent too many updates to ρ.

Algorithm 5.7 defines the algorithm for computing ρ.

Data : $\rho, x, y, \Delta x, \Delta y, g_B, c, \delta$
Result : ρ, δ

1 Compute $\theta \leftarrow g_B^T \Delta x + (y - \Delta y)^T c + \frac{1}{2}\Delta x^T W \Delta x$
Set $r_i \leftarrow c_i^2$ for $i = 1, \ldots, m$
Set $\rho_{\min} \leftarrow \frac{\theta}{c^T c} c + \frac{1}{2}$
2 Set the new ρ values, handling decreases
for *i = 1* ***to*** m **do**
  if $\rho(i) > \rho_{\min}$ **then**
    Set ρ_t per equation (5.5)
    if $\rho_t \leq \frac{1}{2}\rho(i)$ **then**
      $\rho(i) \leftarrow \rho_t$
      $\delta \leftarrow 2\delta$
  else
    $\rho(i) \leftarrow \rho_{\min}(i)$

Algorithm 5.7: Compute ρ

5.3 Curvilinear search

When a direction of negative curvature exists, it is used to compute the next step size, α. This algorithm is based on the curvilinear search of Murray and Prieto [MP95]. Our implementation of the method uses a backtracking search to obtain α.

5.3.1 Curvilinear merit function

The curvilinear search also uses the augmented Lagrangian merit function to find α, combining the descent directions and directions of negative curvature for the primal variable x and using only the descent direction for the dual variable y. Since this routine is independent of the barrier method, all primal variables are combined into x. The gradient and Hessian are functions of the barrier (see (2.7)).

Lemma 5.2. *The univariate function used to compute α when there is a sufficient direction of negative curvature is*

$$\psi(\alpha) = M(x + \alpha^2 \Delta x + \alpha dn, y + \alpha^2 \Delta y) \tag{5.6}$$

and has the following properties:

$$\begin{aligned}
\psi(0) &= M(x,y) \\
\psi'(\alpha) &= \nabla M(x+\alpha^2\Delta x+\alpha dn, y+\alpha^2\Delta y)^T \begin{pmatrix} 2\alpha\Delta x+dn \\ 2\alpha\Delta y \end{pmatrix} \\
\psi'(0) &= \nabla M(x,y)^T \begin{pmatrix} dn \\ 0 \end{pmatrix}.
\end{aligned}$$

Proof. By definition,

$$\psi(0) = M(x,y). \tag{5.7}$$

Let

$$\begin{aligned}
\hat{x} &= x+\alpha^2\Delta x+\alpha dn \\
\hat{y} &= y+\alpha^2\Delta y \\
q &= \begin{pmatrix} 2\alpha\Delta x+dn \\ 2\alpha\Delta y \end{pmatrix}.
\end{aligned}$$

By definition,

$$\nabla M = \begin{pmatrix} g(\hat{x}) - J(\hat{x})^T\hat{y} + J(\hat{x})^T Rc(\hat{x}) \\ -c(\hat{x}) \end{pmatrix}$$

and we have

$$\begin{aligned}
\psi'(\alpha) &= g(\hat{x})^T(2\alpha\Delta x+dn) - \hat{y}^T J(\hat{x})(2\alpha\Delta x+dn) - 2\alpha c(\hat{x})^T\Delta y + J(\hat{x})^T Rc(\hat{x})(2\alpha\Delta x+dn) \\
&= \begin{pmatrix} g(\hat{x}) - J(\hat{x})^T\hat{y} + c(\hat{x})^T RJ(\hat{x}) \\ -c(\hat{x}) \end{pmatrix}^T q \\
&= \nabla M(\hat{x},\hat{y})^T q.
\end{aligned} \tag{5.8}$$

It is now apparent that

$$\begin{aligned}
\psi'(0) &= \begin{pmatrix} g(x) - J^T y + J^T Rc \\ -c \end{pmatrix}^T \begin{pmatrix} dn \\ 0 \end{pmatrix} && (5.9) \\
&= \nabla M(x,y)^T \begin{pmatrix} dn \\ 0 \end{pmatrix}. && (5.10)
\end{aligned}$$

∎

As defined in [MP95], the second derivative of the merit function is only required for

$\alpha = 0$. The second derivative is

$$\psi''(0) = 2\begin{pmatrix}\Delta x^T & \Delta y^T\end{pmatrix}\nabla M + \begin{pmatrix}dn^T & 0^T\end{pmatrix}\nabla^2 M \begin{pmatrix}dn \\ 0\end{pmatrix}. \tag{5.11}$$

5.3.2 Curvilinear Algorithm

Before beginning the curvilinear search, we ensure that we have a direction of sufficient negative curvature. First, since the sign of dn is ambiguous, when

$$g_B(x)^T dn > 0,$$

where $g_B(x)$ is the gradient of $f(x) - \mu \sum_j \ln(x_j)$, we switch the sign of dn. This switch makes sure that dn is a descent direction for the merit function. We next check that

$$\phi''(0) > -\omega + \tfrac{1}{2} dn^T W dn,$$

where

$$\omega = \tfrac{1}{2}(c^T c + \Delta x^T W \Delta x). \tag{5.12}$$

When this inequality holds, we need to increase ρ_i to guarantee descent for ψ. In this situation, ρ is updated by calling the function **compute_rho** discussed in section 5.2.

Lemma 5.3. *Define $\hat{\rho}_{\min}$ to be the minimum multiplier, ρ, that guarantees $(\Delta x + dn, \Delta y)$ is a direction of descent for the augmented Lagrangian function. Then*

$$\hat{\rho} = \frac{\frac{1}{2}\omega + g^T \Delta x + (y - \Delta y)^T c}{\|c\|},$$

as shown in [MP95]. Further, let $\bar{\rho}_{\min}$ be the minimum ρ to guarantee descent for the augmented Lagrangian applied for $(\Delta x, \Delta y)$. Then

$$\bar{\rho} = \frac{g^T \Delta x + (y - \Delta y)^T c + \frac{1}{2}\Delta x^T W \Delta x}{\|c\|^2}$$

as shown in [Eld91]. Given these two definitions, $\hat{\rho} = \bar{\rho} + \frac{1}{2}$.

Proof. From the definition of ω: $\frac{1}{2}\omega = \frac{1}{2}\Delta x^T W \Delta x + \frac{1}{2}\|c\|^2$. ∎

The direction of negative curvature is sometimes discarded [MP95] and the safeguarded linesearch is performed using only Δx, as detailed in section 5.2. When the penalty multipliers remain unchanged, a curvilinear search is carried out to compute α, using a backtracking algorithm.

The first step computes a starting α. Given the bounds on x we compute $\alpha_{\max}$, the

minimum among all $\{\alpha^+_{x_j}, \alpha^-_{x_j}\}$. This set is computed by solving the quadratic equation

$$x_j + \alpha^2_{x_j} \Delta x_j + \alpha_{x_j} dn_{x_j} = 0$$

for its two roots $\alpha^+_{x_j}, \alpha^-_{x_j}$ for each j and defining $\alpha_{x_j} = \min\{\alpha^+_{x_j}, \alpha^-_{x_j}\}$. Finally, $\alpha_{\max}$ is set to

$$\alpha_{\max} = \min_{\alpha_{x_j} > 0} \alpha_{x_j}.$$

The initial value of α is set to $\min(1, \alpha_{\max} * \gamma)$, where $\|\gamma - 1.0\|$ is very small. This step size ensures that the constraints $x \geq 0$ remain strictly satisfied. We now test if α satisfies the termination criterion

$$\psi(\alpha) - \psi(0) \leq \gamma_2(\alpha\psi'(0) + \tfrac{1}{2}\alpha\psi''(0)),$$

and if so, α is returned. Otherwise, α is reduced by multiplying it by a constant, $0 < \beta < 1$, until

$$\begin{aligned} \psi(\alpha) - \psi(0) &\leq \gamma_2(\alpha\psi'(0) + \tfrac{1}{2}\alpha\psi''(0)) \\ \psi'(\alpha) &\geq \gamma_1(\psi'(0) + \alpha\psi''(0)), \end{aligned}$$

where $0 < \gamma_2 \leq \frac{1}{2}$ and $\frac{1}{2} \leq \gamma_1 < 1$. Using this criterion we are guaranteed a minimizer for $\psi(\alpha)$ and $M(x_{k+1}) < M(x_k)$ [MP95].

Algorithm 5.8 presents the algorithm for the curvilinear search.

Data : $x, y, \Delta x, \Delta y, dn, \beta$
Result : α, dn

1 Test if we have a direction of significant negative curvature
begin
  if $g^T dn > 0$ **then**
    $dn = -dn$

  Compute $\psi(0)$ as defined in (5.7)
  Compute $\psi'(0)$ per (5.9)
  Compute $\psi''(0)$ per (5.11)
  Compute ω per (5.12)
  if $\nabla\psi''(0) > -\omega + \frac{1}{2} dn^T W dn$ **then**
    Update ρ per linear search **compute_rho**
    Set dn to all 0's
    return
end

2 Compute $0 < \alpha_{\max} \leq 1$, such that $x + \alpha_{\max}^2 \Delta x + \alpha_{\max} dn > 0$
begin
  For each element x_j,
    Compute roots of $x_j + \alpha_j^2 \Delta x_j + \alpha_j dn_j = 0$
  Set $\alpha_{\max}$ to the minimum for all roots greater than 0
end
$\alpha = \min(1, \gamma * \alpha_{\max})$

3 Backtrack, reducing α until we satisfy the termination conditions
while $\psi(\alpha) - \psi(0) > \gamma_2(\alpha\psi'(0) + \frac{1}{2}\alpha\psi''(0))$
  and $\psi'(\alpha) \geq \gamma_1(\psi'(0) + \alpha\psi''(0))$ **do**
  $\alpha \leftarrow \beta * \alpha$
  Compute $\psi(\alpha)$ per (5.6)
  Compute $\psi'(\alpha)$ per (5.8)

Algorithm 5.8: Curvilinear search

5.4 Dual linesearch

Once the step size for the primal values x has been computed, a separate linesearch is done to update the dual variables, y and z. In this routine, we find the step α_{yz} that minimizes the residuals (4.2) and (4.3). Using an algorithm proposed by Lin [Lin02],

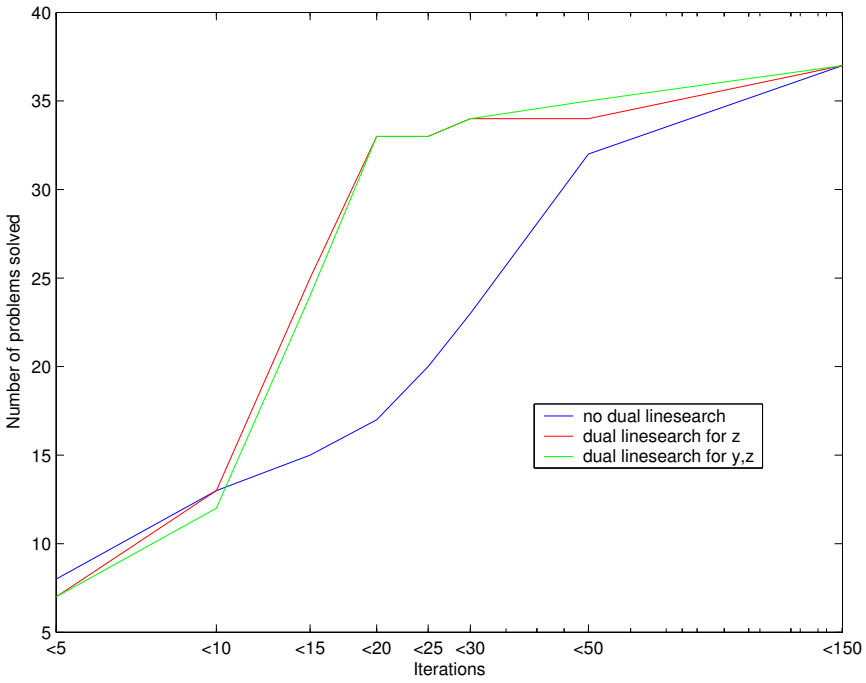

FIGURE 5.3 *Dual step size comparisons*

which determines a step length for z alone, we extend it to calculate one step length for both dual variables (y, z).

For a test set of 36 random problems, Figure 5.3 plots the iterations versus the number of test problems solved within the number of iterations. The plot compares different algorithms for computing the dual variable step sizes:

1. Using one α, as described in previous sections, for all variables.
2. One α for updating x and y and α_z for z.
3. One α for the primal variable and α_{yz} for the dual variables.

Using one α for all variables appears to be the worst approach, not unexpectedly because z is not a component in the merit function. The decision to use either α_{yz} or α_z is inconclusive from these test results. However, additional testing on 80 test problems from the Hock-Schittkowski test cases show α_{yz} is superior because it solved 33% more test problems than were solved using α_z.

This algorithm determines the α_{yz} that minimizes the norm of the residuals defined in section 4.1.1 involving the dual variables. The residuals are

$$\begin{aligned} \mu e - \breve{\mathbf{T}}(z + \alpha_{xy}\Delta z) \quad & \text{See (4.2)} \\ g(\breve{x}) - J(\breve{x})^T(y + \alpha_{yz}\Delta y) - (z + \alpha_{yz}\Delta z) \quad & \text{See (4.3),} \end{aligned}$$

where $(\breve{x}, \breve{t})$ are the updated values, using $\tilde{x} = x + \alpha\Delta x$ from the safeguarded linesearch or $\hat{x} = x + \alpha^2\Delta x + \alpha dn$ from the curvilinear search.

Letting x, t are the updated values, the unconstrained minimization problem solved is

$$\min_{\alpha_{yz}} \beta\|\mu e - \mathbf{T}(z + \alpha_{yz}\Delta z)\|^2 + (1-\beta)\|g - J^T(y + \alpha_{yz}\Delta y) - (z + \alpha_{yz}\Delta z)\|_2^2$$

for $0 \leq \beta \leq 1$. The analytical solution to this is

$$\alpha_{yz} = -\frac{\beta\Delta z^T\mathbf{T}(\mathbf{T}z - \mu e) + (1-\beta)(g - z - J^T y)(-J^T\Delta y - \Delta z)}{\beta\|\mathbf{T}\Delta z\|_2^2 + (1-\beta)(-\Delta z - J^T\Delta y)^2}. \tag{5.13}$$

The algorithm is defined in Algorithm 5.9.

Data : $g, J, \breve{x}, y, \breve{t}, z, \mu, \Delta y, \Delta z, \beta$
Result : α_{yz}

1 Set N to the numerator of (5.13)
$N = \beta\Delta z^T\mathbf{T}(\mathbf{T}z - \mu e) + (1-\beta)(g - z - J^T y)(-J^T\Delta y - \Delta z)$
2 Set D to the denominator of (5.13)
$D = \beta\|\mathbf{T}\Delta z\|_2^2 + (1-\beta)(-\Delta z - J^T\Delta y)^2$
$\alpha_{yz} = -\frac{N}{D}$

Algorithm 5.9: Dual linesearch

Chapter 6

Convergence

In this chapter we give a proof of global convergence and prove our barrier algorithm converges at a pure quadratic rate. In order to prove a quadratic rate of convergence we need to be more precise about the procedure for determining the maximum step and the procedure for reducing the penalty parameter. We also introduce a refinement of the termination criteria for solving the subproblem. For simplicity we assume a single barrier parameter and assume the problem is in the form

$$\begin{array}{lll} \underset{x}{\text{minimize}} & f(x) & \\ \text{subject to} & c(x) = 0, & x \geq 0. \end{array}$$

Let (x^*, y^*) be a solution of this system, where y is the dual variable. This solution is found by solving the sequence of subproblems

$$\begin{array}{lll} \underset{x>0,\ x\in I\!R^n}{\text{minimize}} & f(x) - \mu \sum_{j=1}^{n} \ln(x_j) & \\ \text{subject to} & c_i(x) = 0 & i = 1, 2, \ldots, m. \end{array} \tag{6.1}$$

for $\mu \in \{\mu_s\}$. The solution to (6.1) is represented by $(x^*(\mu), y^*(\mu))$. Our algorithm does not find $(x^*(\mu), y^*(\mu))$ for this subproblem, but finds an approximate solution to (6.1), $(\overline{x}(\mu), \overline{y}(\mu))$. This approximation is found by applying a damped Newton method to the KKT conditions (2.9). The search directions for Newton's method are given by the primal-dual KKT system

$$\begin{pmatrix} -H & I & J^T \\ \mathbf{Z} & \mathbf{X} & 0 \\ J & 0 & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta z \\ \Delta y \end{pmatrix} = \begin{pmatrix} g(x) - J^T y - z \\ \mu e - \mathbf{X} z \\ -c(x) \end{pmatrix}, \tag{6.2}$$

where (x, y, z) is the current estimate of the solution to (2.9). Recall that this system reduces to

$$\begin{pmatrix} -W & J^T \\ J & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} w \\ -c(x) \end{pmatrix}, \tag{6.3}$$

where

$$\begin{aligned} W &\equiv H_L + \mathbf{X}^{-1}\mathbf{Z}, \\ w &= g - J^T y - z - \mathbf{X}^{-1}(\mu e - \mathbf{X}z) \\ &= g - J^T y - \mu \mathbf{X}^{-1} e. \end{aligned}$$

The termination conditions (along with the assumptions given later) imply that the approximate solution to (6.1) satisfies

$$\lim_{\mu \to 0} \overline{x}(\mu) \to x^*, \quad \lim_{\mu \to 0} \overline{y}(\mu) \to y^*.$$

The merit function used for the linesearch routine is

$$M(x, y, \rho) = f(x) - \mu \sum_{j=1}^{n} \ln(x_j) - y^T c(x) + \tfrac{1}{2}\rho \|c(x)\|_2^2$$

for $\rho \geq 0$, $\rho \in I\!R$.

In order to show convergence, we must show at least that the algorithm for the subproblem always terminates successfully. The termination criteria ensure convergence to the KKT conditions as $\mu \to 0$. The criteria are

$$\begin{aligned} \|c(\breve{x})\|_\infty &\leq \tau_1 \mu \\ \|J^T \breve{y} + \breve{z} - g(\breve{x})\|_\infty &\leq \tau_2 \mu \\ \|\breve{\mathbf{X}}\breve{z} - \mu e\|_\infty &\leq \tau_3 \mu \\ \lambda_{\min}(Z^T W Z) &> -\tau_4 \mu, \end{aligned} \tag{6.4}$$

where

$$(\breve{x}, \breve{y}, \breve{z}) = \begin{cases} (\tilde{x}, \tilde{y}, \tilde{z}) & \text{if a linear search is used to compute } \alpha, \\ (\hat{x}, \hat{y}, \hat{z}) & \text{when a curvilinear search is used to compute } \alpha, \end{cases}$$

and $\tau_1, \ldots \tau_4$ are positive scalar parameters. Although (6.4) can always be satisfied by choosing τ_i sufficiently large, we will in general not know the lowest acceptable value for these parameters. Instead, we introduce a second condition for terminating the subproblem, namely:

$$\begin{aligned} \|c(\breve{x})\|_\infty &\leq \theta c(x) \\ \|J^T \breve{y} + \breve{z} - g(\breve{x})\|_\infty &\leq \theta(J^T y + z - g(x)) \\ \|\breve{\mathbf{X}}\breve{z} - \hat{\mu} e\|_\infty &\leq \theta(\mathbf{X}z - \mu_p) \\ \lambda_{\min}(Z^T W Z) &> 0, \end{aligned} \tag{6.5}$$

where $0.01 \leq \theta \leq 0.1$ and μ_p is the previous μ.

The proof that our algorithm terminates successfully is done in two parts. We first show that if μ is sufficiently small then for each subsequent choice of μ our algorithm requires only one step provided μ is not reduced too quickly and the initial point (x, y) is the termination point for the previous μ.

The result that only one step is required is also used to show that our algorithm converges at a quadratic rate. Note that this is a "true" quadratic rate of convergence for the original NLP and not simply that $\overline{x}(\mu)$ is converging quadratically for fixed μ.

6.1 Assumptions

The following assumptions refer to problem (1.1) and are based on [MP95].

Assumption 1 x^* is an isolated, local minimizer and satisfies the sufficient optimality conditions defined in section 1.3.

Assumption 2 There exist no first-order KKT points at ∞.

Assumption 3 $f, c_1, c_2, \ldots, c_m$ and their first, second and third derivatives are continuous and uniformly bounded in norm on any compact set.

Assumption 4 The isolated local minimizer x^* is in the closure of $x \geq 0$.

Assumption 5 Strict complementarity holds at all stationary points.

Assumption 6 The reduced Hessian of the Lagrangian function is nonsingular at all first-order KKT points.

Assumption 7 The Jacobian of the active constraints at any second-order KKT point has full rank.

Assumption 8 The set $\{\mu_s\}$ is a strictly decreasing positive sequence converging to 0.

Assumption 9 For some constants $\beta_c > 0$ and $\beta_x > 0$, the optimal objective value for

$$\begin{array}{lll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) & \\ \text{subject to} & |c_i(x)| \leq \beta_c e & i = 1, 2, \ldots, m \\ & x_j \geq -\beta_x e & j = 1, 2, \ldots, n \end{array}$$

is bounded below.

Assumption 10 At all iterates $\{x_k\}$ generated by the algorithm there exists a feasible step p from x_k that satisfies the linearization of the constraints and is such that

$$\|p\| \leq \beta_p \|c(x)\|, \quad g^T p \leq \beta_p \|c(x)\|$$

for some constant $\beta_p > 0$ and $x + p \geq 0$.

It is transparent from the proofs in [MP95] that the curvilinear search is required only in a finite neighborhood of a stationary point. Outside of this it does not matter.

6.1.1 Elastic variables

Assumption 10 can always be made to hold if elastic variables are used (see [GMS02] and [MP95a]). Elastic variables mean that two variables, v_i, w_i, are defined for each equality constraints. The constraints are modified to $c_i(x) + v_i - w_i = 0$ and $v_i, w_i \geq 0$. The modified constraints produce a Jacobian, $J = \begin{pmatrix} Jo & I & -I \end{pmatrix} \in I\!R^{m\times(n+2m)}$, where Jo is the original Jacobian.

The term $\theta \sum(v_i + w_i)$, $0 < \theta < \infty$ is also added to the objective. Apart from ensuring a feasible step to the linearized system always exists the addition of this term implies the Lagrange multipliers are bounded. In many instances the use of elastic variables allows a temporary problem with infeasibility to be overcome. When the linearization changes the issue of infeasibility may disappear. However, it is possible it never disappears and consequently the elastic variables do not converge to zero.

6.2 Theory

Lemma 6.1. *Let Z be a basis for the nullspace of J. There exists a $\overline{\mu}$ such that for all $\overline{\mu} \geq \mu > 0$, $Z^T(x^*(\mu))W_B(x^*(\mu), y^*(\mu))Z(x^*(\mu)) \succ 0$, where $W_B(x^*(\mu), y^*(\mu)) = H_L(x^*(\mu), y^*(\mu)) + \mu\mathbf{X}^{-2}$.*

Proof. We may assume without loss of generality that the full KKT-system is partitioned such that the variables that are zero at the solution are grouped first. Consider the following partitioned form of the KKT matrix in which J_a are the columns of the Jacobian matrix corresponding to these variables that are not zero in the solution. This system is

$$\begin{pmatrix} H_L^{11} & H_L^{12} & J_1^T \\ H_L^{21} & H_L^{22} & J_a^T \\ J_1 & J_a & 0 \end{pmatrix}.$$

Let

$$K_a = \begin{pmatrix} H_L^{22} & J_a^T \\ J_a & 0 \end{pmatrix}$$

and let Z_a be a basis for the nullspace of J_a. Let $\tilde{n}$ define the number of variables for which $x_j^* > 0$.

By assumption, $Z_a^T(x^*)H_L^{22}(x^*,y^*)Z_a(x^*) \succ 0$. It follows by continuity that for μ sufficiently small,

$$Z_a^T(x^*(\mu))H_L^{22}(x^*(\mu),y^*(\mu))Z_a(x^*(\mu)) \succ 0. \tag{6.6}$$

Part 1 The inertia of a symmetric matrix M is a triplet $\{i_p(M), i_n(M), i_z(M)\}$ containing the number of positive, negative and zero eigenvalues of M.

It is shown in [Gou85] that

$$In(K_a) = In(Z_a^T H_L^{22} Z_a) + (r, r, m-r).$$

Since J_a has full rank, (6.6) implies that $In(K_a) = (\tilde{n}-n, m, 0)$.

Now consider

$$K_b = \begin{pmatrix} H_L^{22} + \mu \mathbf{X}_2^{-2} & J_a^T \\ J_a & 0 \end{pmatrix} \equiv K_a + D.$$

It follows immediately that $In(K_b) = In(K_a)$ for μ small enough.

Now, consider the full matrix

$$K = \left(\begin{array}{c|cc} H_L^{11} + \mu \mathbf{X}_1^{-2} & H_L^{12} & J_1^T \\ \hline H_L^{21} & H_L^{22} + \mu \mathbf{X}_2^{-2} & J_a^T \\ J_1 & J_a & 0 \end{array}\right) \equiv \begin{pmatrix} A & B^T \\ B & K_b \end{pmatrix}, \tag{6.7}$$

where

$$\begin{aligned} A &= H_L^{11} + \mu \mathbf{X}_1^{-2} \\ B &= \begin{pmatrix} H_L^{12} & J_1^T \end{pmatrix}^T. \end{aligned}$$

From Sylvester's law of inertia, we know that K and its Schur complement have the same inertia. Reformulating K with its Schur complement we obtain the matrix

$$\begin{pmatrix} A & 0 \\ 0 & K_b - B^T A^{-1} B \end{pmatrix},$$

and A is positive definite by definition. Hence,

$$In(K) = In(A) + In(K_b - B^T A^{-1} B).$$

Since $\lim_{\mu\to 0} \|B^T A^{-1} B\|_2 = 0$ it follows that for μ small enough,

$$In(K) \quad = \quad In(A) + In(K_b).$$

Since $\lim_{\mu\to 0} \mu \mathbf{X}^{-1} e = x^* > 0$, by construction, and $z^* > 0$ it follows that

$$In(K) = (n, m, 0).$$

Using Gould once again, we can show that the reduced Hessian of the Lagrangian is positive definite for solutions that are close to the solution, or

$$Z^T(x^*(\mu)) W_B(x^*(\mu), y^*(\mu), z^*(\mu)) Z(x^*(\mu)) \succ 0. \quad \blacksquare$$

Lemma 6.2. *If $Z^T(x^*(\mu)) W_B(x^*(\mu), y^*(\mu), z^*(\mu)) Z(x^*(\mu))$ is positive definite, then the primal-dual reduced Hessian, $Z^T(x^*(\mu)) W(x^*(\mu), y^*(\mu), z^*(\mu)) Z(x^*(\mu))$, is positive definite for $\mu < \bar{\mu}$, as defined in Lemma 6.1, and $W = H_L(x^*(\mu), y^*(\mu)) + \mathbf{X}^{-1}\mathbf{Z}$.*

Proof. Theorem 3.3 in [Ng02]. $\blacksquare$

Lemma 6.3. *If $Z^T(x^*(\mu)) W(x^*(\mu), y^*(\mu)) Z^T(x^*(\mu))$ is positive definite, then there exists $\varepsilon > 0$, which is independent of μ, such that if $(\overline{x}(\mu), \overline{y}(\mu))$ satisfies*

$$\|\overline{x}(\mu) - x^*(\mu)\| < \varepsilon, \quad \|\overline{y}(\mu) - y^*(\mu)\| < \varepsilon,$$

then $Z^T(\overline{x}(\mu)) W(\overline{x}(\mu), \overline{y}(\mu)) Z(\overline{x}(\mu)) \succ 0$.

Proof. The eigenvalues $\lambda(\mu)$ of $Z^T(x) W(x, y) Z^T(x)$ vary continuously with $W(x, y)$ [FM90]. Given $Z^T(x^*(\mu)) W(x^*(\mu), y^*(\mu)) Z(x^*(\mu))$ is positive definite, all of its eigenvalues, $\lambda^*(\mu)$, are greater than 0. Let $\lambda_{\min}(\mu)$ and $\lambda^*_{\min}(\mu)$ denote the smallest eigenvalues. Then by continuity, for all $\delta > 0$, there exists a $\varepsilon > 0$ such that if

$$\left\| \begin{pmatrix} \overline{x}(\mu) \\ \overline{y}(\mu) \end{pmatrix} - \begin{pmatrix} x^*(\mu) \\ y^*(\mu) \end{pmatrix} \right\| < \varepsilon$$

then

$$\|\lambda_{\min}(\mu) - \lambda^*_{\min}(\mu)\| < \delta.$$

For δ small enough, $\lambda_{\min}(\mu)$ must be positive. The smallest eigenvalue of $Z^T(\overline{x}(\mu)) W(\overline{x}(\mu), \overline{y}(\mu)) Z^T(\overline{x}(\mu))$ is greater than 0, so $Z^T(\overline{x}(\mu)) W(\overline{x}(\mu), \overline{y}(\mu)) Z^T(\overline{x}(\mu))$ must be positive definite. $\blacksquare$

Lemma 6.4. *Consider $\overline{\mu}$ as defined in (6.1) and $\mu < \overline{\mu}$, where $(x(\mu), y(\mu), z(\mu))$ satisfies the termination criteria for μ, but not for μ^2. If*

$$\begin{aligned} \tilde{x} &= x + \alpha\Delta x \\ \tilde{y} &= y + \alpha\Delta y \\ \tilde{z} &= z + \alpha\Delta z, \end{aligned}$$

where $\Delta x, \Delta y, \Delta z$ are the computed primal-dual directions obtained from our algorithm, then $(\tilde{x}, \tilde{y}, \tilde{z})$ satisfies the termination criteria for μ^2. In other words,

$$\begin{aligned} \tilde{x} &> 0 \\ \|c(\tilde{x})\|_\infty &\leq M\mu^2 \\ \|g(\tilde{x}) - J(\tilde{x})^T\tilde{y} - \tilde{z}\|_\infty &\leq M\mu^2 \\ \lambda_{\min}(\tilde{x}, \tilde{y}, \tilde{z}) &> 0 \\ \|\tilde{\mathbf{X}}\tilde{z} - \mu^2 e\|_\infty &\leq M\mu^2, \end{aligned}$$

for some $M > 0$.

Proof. First, we show that feasibility is maintained for x and z. Then, we show the termination criteria are satisfied for $(\tilde{x}, \tilde{y}, \tilde{z})$.

1. Show $\tilde{x}, \tilde{z} > 0$.

1a. Show $\tilde{x} > 0$. We see from the top row of the KKT system (6.3) that

$$-(H_L + \mathbf{X}^{-1}\mathbf{Z})\Delta x = g - J^T(y + \Delta y) - \mu\mathbf{X}^{-1}e.$$

Consider any $\tilde{x}_j$ where $x_j^* = 0$. Omitting the subscripts, and solving this equation for the current barrier multiplier, μ^2,

$$\frac{z}{x}\Delta x = \frac{\mu^2}{x} - g(x) + \tilde{j} - \tilde{h},$$

where $\tilde{j} = (J^T(y + \Delta y))_j$ and $\tilde{h} = (H_L\Delta x)_j$. Since (x, y, z) terminated for μ, we know that

$$g(x) - J^T y - z = O(\mu)$$

and $\Delta x, \Delta y$ are also $O(\mu)$. Therefore,

$$\frac{z}{x}\Delta x = -z + O(\mu) + \frac{\mu^2}{x}$$

or

$$\Delta x = -x + xO(\mu) + \frac{\mu^2}{z}.$$

It follows that

$$x + \Delta x = Mx\mu + \frac{\mu^2}{z}, \tag{6.8}$$

where M is bounded. If $M \geq 0$ there is not an issue because $\tilde{x} > 0$. Therefore, suppose $M < 0$. Consider taking the step $\alpha = (1 - \theta\mu)$. We obtain

$$\begin{aligned} x + \Delta x - \theta\mu\Delta x - \frac{\mu^2}{z} &= Mx\mu - \theta\mu\Delta x \\ &= x\mu(M - \theta\frac{\Delta x}{x}). \end{aligned}$$

From equation (6.8), we have

$$\frac{\Delta x}{x} = -1 + M\mu + \frac{\mu^2}{xz} < 0,$$

or

$$xz > \frac{\mu^2}{1 - M\mu}.$$

This gives us

$$\begin{aligned} x + (1 - \theta\mu)\Delta x &= x\mu(M + \theta(1 - M\mu - \frac{\mu^2}{xz})) + \frac{\mu^2}{z} \\ &= x\mu(M + \theta(1 - M\mu - \frac{\mu^2}{xz}) + \frac{\mu}{xz}). \end{aligned}$$

If $\theta = -M$ and for $\alpha \leq 1$ we obtain

$$x + (1 - \theta\mu)\Delta x > \frac{\mu^2}{z} > 0.$$

1b. Show $\tilde{z} > 0$.

Again we consider a specific element of z and omit the subscript. Strict complementarity requires that $x_j^* > 0$. From the KKT system (6.2) we then have

$$z\Delta x + x\Delta z = \mu^2 - xz. \tag{6.9}$$

Rearranging (6.9) gives us

$$\Delta z = \frac{\mu^2 - (x + \Delta x)z}{x}. \tag{6.10}$$

When $\Delta z \geq 0$, then $z + \Delta z \geq 0$. Therefore, there is only an issue when $\Delta z < 0$.

Rearranging equation (6.10) we get

$$\Delta z = -z + \frac{\mu^2}{x} - \Delta x \frac{z}{x},$$

where the dominant term in the right-hand side of the equation is z because μ^2 and Δx are both small. Rearranging again, we have

$$z + \Delta z = \frac{\mu^2}{x} - \Delta x \frac{z}{x}.$$

Including a stepsize, $\alpha = 1 - \theta\mu$ we obtain

$$z + (1 - \theta\mu)\Delta z = \frac{\mu^2}{x} - \Delta x \frac{z}{x} - \theta\mu\Delta z.$$

Substituting for z in the right-hand side, we obtain

$$z + (1 - \theta\mu)\Delta z = \frac{\mu^2}{x} - \Delta x \frac{(-\Delta z + \frac{\mu^2}{x} - \Delta x \frac{z}{x})}{x} - \theta\mu\Delta z.$$

The terms $\Delta x\mu^2/x^2$ and $(\Delta x)^2 z/x^2$ are both $O(\mu^3)$ so we can simplify to

$$z + (1 - \theta\mu)\Delta z = \frac{\mu^2}{x} - (-\Delta x + \theta\mu)\Delta z + O(\mu^3).$$

When $\theta > \frac{\Delta x}{\mu}$ and μ is sufficiently small, then

$$z + (1 - \theta\mu)\Delta z > 0.$$

2. Given $(\tilde{x}, \tilde{y}, \tilde{z})$ show that the termination criteria for μ^2 are satisfied.

2a. Show $\|c(\tilde{x})\| \leq M\mu^2$.

$$\begin{aligned} \|c(\tilde{x})\| &= \|c + \alpha J\Delta x + O(\Delta x^2)\| \\ &\leq \|c - \alpha c + O(\Delta x^2)\| \\ &\leq \|(1 - \alpha)c + O(\mu^2)\| \\ &\leq \|\theta\mu c + O(\mu^2)\| \\ &\leq M\mu^2. \end{aligned}$$

2b. $\|g(\tilde{x}) - J(\tilde{x})^T\tilde{y} - \tilde{z}\| \leq M\mu^2$.
Define $w(\chi, \psi, \omega) = g(\chi) - J^T(\chi)\psi - \omega$.

A Taylor's expansion of w for $(\tilde{x}, \tilde{y}, \tilde{z})$ gives us

$$w(\tilde{x}, \tilde{y}, \tilde{z}) = w(x, y, z) + \alpha \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}^T \nabla w + O(\mu^2).$$

From solving the Newton equations (6.2) we know

$$w(x + \Delta x, y + \Delta y, z + \Delta z) = w(x, y, z) + \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}^T \nabla w.$$

This gives us

$$\begin{aligned} w(x + \alpha\Delta x, y + \alpha\Delta y, z + \alpha\Delta z) &= w(x, y, z) + (1 - \theta\mu) \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}^T \nabla w + O(\mu^2) \\ &= -\theta\mu \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix}^T \nabla w + O(\mu^2) \\ &= O(\mu^2). \end{aligned}$$

2c $\lambda_{\min}(x + \alpha\Delta x, y + \alpha\Delta y, z + \alpha\Delta z) > 0$ by Lemma (6.3).

2d For the last part, we have

$$|(x_j + \alpha\Delta x_j)(z_j + \alpha\Delta z_j)| \leq M\mu^2$$

since either $x_j + \alpha\Delta x_j$ or $z_j + \alpha\Delta z_j$ is $O(\mu^2)$.

∎

Lemma 6.5. *The initial step taken in the linesearch algorithm (see section 5.2.2) is of the form* $1 - \tau\mu$, $\tau \geq 0$.

Proof. We may assume without loss of generality that $x_i \to 0$ for all i and the unit step is not feasible. The initial step is given by the unique minimizer of the function

$$\vartheta(\alpha) = a\alpha - \mu^2 \sum_{j=1}^{n} \ln(x_j + \alpha\Delta x_j)$$

in the interval $(0,1)$ and $a = \Delta x^T g + c^T y - \frac{\rho}{2} c^T c - c^T \Delta y$. Note that $a < 0$ and $a = O(\mu)$.

Let $\overset{*}{\alpha}$ denote the minimizer. We have

$$\vartheta'(\overset{*}{\alpha}) = a - \mu^2 \sum \frac{\Delta x_j}{x_j + \overset{*}{\alpha}\, \Delta x_j} = 0.$$

This may be rewritten as

$$a - \mu^2 \sum \frac{1}{x_j/\Delta x_j + \overset{*}{\alpha}} = 0.$$

Let $\dfrac{x_r}{\Delta x_r} \leq \dfrac{x_j}{\Delta x_j}$, for all $j = 1, \ldots, n$. It follows $\overset{*}{\alpha} \geq \alpha_{\min}$, where

$$a - \mu^2 n \frac{1}{x_r/\Delta x_r + \alpha_{\min}} = 0.$$

Since $a = O(\mu)$ there exists τ, with $0 < \tau < \infty$ such that

$$-\frac{\Delta x_r}{x_r + \alpha_{\min} \Delta x_r} = \frac{1}{\mu\tau}.$$

Rearranging gives

$$\alpha_{\min} = -\tau\mu - \frac{x_r}{\Delta x_r}.$$

From (6.8) we get

$$\alpha_{\min} = -\tau\mu - (-1 + M x_r \mu + \frac{\mu^2}{z_r}),$$

which gives

$$\alpha_{min} = 1 - \tau\mu + O(\mu^2).$$

Since $1 \geq \overset{*}{\alpha} \geq \alpha_{\min}$ the required result follows. ∎

Corollary 6.6. *The initial step $\overset{*}{\alpha}$ satisfies $\overset{*}{\alpha} \leq 1 - \tau\mu + O(\mu^2)$.*

Corollary 6.7. *The termination conditions for the subproblem are satisfied at the initial step of the linesearch.*

Proof. We have from Lemma 6.5 that $\overset{*}{\alpha}$, the initial step, is of the form

$$\overset{*}{\alpha} = 1 - \tau\mu + (\mu^2). \tag{6.11}$$

The merit function along the search direction $(\Delta x, \Delta y)$ is of the form

$$\psi(\alpha) = \vartheta(\alpha) - \mu^2 \sum \ln(x_j + \alpha \Delta x_j).$$

If there is no index, j, such that $x_j \to 0$ the result follows from Lemma 4.1 in [MP95].

There is no loss of generality if we assume $x_j \to 0$ for all j. We have

$$\vartheta'(\alpha) = a = -\frac{1}{\tau}\mu \quad \text{and} \quad \psi(0) = 0,$$

where $0 < \tau < \infty$. We need to show that

$$\psi(\overset{*}{\alpha}) \;\leq\; -\frac{1}{2\tau}\mu\,\overset{*}{\alpha}\,. \tag{6.12}$$

From Taylor's theorem we have

$$\vartheta(\overset{*}{\alpha}) = \vartheta(0) + \overset{*}{\alpha}\,\vartheta'(0) + O(\mu^2).$$

It follows

$$\vartheta(\overset{*}{\alpha}) \leq \frac{\overset{*}{\alpha}}{\tau}\mu + O(\mu^2).$$

We have

$$-\mu^2 \sum \ln(x_i + \overset{*}{\alpha}\,\Delta x_i) \leq -\beta M \mu^2 \ln \mu^2,$$

where $\beta < \infty$. From (6.11) it follows that if μ is sufficiently small that

$$-\beta M \mu^2 \ln \mu^2 < \frac{1}{2\tau}\mu\,\overset{*}{\alpha}$$

and (6.12) holds.

∎

In practice, a judgement needs to be made when to start reducing μ at a quadratic rate. One option is to do this gradually simply by increasing the reduction in μ (by doubling). Even if a gradual approach is adopted we still need to decide when to increase the reduction. A simple test is when a unit step is taken, which Lemma 6.4 shows will eventually happen.

The following theorem is an immediate consequence of Lemmas 6.4 through 6.7.

Theorem 6.8. *Given an initial point $(\overline{x}_0, \overline{y}_0)$ that satisfies the termination conditions (2.12) for μ sufficiently small then the sequence $\{\overline{x}_k, \overline{y}_k\}$ generated by a single step on each subproblem in which $\mu_{k+1} = \mu_k^2$ converges at a quadratic rate.*

6.2.1 Global convergence

Global convergence follows if we can show that for $\mu \geq \epsilon > 0$ our algorithm satisfies the termination criteria in a finite number of iterations. This follows if the curvilinear search algorithm converges to a solution of the subproblem. In [MP95a] it is shown that the curvilinear search algorithm is shown to converged to a second-order KKT point provided the assumptions 1 to 10 applied to the subproblem hold. The only assumptions

for which there is an issue are assumptions 3 and 9 since the inclusion of the barrier term impacts the properties of the objective. The following lemma shows 3 holds.

Lemma 6.9. *Provided $\mu \geq \epsilon > 0$, the function $f(x) - \mu \sum_{j=1}^{n} \ln x_j \in C3$, for $x \in S_x \cup D$, where S_x is the set of iterates $\{x_k\}$ generated by our algorithm and D is the set $\|x\| \leq \Theta < \infty$.*

Proof. The only term of concern is $\sum \ln x_j$ since $f(x) \in C3$ by assumption 3. Let S denote the set $\{x_k, y_k, z_k\}$ generated by our algorithm. It follows from the definition of our algorithm that for $(x, y, z) \in S$

$$M(x, y, z) < M_0,$$

where M_0 is the initial value of the merit function. Consequently, $\delta > 0$ must exist such that for $x_j \in S_x$

$$x_j \geq \delta > 0.$$

The boundness of $\ln x_j$ and its derivative follows immediately. ∎

The objective in the subproblems is

$$f(x) - \mu \sum_{j=1}^{n} \ln x_j.$$

Consequently, it is this function we require to be bounded below on the feasible set. Since $lim_{x_j \to 0} - \ln x_j = -\infty$ it could be that this property is not true even when $f(x)$ is bounded below. A simple remedy is to add a term of the form $\mu(x - \overline{x}_\mu)^T(x - \overline{x}_\mu)/2$ to the objective. A good choice of $\overline{x}_\mu$ is x_0 and subsequently it can be the point the previous subproblem terminated. The inclusion of this term has no impact on any of the earlier results. To make that complete transparent it is enough if μ is replaced in this term by$\mu 2$).

In the following theorem we assume either Assumption 9 is modified to assume $f(x) - \mu_0 \sum_{j=1}^{n} \ln x_j$ is bounded below or the algorithm is modified from our description to include a term of the form $\mu(x - \overline{x}_\mu)^T(x - \overline{x}_\mu)/2$ added to the objective of the subproblems.

Theorem 6.10. Global Convergence. *Under Assumptions 1-10 applied to the original problem, for any starting point (x_0, y_0), our algorithm converges to (x^*, y^*), a second-order KKT point of problem (1.1).*

∎

Chapter 7

Computational Results

The implementation of our barrier method is called MELBA (MATLAB Experimental Linesearch Barrier Algorithm). This chapter summarizes the results produced by MELBA solving a subset of the CUTE test suite [BCGT95]. The results for MELBA are compared to the results obtained using KNITRO and SNOPT [BHN99, GMS98].

The number of iterations for MELBA is, and should be, primarily compared to KNITRO. KNITRO is a trust region implementation for solving nonlinearly constrained optimization problems using first and second derivatives of the objective and constraint functions. KNITRO applies the Byrd-Omojokun method [Omo91] to solve the barrier subproblem (2.7). Trust-region methods solve first for the step length, or a radius for the next iterate location. KNITRO then solves for both a vertical and a horizontal step within the radius. The trust-region radius is contracted until the required accuracy is obtained for the vertical and horizontal steps. The vertical step should lie within the trust region and makes the linear constraints satisfied. The horizontal step is computed by solving an equality-constrained QP. A merit function is used to determine if the step is accurate enough.

KNITRO, version 3, was run on the Network-Enabled Optimization Server (NEOS) [CMM98]. The default parameters were used for KNITRO, including 10^{-6} for the optimality tolerance and feasibility tolerance. Therefore, when KNITRO terminates successfully the maximum error in the KKT conditions is 10^{-6}. We report the total iterations for KNITRO.

We were also interested in comparing the results of MELBA to a second-order SQP method with proven convergence properties. Our goal was to run the tests on the NEOS server and no method was available at the time of testing (although an experimental version of SNOPT does use second-derivatives). Instead, we compare our solver to SNOPT 6.1 a sparse SQP solver that uses only first-order information to approximate the Hessian of the Lagrangian. The tests were run with SNOPT.

The accuracy controls for SNOPT are the major feasibility tolerance and major optimality tolerance. Both were set to the default value of 10^{-6}. The major feasibility tolerance specifies how accurately the nonlinear constraints are satisfied. The major

optimality tolerance specifies the final accuracy of the dual variables. We report the total major iterations for SNOPT.

KNITRO and SNOPT both used AMPL description files generated by Vanderbei [Vand02].

The barrier algorithm was implemented using MATLAB version 6.5 and the CUTE mex interface [BCGT95] running on a Silicon Graphics Origin 2000. The CUTE mex interface is limited to problems with $n \leq 2010$ and $m \leq 1500$ in order to link in the necessary FORTRAN files. Although not all selected problems are solved successfully, analysis supports that this is caused by insufficient tuning of the parameters for each problem and lack of adequate special-case implementation in MELBA itself. For example, when the Jacobian is not of full rank there may be early termination of MELBA.

MELBA has one parameter that controls the accuracy, namely the final value μ. This was set to be $\mu_S = 10^{-7}$. Success is declared when (2.12) is satisfied with $\mu = \mu_S$.

Please note that there can be few conclusions made comparing iterations for a Quasi-Newton method such as SNOPT to the two barrier methods. The comparison is included to demonstrate that the total iterations are not significantly different. If they were significantly greater it would probably indicate an error.

7.1 Hock-Schittkowski test problems

The Hock-Schittkowski (HS) test set is a collection of small, nonlinear problems [HS81]. Most are included in the CUTE test set. The following problems were excluded:

- 3 non-smooth problems: HS67, HS85, and HS87.
- 2 problems that require external functions: HS68 and HS69.
- 3 problems for which the values returned from the CUTE functions [BCGT95] did not match the Hock-Schittkowski descriptions: HS101, HS102, and HS103.

Unless otherwise noted, the parameters in MELBA were as follows:

- The set of barrier parameters: MU_SET $= [1, 5\text{e}-1, 1\text{e}-1, 5\text{e}-2, 1\text{e}-2, 5\text{e}-3, 1\text{e}-3, 5\text{e}-4, 1\text{e}-4, 1\text{e}-5, 1\text{e}-6, 1\text{e}-7]$.
- The initial value for the non-negative variables t_1 and t_2: INIT_T $= 0.1$.
- The multipliers for the termination conditions (2.12): $\tau_1 = \tau_2 = \tau_3 = 10$ and $\tau_4 = 0$.

The following problems required special parameters to achieve convergence:

- HS15, HS57 used a finer grained MU_SET

- HS18, HS109 had starting $\mu = 10$
- HS96, HS97, HS99, HS114 had starting $\mu = 100$
- HS106 had starting $\mu = 1000$

Also, HS111 and HS111LNP used a simple backtracking algorithm.

Table 7.1 contains the total iterations for MELBA, KNITRO, and SNOPT (major iterations).

TABLE 7.1
Hock-Schittkowski test problems

Name	SNOPT	KNITRO	MELBA
HS1	39	38	35
HS2	12	14	21
HS3	1	4	4
HS4	1	4	5
HS5	6	6	7
HS6	6	8	10
HS7	17	8	26
HS8	5	5	6
HS9	6	6	3
HS10	19	11	21
HS11	10	6	7
HS12	9	7	10
HS14	6	6	10
HS15	6	13	19
HS16	3	7	41
HS17	10	32	29
HS18	17	8	15
HS19	7	19	42
HS20	3	10	13
HS21	4	9	8
HS22	4	6	8
HS23	5	8	17
HS24	3	10	12
HS25	0	33	7
HS26	23	17	25
HS27	20	26	12
HS28	13	2	1
HS29	13	8	10
HS30	3	160	13

TABLE 7.1
Hock-Schittkowski test problems (continued)

Name	SNOPT	KNITRO	MELBA
HS31	7	6	8
HS32	3	9	11
HS33	5	7	14
HS34	6	11	10
HS35	5	8	7
HS36	5	6	35
HS37	8	6	8
HS38	84	49	14
HS39	18	12	128
HS40	6	3	5
HS41	6	7	8
HS42	6	3	10
HS43	8	7	19
HS44	1	8	40
HS45	0	11	19
HS46	25	19	18
HS47	22	17	14
HS48	31	2	1
HS49	31	16	16
HS50	19	8	9
HS51	8	2	1
HS52	8	2	1
HS53	8	4	4
HS54	1	5	6
HS56	10	6	7
HS57	2	7	18
HS59	12	18	12
HS60	9	7	18
HS62	10	5	8
HS63	10	12	10
HS64	23	16	22
HS65	9	17	19
HS66	4	9	9
HS70	14	23	25
HS71	6	7	10
HS72	25	18	145
HS73	5	7	10
HS74	12	12	10

TABLE 7.1
Hock-Schittkowski test problems (continued)

Name	SNOPT	KNITRO	MELBA
HS75	10	11	16
HS77	12	10	18
HS78	6	4	5
HS79	11	4	6
HS80	11	7	9
HS81	7	7	13
HS83	5	9	13
HS84	5	7	42
HS93	31	6	37†
HS95	1	60	36
HS96	1	110	38
HS97	14	49	104
HS98	14	10	52
HS99	10	9	10
HS100	13	8	12
HS100LNP	14	8	11
HS100MOD	14	8	8
HS104	21	14	59
HS105	85	36	24
HS106	12	99	44
HS107	8	‡	16
HS109	30	123	22
HS111	51	10	36
HS111LNP	71	10	43
HS112	22	6	9
HS113	17	9	205
HS114	37	23	50
HS116	22	42	177
HS117	17	17	86
HS118	15	14	32
HS119	15	26	19

‡: Solver did not find solution.

†: Modified bounds from [Vand02] to match Hock-Schittkowski [HS81].

Figure 7.1 summarizes the HS problem results. The number of HS tests completed (y-axis) is plotted for x iterations. The plot shows that SNOPT solves most problems

in fewer iterations than either barrier approach. However, the iterations are within an order of magnitude of each other for most problems.

An Analysis of Variance (ANOVA) and pairwise comparison tests [WM72] were performed on the iteration counts using MATLAB's statistics toolkit. The null hypothesis tested was that the means for the three data sets are the same. This hypothesis could not be rejected with a typical 95% confidence. The ANOVA f statistic was 1.43 with its two degrees of freedom (2, 291), which means there is 24% chance that the results are from the same distribution. The pairwise comparison, likewise, did not demonstrate significantly different means. This is a simple test comparing only the means and standard deviations for the iteration counts. However, it supports the observable results in Figure 7.1.

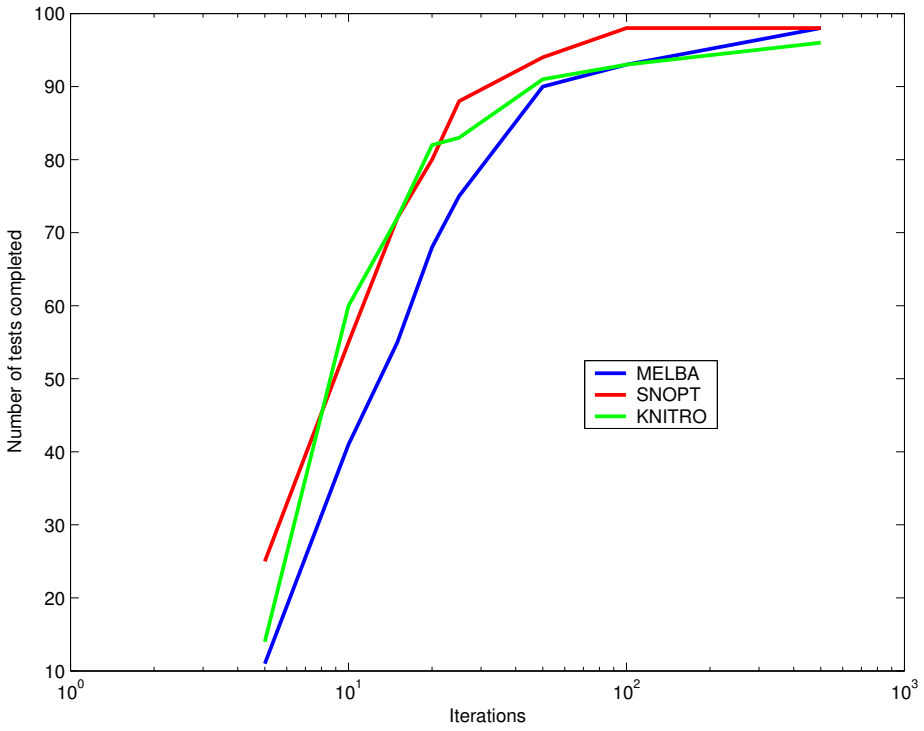

FIGURE 7.1 *Hock-Schittkowski - Number of Iterations*

7.2 Nonlinearly Constrained Problems from CUTE

One of our goals was to demonstrate that MELBA can solve larger problems. The HS test set problems are too small to benefit from the sparse matrix procedures. Using CUTE's select tool, 260 nonlinearly constrained problems were identified in the CUTE test set. Tests that were omitted were:

- 48 tests already run as part of the Hock-Schittkowski test set
- 11 tests with $n > 2010$ and/or $m > 1500$ (BRAINPC0–BRAINPC9, BRIDGEND, TWIRIBG1)
- 47 tests with variable parameters $n > 2010$ and/or $m > 1500$ (CAMSHAPE, CATMIX, CHANNEL, CLNLBEAM, CRESC132, DTOC2, DTOC6, EIGENC, ELEC, GASOIL, GAUSSELM, GLIDER, GPP, LUBRIFC, LUKVLE1–9, LUKVLE11, LUKVLI1–9, LUKVLI11, MARINE, METHANOL, ORTHRDM2, ORTHRGDM, PINENE, POLYGON, READING7, READING8, ROBOTARM, ROCKET, SSNLBEAM, STEERING, TWIRIMD1)

Of these 154 tests, MELBA successfully solved 40 for the set of parameters defined later in this section. Known problems in the results were

- 17 tests had Jacobians without full rank.
- 22 tests, which did not use the reduced gradient method, had singular KKT systems.

Additional work is needed to tune parameters, correctly handle the issues above and, possibly, correct the implementation.

The test problems in Table 7.2 were selected because MELBA solved them and AMPL files existed for comparisons to KNITRO and SNOPT. MELBA certainly did not solve all of the CUTE nonlinear problems, but it did solve more problems than presented here. The columns in Table 7.2 define the number of variables n and constraints m along with the iterations needed to solve the problem.

TABLE 7.2
CUTE-subset test problems

Problem	n	m	SNOPT	KNITRO	MELBA
ALSOTAME	2	1	4	8	7
ARGAUSS	3	15	1	8	3
BROYDNBD	10	10	7	5	11
BT11	5	3	11	7	10
BT6	5	2	14	10	12
BT9	4	2	20	12	128
CB2	3	3	1	10	46
CHACONN2	3	3	22	7	9
CHANDHEQ	10	10	10	10	9
CHEMRCTA	500	500	6	5	9
CHEMRCTB	10	10	4	44	6
CONCON	15	11	6	25	82
DIPIGRI	7	4	13	8	68
DIXCHLNG	10	5	29	8	10
GROWTH	3	12	‡	164	8
HIMMELBK	24	14	7	21	30
INTEGREQ	52	50	2	2	3
LEWISPOL	6	9	0	>1000	8
MADSEN	3	6	1	9	21
MATRIX2	6	2	13	17	18
MCONCON	15	11	6	25	68
METHANB8	31	31	0	290	3
OET2	2007	1002	‡	34	32
POLAK1	3	2	46	10	11
POLAK5	3	2	38	127	16
SPIRAL	3	2	76	165	52

‡: Solver did not find a solution

Note that NEOS upgraded SNOPT to release 6.2 near the end of testing and two tests from this test set, MADSEN and METHANB8, were run with version 6.2.

The parameters used in MELBA to solve this problem set were

- MU_SET = [1, 5e−1, 1e−1, 5e−2, 1e−2, 5e−3, 1e−3, 5e−4, 1e−4, 1e−5, 1e−6].
- The initial value for the non-negative variables t_1 and t_2: INIT_T = 10.
- Multipliers for the termination conditions of (2.12): $\tau_1 = \tau_2 = \tau_3 = 10$ and $\tau_4 = 0$.

Table 7.2 shows a larger range of problem sizes and differences between the solvers. This test set shows that MELBA successfully solves larger problems (e.g., CHEM-

RECTA, OET2) and that it sometimes performs a little better than the other solvers (e.g., GROWTH, OET2 and SPIRAL) and sometimes worse (e.g., CB2).

Figure 7.2 presents the number of tests completed in x iterations. For this graph a maximum iteration count of 300 was used. SNOPT often solves problems in fewer iterations than either other method, but not to a 95% level of significance. ANOVA and multiple comparison tests could not reject the hypothesis that the results are from the same distribution. The ANOVA test indicates a 45% probability that these test results are from the same distribution.

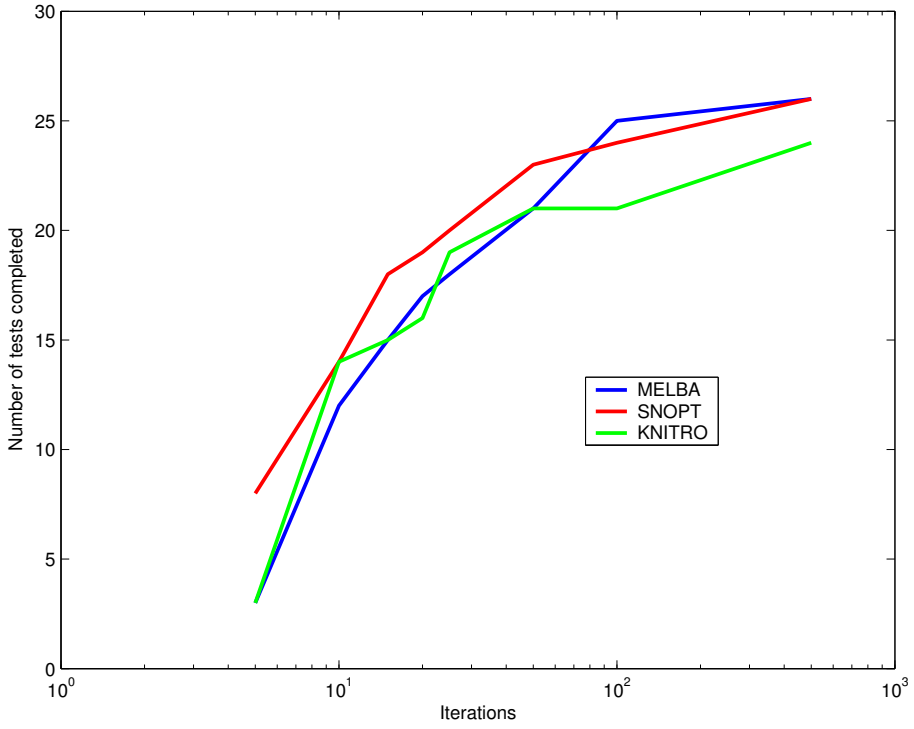

FIGURE 7.2 *CUTE nonlinear test problems – Number of Iterations*

7.3 Linear Constraints

The total number of iterations could probably be reduced if we take into account the existence of linear constraints. SNOPT and MINOS both do this and Lin [Lin02] demonstrated faster convergence for a network barrier method when the starting points satisfied the linear constraints. A specialized LP or QP solver would be needed for the initialization, but no function evaluations would be required. We also may see a reduction partly because our merit function emphasizes satisfying the nonlinear constraints and does not include a measure of feasibility for the linear constraints.

7.4 Conclusions and Future Work

Given the variety and difficulty of nonlinearly constrained optimization problems, it is unlikely that a single best algorithm will emerge. We expect there to be a variety of valuable algorithms, each occupying a niche within the tapestry of real-world applications. How large the proposed algorithm's niche remains to be determined, but we feel it could well be significant. A promising class of problems is those with many degrees of freedom. The reduced-Hessian system would then require an iterative solver, but the theoretical basis of our algorithm does not require an accurate solution of the primal-dual equations, and only one system needs to be solved.

Much needs to be done to improve the efficiency and robustness of the algorithm. Key issues are the selection of starting points for the primal and dual variables, choice of initial barrier parameter, and subsequent adjustment of the barrier parameter according to progress. Theory has little to say about such matters, but poor choices lead to poor performance. To address the most relevant problem class, the core sparse-matrix methods need to be further developed. In particular, the Jacobian basis-selection could be based on sparse LU factors of $\mathbf{X}J^T$, and the reduced Hessian system could be solved by a preconditioned conjugate gradient method. Indefinite reduced Hessians could then be treated as in Nash [Na84], Lin [Lin02] and Ng [Ng02]. Another key issue is when and how best to incorporate elastic variables. As we have mentioned, elastic variables are a remedy for singular Jacobians, but in practice the Jacobian may be merely ill-conditioned and it is not clear at what point this becomes an issue.

Although we have a sophisticated linesearch, we currently use a simple backtracking procedure for the curvilinear search. Both may need improving. The linesearch presupposes we are minimizing a barrier function and that a singularity is encountered somewhere along the search direction. The search does not fail when there is no singularity, but it may be possible to treat this case better.

A feature perhaps worth incorporating is a proximal-point term in the subproblem objective. This was raised as a technical issue in section 6.2.1, but such terms are often beneficial for other reasons and give linesearch methods some of the features of a trust-region method without the need to solve multiple linear systems.

Bibliography

[BSV00] H. Y. Benson, D. F. Hribar, R. J. Vanderbei. 2000. Interior-point methods for nonconvex nonlinear programming: jamming and comparative numerical testing. *Technical Report ORFE–00–02*, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ.

[Bom99] E. G. Boman. 1999. *Infeasibility and Negative Curvature in Optimization.* Ph.D. Dissertation, Stanford University, Stanford, CA.

[BCGT95] I. Bongartz, A. R. Conn, N. Gould, P. Toint. 1995. CUTE: Constrained and unconstrained testing environment, *ACM Trans. Math. Software,* 21, 123–160.

[BHN99] R. Byrd, M. E. Hribar, J. Nocedal. 1999. An interior-point method for large scale nonlinear programming, *SIAM J. Optim.*, 9, 877–900.

[BGN96] R. Byrd, J. C. Gilbert, J. Nocedal. 1996. A trust region method based on interior point techniques for nonlinear programming. *Technical Report OTC-96/2*, Optimization Technology Center, Northwestern University, Evanston, IL.

[Car59] C. W. Carroll. 1959. *An Operations Research Approach to the Economic Optimization of a Kraft Pulping Process.* Ph.D. Dissertation, Institute of Paper Chemistry, Appleton, WI.

[Car61] C. W. Carroll. 1961. The created response surface technique for optimizing nonlinear restrained systems, *Operations Research*, 9:169–184.

[CMM98] J. Czyzyk, M. Mesnier, J. Moré. 1998. The NEOS Server, *IEEE J. on Comp. Science and Engineering*, 5, 68–75.

[Dav02] T. A. Davis. 2003. UMFPACK Version 4.1 User Guide, *Technical Report TR-03-008*, Dept. of Computer and Information Science and Engineering, University of Florida, Gainesville, FL.

[Eld91] S. K. Eldersveld. 1991. *Large-Scale Sequential Quadratic Programming Algorithms,* Ph.D. Dissertation, Stanford University, Stanford, CA.

[FM68] A. Fiacco and G. McCormick. 1968. *Nonlinear Programming*, SIAM, Philadephia, PA.

[FGW02] A. Forsgren, P. E. Gill and M. H. Wright. 2002. Interior methods for nonlinear optimizaton, *SIAM Review*, 44, 4, 525–597.

[FM90] A. Forsgren and W. Murray. 1990. Newton methods for large-scale linear equality-constrained minimization, *Technical Report SOL 90-6*, Department of Operations Research, Stanford University, Stanford, CA.

[Fri55] K. R. Frisch. 1955. The logarithmic potential method of Convex Programming, *Technical Report*, University Institute of Economics, Oslo, Norway.

[GG02] E. M. Gertz and P. E. Gill. 2002. A primal-dual trust region algorithm for nonlinear programming, *Optimization Technical Report 02-09*, UW-Madison CS Department, Madison, WI.

[GMS98] P. E. Gill, W. Murray and M. A. Saunders. 1998. User's Guide for SNOPT version 5.3: A Fortran package for large-scale nonlinear programming, *Technical Report SOL 98-1*, Department of Operations Research, Stanford University.

[GMS02] P. E. Gill, W. Murray and M. A. Saunders. 2002. SNOPT: An SQP Algorithm for large-scale constrained optimization, *SIAM J. Optim.*, 12, 979–1006.

[GMSTW86] P. E. Gill, W. Murray, M. A. Saunders, J. Tomlin and M. H. Wright. 1986. On projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method, *Math. Prog.*, 36, 183–209.

[GMW81] P. E. Gill, W. Murray and M. H. Wright. 1981. *Practical Optimization*, Academic Press, San Diego, CA.

[GV83] G. Golub and C. Van Loan. 1983. *Matrix Computations*, John Hopkins University Press, Baltimore MD.

[Gou85] N. I. M. Gould. 1985. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem, *Math. Prog.*, 18, 31–40.

[HS81] W. Hock and K. Schittkowski. 1981. *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems 187, Springer-Verlag, Berlin, Heidelberg, and New York.

[Hun98] M. S. Hung. 1998. *Optimization With IBM-OSL: Manual*, Scientific Press, South San Francisco, CA.

[ICPLEX] ILOG CPLEX 8.0. 2002. *User's Manual*. ILOG SA, Gentilly, France.

[Kar84] N. Karmarkar. 1984. A new polynomial-time algorithm for linear programming, *Combinatorica,* 4, 373–395.

[KMY89] M. Kojima, S. Mizuno and A. Yoshishe. 1989. A primal-dual interior point algorithm for linear programming, in *Progress in Mathematical Programming: Interior Point and Related Methods*, N. Megiddo, ed., Springer-Verlag, New York, 29–48.

[Lin02] C. Lin. 2002. *A Null-Space Primal-Dual Algorithm For Nonlinear Network Optimization,* Ph.D. Dissertation, Stanford University, Stanford, CA.

[Meg89] N. Megiddo. 1989. Pathways to the optimal set in linear programming, in *Progress in Mathematical Programming: Interior Point and Related Methods*, N. Megiddo, ed., Springer-Verlag, New York, 131–158.

[MP03] J. Moguerza and F. Prieto. 2003. An augmented Lagrangian interior-point method using directions of negative curvature, *Math. Prog.*, A(95), 573–616.

[MS79] J. Moré and D. Sorensen. 1979. On the use of directions of negative curvature in a modified Newton method, *Math. Prog.*, 15, 1–20.

[MOSEK] MOSEK ApS. 2002. *The MOSEK optimization tools version 2.5 (Revision 166) User's manual and reference*, MOSEK ApS, Denmark.

[Mur71] W. Murray. 1971. Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions, *J. Optim. Theory Appl.,* 7, 189–196.

[MP95] W. Murray and F. Prieto. 1995. A second-derivative method for nonlinearly constrained optimization, *Technical Report SOL 95-3*, Department of Operations Research, Stanford University, Stanford, CA.

[MP95a] W. Murray and F. Prieto. 1995. A sequential quadratic programming algorithm using an incomplete solution of the subproblem. *SIAM J. Optim.*, 5, 590–640.

[MW94] W. Murray and M. H. Wright. 1994. Line search procedures for the logarithmic barrier function *SIAM J. Optim.*, 4, 229–246.

[MS78] B. A. Murtagh and M. A. Saunders. 1978. Large-scale linearly constrained optimization, *Math. Prog.,* 14, 41–72.

[MS98] B. A. Murtagh and M. A. Saunders. 1998. MINOS 5.5 User's Guide, *Technical Report SOL 83-20R*, Department of Operations Research, Stanford University, Stanford, CA.

[Na84] S. G. Nash. 1984. Truncated-Newton methods for large scale function minimization, *Applications of Nonlinear Programming to Optimization and Control,*, H. E. Rauch, ed., Pergamon Press, Oxford, 91–100.

[Ng02] K. M. Ng. 2002. *A Continutation Approach For Solving Nonlinear Optimization Problems With Discrete Variables,* Ph.D. Dissertation, Stanford University, Stanford, CA.

[Omo91] E. O. Omojokun. 1991. *Trust Region Algorithms For Optimization With Nonlinear Equations And Inequality Constraints,* Ph.D. Dissertion, University of Colorado.

[Pon90] D. B. Ponceleón. 1990. *Barrier Methods for Large-Scale Quadratic Programming,* Ph.D. Dissertation, Stanford University, Stanford, CA.

[Roc73] R. T. Rockafellar. 1973. The multiplier method of Hestenes and Powell applied to convex programming, *J. Optim Theory Appl.,* 12, 555–562.

[SD00] R. W. Sargent and M. Ding. 2000. A new SQP algorithm for large-scale nonlinear programming, *SIAM J. Optim.*, 11, 716–747.

[SV00] D. Shanno and R. Vanderbei. 2000. Interior-point methods for nonconvex nonlinear programming: orderings and higher order, *Math. Prog.*, 87(2), 303–316.

[Vand02] R. Vanderbei. 2002. Benchmarks for nonlinear optimization, http://www.princeton.edu/~rvdb/bench.html.

[WM72] R. Walpole and R. Myers. 1972. *Probability and Statistics for Engineers and Scientists*, Macmillan Publishing Co., Inc., New York, NY.

[Wri97] S. J. Wright. 1997. *Primal-Dual Interior Point Methods*, SIAM, Philadephia, PA.