A GLOBALLY CONVERGENT LINEARLY CONSTRAINED LAGRANGIAN METHOD FOR NONLINEAR OPTIMIZATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF
MANAGEMENT SCIENCE AND ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
OPERATIONS RESEARCH

Michael P. Friedlander
August 2002

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
(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.

Walter Murray

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.

Philip E. Gill

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

For optimization problems with nonlinear constraints, linearly constrained Lagrangian (LCL) methods sequentially minimize a Lagrangian function subject to linearized constraints. These methods converge rapidly near a solution but may not be reliable from arbitrary starting points. The well known example MINOS has proven effective on many large problems. Its success motivates us to propose a globally convergent variant. Our stabilized LCL method possesses two important properties: the subproblems are always feasible, and they may be solved inexactly. These features are present in MINOS only as heuristics.

The new algorithm has been implemented in MATLAB, with the option to use either the MINOS or SNOPT Fortran codes to solve the linearly constrained subproblems. Only first derivatives are required. We present numerical results on a nonlinear subset from the COPS, CUTE, and HS test-problem sets, which include many large examples. The results demonstrate the robustness and efficiency of the stabilized LCL procedure.

To Anna

Acknowledgments

Professor Michael Saunders has been a wonderful thesis advisor. He has guided and encouraged me, and has pushed me when I needed it—always patiently and with cheerful spirit. He has been generous with his time, his knowledge, and his friendship. I will always remember the example he has set for me. I cannot thank him enough.

I thank Professor Walter Murray for generously offering insightful and critical comments that have helped this research. His constant good cheer is infectious and admirable.

I am grateful to Professor Philip Gill for his careful reading of this dissertation, and for going out of his way to take part in my oral defense. He has made many invaluable comments and has been helpful in countless ways. It is a pleasure to have had the opportunity to get to know him.

Professor Gene Golub is both a mentor and a friend. I wish to thank him for making Stanford a home away from home, and for his strong encouragement throughout.

Thanks also to Professor Richard Cottle, who first introduced me to optimization. I am indebted to him for his strong support and encouragement when I first entered the Ph.D. program at Stanford.

I have enjoyed the support and warm company of my (past and present) colleagues in the Systems Optimization Laboratory: Alexis Collomb, Maureen Doyle, Byunggyoo Kim, Angel-Victor de Miguel, Kien Ming Ng, Lorrie Papadakis, Heiko Pieper, Vinayak "Uday" Shanbhag, and Che-Lin Su.

Thanks to the Dinner Club for many happy evenings and for providing a welcome distraction from academic life. They have stood in for family during my time at Stanford.

Above all, I thank Anna Eberhard Friedlander, my wife and my friend, for her work editing and proofreading so many versions of this dissertation, her unbounded support, and her guidance. Anna gives me something to look forward to each and every day.

Contents

Tables, Figures, and Algorithms

Tables

Figures

Algorithms

Chapter 1

Introduction

For optimization problems with nonlinear constraints, *linearly constrained Lagrangian* (LCL) methods sequentially minimize a Lagrangian function subject to linearized constraints. As currently defined, these methods converge rapidly near a solution but may not be reliable from arbitrary starting points. The well known example MINOS [Murtagh and Saunders, 1982] has proven effective on many large and small problems, especially within the GAMS [Brooke et al., 1988] and AMPL [Fourer et al., 1993] environments, and is widely used in industry and academia. Its success motivates us to propose a globally convergent variant of the LCL method.

Our stabilized LCL algorithm solves a sequence of linearly constrained subproblems. Each subproblem minimizes an augmented Lagrangian function within a linear manifold that describes a current approximation to the nonlinear constraints. This manifold is nominally a linearization of the constraint space, but may be a relaxed (i.e., larger) space at any stage, particularly during early iterations. Few conditions are imposed on the nature of the subproblem solutions; consequently, the subproblems may be solved with any of a variety of optimization routines for linearly constrained problems, providing much flexibility.

The stabilized LCL method possesses two important properties: the subproblems are always feasible, and they may be solved inexactly. These features are present in MINOS only as heuristics.

The stabilized LCL method presented in this dissertation can be regarded as a generalization of sequential augmented Lagrangian methods (see, for example, Gill et al. [1981], Bertsekas [1982], and Fletcher [1987]). The theory we develop provides a framework that unifies Robinson's LCL method [Robinson, 1972] with the bound-constrained Lagrangian (BCL) method used, for example, by LANCELOT [Conn et al., 1991a]. In the context of our theory, the proposed algorithm is actually a continuum of methods, with LCL and BCL methods at opposite ends of a spectrum. The stabilized LCL algorithm exploits this connection between BCL and LCL methods, preserving the fast local convergence properties of LCL methods while inheriting the global convergence properties of BCL methods.

Our focus is on large-scale problems. We implemented the stabilized LCL method using the reduced-gradient part of MINOS [Murtagh and Saunders, 1978] and the sequential quadratic programming code SNOPT [Gill et al., 2002] to solve the linearly constrained subproblems. These solvers are most efficient on problems with few degrees of freedom. Also, they use only first derivatives, and consequently our implementation requires only first derivatives. We discuss how the stabilized LCL method might be used with first- or second-derivative linearly constrained solvers.

1.1 The Optimization Problem

The proposed method solves nonlinearly constrained optimization problems of the form

$$\text{(NP)} \qquad \begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & l \le \begin{pmatrix} x \\ c(x) \\ Ax \end{pmatrix} \le u, \end{array}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a linear or nonlinear objective function, $c : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a vector of constraint functions, A is a matrix, and l and u are vectors of bounds. Because we are considering nonlinearly constrained optimization problems, we assume that at least one of the constraints represented by c is nonlinear. We also assume that A and the derivatives of c are sparse and that the problem (NP) is feasible. Infeasible problems are discussed later.

One of the strengths of the method described herein is that it does not explicitly require second-order information. However, the fast convergence rate of the algorithm relies on sufficient smoothness of the nonlinear functions, indicated by the existence of second derivatives. We make that assumption:

Assumption 1.1. *The functions f and c are twice continuously differentiable on an open neighborhood containing the region*

$$l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u.$$

Note that second derivatives could be used if they were available, thus accelerating the solutions of the subproblems and changing the properties of the solutions obtained by the algorithm. We discuss this further in §3.6.

1.2 The LCL Approach

The acronym "LCL" is new. Methods belonging to this class typically have been referred to in the optimization literature as sequential linearized constraint (SLC) methods (cf. Gill et al. [1981]; Nocedal and Wright [1999]). This term was chosen for compatibility with the terms sequential quadratic programming (SQP) and sequential linear programming (SLP). These methods also sequentially linearize the constraints. However, *linearly constrained Lagrangian* emphasizes that the Lagrangian itself, and not an approximation of it, is used in the subproblems. Moreover, there is a useful relationship (which we exploit) between LCL and BCL methods, and this is hinted at by the nomenclature.

Methods for tackling nonlinearly constrained optimization problems based on solving sequences of linearly constrained (LC) subproblems date back to the method of approximation programming (MAP) [Griffith and Stewart, 1961], one of the first practical algorithms for solving large-scale problems. The MAP approach belongs to the class of SLP methods, and differs from later LCL methods in that it linearizes *both* the objective and constraint functions. A significant drawback of this technique, as its authors point out, is that in order for it to achieve arbitrary accuracy the solution must lie at a vertex of the linearized constraint space. This condition follows necessarily from the nature of the MAP subproblems: at each major iteration, a linear program is solved and only basic solutions are selected. The variable bounds need to be manipulated in order to increase accuracy, but the only known methods for doing so are heuristic. A more recent survey of SLP methods is given by Palacios-Gomez et al. [1982].

The first LCL methods were proposed independently in 1972. Robinson [1972] and Rosen and Kreuser [1972] describe similar algorithms based on minimizing a sequence of Lagrangian functions subject to linearized constraints. Robinson is able to prove that, under suitable conditions, the sequence of subproblem solutions converges quadratically to a solution of (NP). A strength of this method is that efficient large-scale methods exist for the solution of the LC subproblems formed at each iteration. Any suitable example of these subproblem solvers may be called as a black box.

1.3 Other Work on Stabilizing LCL Methods

Some alternative approaches to stabilizing LCL algorithms include two-phase methods proposed by Rosen [1978] and Van Der Hoek [1982]. In this approach, a Phase 1 problem is formed by moving the nonlinear constraints into the objective via a quadratic penalty function. The solution of the Phase 1 problem is used to initialize Robinson's method (Phase 2). With a sufficiently large penalty parameter, the Phase 1 solution will yield a starting point that allows Robinson's method to converge quickly to a solution. However, these two-phase choose the penalty parameter arbitrarily and do not deal methodically with infeasible linearizations.

In a 1981 paper, Best et al. describe a variant of the two-phase method whereby the Phase 1 penalty parameter is gradually increased by repeated return to the Phase 1 problem if the Phase 2 iterations are not converging. This two-phase method differs further from Rosen's and Van Der Hoek's methods in that the Phase 2 iterations only involve equality constraints identified as active by the Phase 1 problem. The authors are able to retain local quadratic convergence of the Phase 2 LCL iterations while proving global convergence to a stationary point. A drawback of their method, however, is that it requires a fourth-order penalty term to ensure continuous second derivatives of the penalty objective. This may introduce significant numerical difficulty for the solution of the Phase 1 problem (though probably a quadratic-penalty term would suffice in practice).

Both two-phase methods share the disadvantage that the Phase 1 penalty problems need to be optimized over a larger subspace than the subsequent LCL phase. We seek a method that retains the linearized constraints as part of the subproblem, in order to keep the number of degrees of freedom small, and, as in Robinson's 1972 method, we allow the subproblem to determine the final set of active constraints.

1.4 The Generic Problem

For the theoretical development of a stabilized LCL method, we consider a simplified, generic formulation of (NP), and take the optimization problem to be

$$\boxed{\text{(GNP)}\qquad \begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) = 0 \\ & \qquad x \geq 0, \end{array}}$$

where $c : \mathbb{R}^n \mapsto \mathbb{R}^m$. This formulation is general enough to accommodate nonlinear inequality constraints—only the appropriate bound-constrained slack variables need be introduced. Chapter 4 returns to the formulation (NP) in its discussion of the implementation of the stabilized LCL method.

We define the *augmented Lagrangian function* corresponding to (GNP) as

$$\mathcal{L}(x, y, \rho) = f(x) - y^T c(x) + \tfrac{1}{2}\rho \|c(x)\|_2^2, \tag{1.1}$$

where x, the m-vector y, and the scalar ρ are independent variables. Note that the usual Lagrangian function is

$$\mathcal{L}(x, y, 0) = f(x) - y^T c(x).$$

For later reference, we write out the expressions for the gradient and Hessian (with respect to x) of $\mathcal{L}$. Let $g(x)$ denote the gradient of the objective function $f(x)$, and $J(x)$ denote the Jacobian matrix of the constraint vector $c(x)$. Denote by $H(x)$ and $H_i(x)$ the Hessian matrices of $f(x)$ and $[c(x)]_i$, respectively, where we use the notation $[\cdot]_i$ to refer to the ith component of a vector. Define

$$\widehat{y}(x, y, \rho) = y - \rho c(x). \tag{1.2}$$

The derivatives of $\mathcal{L}$ with respect to x may be written as follows:

$$\nabla_x \mathcal{L}(x, y, \rho) = g(x) - J(x)^T \widehat{y}(x, y, \rho) \tag{1.3}$$

$$\nabla^2_{xx} \mathcal{L}(x, y, \rho) = H(x) - \sum_{i=1}^{m} [\widehat{y}(x, y, \rho)]_i H_i(x) + \rho J(x)^T J(x). \tag{1.4}$$

We assume that problem (GNP) is feasible and has at least one point (x_*, y_*, z_*) that satisfies the first-order Karush-Kuhn-Tucker (KKT) optimality conditions.

Definition 1.2 (First-order optimality conditions). *A triple (x_*, y_*, z_*) is a first-order KKT point for* (GNP) *if for any $\rho \ge 0$ all of the following hold:*

$$x_* \ge 0 \tag{1.5a}$$

$$z_* \ge 0 \tag{1.5b}$$

$$c(x_*) = 0 \tag{1.5c}$$

$$\nabla_x \mathcal{L}(x_*, y_*, \rho) = z_* \tag{1.5d}$$

$$\min(x_*, z_*) = 0. \tag{1.5e}$$

The methods we examine are iterative—they converge to a solution only in the limit. Let $\eta_* > 0$ and $\omega_* > 0$ be specified as primal and dual convergence tolerances. We regard the point (x, y, z) to be an acceptable solution of (GNP) if it satisfies (1.5) to within these tolerances. Specifically, we identify (x, y, z) as an approximate solution of (GNP) if

$$x \geq 0 \tag{1.6a}$$
$$z \geq -\omega_* e \tag{1.6b}$$
$$\|c(x)\| \leq \eta_* \tag{1.6c}$$
$$\nabla_x \mathcal{L}(x, y, \rho) = z \tag{1.6d}$$
$$\min(x, z) \leq \omega_* e, \tag{1.6e}$$

where e is a vector of ones. In practice, we might also relax (1.6a) to $x \geq -\delta_* e$, for some $\delta_* > 0$. However, we ignore this detail for now.

For theoretical purposes, we need to assume that *strict complementarity* and the *second-order sufficiency conditions* hold at each (x_*, y_*, z_*). We define these conditions as follows.

Definition 1.3 (Strict Complementarity). *The point (x_*, y_*, z_*) satisfies strict complementarity if it satisfies* (1.5) *and either*

$$[x_*]_j > 0 \quad \text{or} \quad [z_*]_j > 0,$$

but not both, for each $j = 1, \ldots, n$.

Definition 1.4 (Second-Order Sufficiency). *The point (x_*, y_*, z_*) satisfies the second-order sufficiency conditions for* (GNP) *if it satisfies* (1.5) *and strict complementarity, and if for any $\rho \geq 0$,*

$$p^T \nabla^2_{xx} \mathcal{L}(x_*, y_*, \rho) p > 0 \tag{1.7}$$

for all $p \neq 0$ satisfying $J(x_*)p = 0$ and $[p]_j = 0$ for all j such that $[x_*]_j = 0$ (and $[z_*]_j > 0$).

Assumption 1.5. *The point (x_*, y_*, z_*) satisfies the second-order sufficiency conditions for* (GNP).

We recognize that not all optimization problems are feasible. This possibility is addressed in §3.5, where we explain how the proposed algorithm reveals an infeasible optimization problem and discuss properties of the points to which it converges.

1.5 Minimizing the Augmented Lagrangian

The augmented Lagrangian was first proposed by Arrow and Solow [1958], and later rediscovered by Hestenes [1969] and Powell [1969]. It is motivated by the following result on quadratic forms.

Theorem 1.6 (Debreu, 1952). *If A is a symmetric matrix and if $x^T Ax > 0$ for every $x \neq 0$ satisfying $Bx = 0$, then, for all ρ sufficiently large, $A + \rho B^T B$ is positive definite.*

Note that $c(x_*) = 0$, so that $\widehat{y}(x_*, y_*, \rho) = y_*$ and

$$\nabla^2_{xx}\mathcal{L}(x_*, y_*, \rho) = H(x_*) - \sum_{i=1}^{m} [y_*]_i H_i(x_*) + \rho J(x_*)^T J(x_*).$$

Set $A = \nabla^2_{xx}\mathcal{L}(x_*, y_*, 0)$ and $B = J(x_*)$. Then, the hypotheses of Debreu's theorem hold under Assumption 1.5, and the theorem implies that $\nabla^2_{xx}\mathcal{L}(x_*, y_*, \rho)$ is positive definite for all ρ sufficiently large. Consequently, if y is close to y_* and ρ is sufficiently large, x_* is an *isolated minimizer* (perhaps bound-constrained) of the augmented Lagrangian. In contrast, x_* might be a saddle point or even a *maximizer* of the Lagrangian function when $\rho = 0$.

The augmented Lagrangian is a vehicle for leveraging well-developed algorithms in the solution of (GNP). It forms the basis for both BCL methods and, in some sense, the LCL method MINOS. We discuss these in more detail in Chapter 2.

1.6 The MINOS Approach

The software package MINOS solves the nonlinearly constrained problem (GNP) by minimizing a sequence of augmented Lagrangian functions subject to linearized constraints. Define the *constraint linearization* and *departure from linearity* functions, at the point x_k, as

$$\begin{aligned} \bar{c}_k(x) &= c(x_k) + J(x_k)(x - x_k) \\ d_k(x) &= c(x) - \bar{c}_k(x). \end{aligned}$$

The MINOS subproblem objective is defined by

$$\widetilde{\mathcal{M}}_k(x, y, \rho) = f(x) - y^T d_k(x) + \tfrac{1}{2}\rho \|d_k(x)\|_2^2.$$

The sequence of LC subproblems generated by MINOS is defined by x_k and y_k (current estimates of x_* and y_*), and a given penalty parameter ρ_k. The subproblems take the form

$$\begin{aligned} &\underset{x}{\text{minimize}} && \widetilde{\mathcal{M}}_k(x, y_k, \rho_k) \\ &\text{subject to} && \bar{c}_k(x) = 0 \quad \text{(linearized constraints)} \\ &&& x \geq 0. \end{aligned} \tag{1.8}$$

The function $\widetilde{\mathcal{M}}_k$ is closely related to the *modified augmented Lagrangian* defined by

$$\mathcal{M}_k(x, y, \rho) = f(x) - y^T d_k(x) + \tfrac{1}{2}\rho \|c(x)\|_2^2. \tag{1.9}$$

Only variables that appear nonlinearly in the constraints also appear nonlinearly in $\widetilde{\mathcal{M}}_k$ (in contrast, all variables x appear nonlinearly in $\mathcal{M}_k$). This can be an advantage for large-scale problems, and is the primary reason why Murtagh and Saunders [1982] use $\widetilde{\mathcal{M}}_k$ rather than $\mathcal{M}_k$.

The augmented Lagrangian and the MINOS Lagrangian function are *equivalent* in a certain vital sense:

$$\widetilde{\mathcal{M}}_k(x, y, \rho) \equiv \mathcal{M}_k(x, y, \rho) \quad \text{if} \quad \bar{c}_k(x) = 0. \tag{1.10}$$

In other words, the MINOS subproblem objective is equivalent to the augmented Lagrangian as long as the linearized constraints are satisfied.

The gradient and Hessian of $\mathcal{M}_k$, with respect to x, are

$$\nabla_x \mathcal{M}_k(x, y, \rho) = g(x) - J(x)^T \widehat{y}(x, y, \rho) + J(x_k)^T y \tag{1.11}$$

$$\nabla^2_{xx} \mathcal{M}_k(x, y, \rho) = H(x) - \sum_{i=1}^{m} [\widehat{y}(x, y, \rho)]_i H_i(x) + \rho J(x)^T J(x), \tag{1.12}$$

where $\widehat{y}(x, y, \rho) = y - \rho c(x)$ was introduced in (1.2). Comparing (1.4) and (1.12), we see that $\nabla^2_x \mathcal{L}(x, y, \rho) \equiv \nabla^2_x \mathcal{M}_k(x, y, \rho)$, so that the curvatures of these Lagrangian functions are identical at all points x.

Empirically, a positive penalty parameter ρ has proven effective in some problems, as it increases the radius of convergence for Robinson's method, but for other problems it has been ineffective. A theoretical understanding of when and how to modify the penalty term has been lacking. In §2.2.2 we discuss why the penalty term alone is not sufficient to induce convergence.

1.7 Notation

The symbol x_* is used in two senses: as a limit point of the sequence $\{x_k\}$, and as the primal solution of (GNP). We distinguish between the two cases when the context is not clear. Denote by $\widehat{g}(x)$ the vector of components of $g(x)$ corresponding to inactive bounds at x_*, so that if $\mathcal{I} = \{i \in 1,\ldots,n \mid [x_*]_i > 0\}$, $\widehat{g}(x) = [g(x)]_{\mathcal{I}}$ (where $[\cdot]_{\mathcal{I}}$ is a shorthand notation for a sub-vector formed from the indices in $\mathcal{I}$). Similarly, let $\widehat{J}(x)$ denote the corresponding columns of the Jacobian matrix.

Unless otherwise specified, the function $\|x\|$ represents the Euclidean-norm of the vector x. When the arguments are vectors, define the function $\min(\cdot,\cdot)$ component-wise. Then, if a and b are n-vectors,

$$\min(a,b) = \begin{bmatrix} \min\big([a]_1,[b]_1\big) \\ \vdots \\ \min\big([a]_n,[b]_n\big) \end{bmatrix}.$$

The following notation is used throughout:

(x,y,z)	primal variables, dual variables, and reduced costs for (GNP)
(x_*,y_*,z_*)	optimal variables for (GNP)
(x_k,y_k,z_k)	the kth estimate of (x_*,y_*,z_*)
(x_k^*,y_k^*,z_k^*)	solution of the kth subproblem
f_k, g_k, c_k, J_k	functions and gradients evaluated at x_k
f_*, g_*, c_*, J_*	functions and gradients evaluated at x_*
$\widehat{g}_k$	inactive gradient components evaluated at x_k
$\widehat{J}_k$	inactive Jacobian columns evaluated at x_k
$\overline{c}_k(x)$	$c_k + J_k(x-x_k)$; the linearization of $c(x)$ at x_k
$d_k(x)$	$c(x) - \overline{c}_k(x)$; the departure from linearity of $c(x)$ at x_k
$\|x\|$, $\|x\|_1$, $\|x\|_\infty$	Euclidean-, ℓ_1-, and ℓ_∞-norm of x
$[x]^-$	$-\min(x,0)$; the negative part of the vector x
$[x]^+$	$\max(x,0)$; the positive part of the vector x
$\widehat{y}(x,y,\rho)$	$y-\rho c(x)$; the first-order multiplier estimate
$\widetilde{y}(x)$	least-squares multiplier estimate; see (2.3)
$B(x,\delta)$	an open neighborhood of x with radius δ

The augmented and modified augmented Lagrangian functions are particularly important for our analysis. We often make use of the shorthand notation

$$\mathcal{L}_k(x) \equiv \mathcal{L}_k(x, y_k, \rho_k) \;= f(x) - y_k^T c(x) + \tfrac{1}{2}\rho_k \|c(x)\|^2 \tag{1.13a}$$
$$\mathcal{M}_k(x) \equiv \mathcal{M}_k(x, y_k, \rho_k) = f(x) - y_k^T d_k(x) + \tfrac{1}{2}\rho_k \|c(x)\|^2, \tag{1.13b}$$

when y_k and ρ_k are fixed. All of the algorithms we discuss are structured around *major* and *minor* iterations. Each major iteration solves a subproblem and generates an element of the sequence $\{(x_k, y_k, z_k)\}$. Under certain (desirable) circumstances, this sequence converges to a solution (x_*, y_*, z_*). For each major iteration k, there is a corresponding set of minor iterations converging to (x_k^*, y_k^*, z_k^*), the solution of the current subproblem. In our development and analysis of a stabilized LCL method, we are primarily concerned with the "outer" level algorithm. Unless stated otherwise, "iterations" refers to major iterations.

Chapter 2

Augmented Lagrangian Methods

In this chapter we discuss two existing approaches for solving the nonlinearly constrained optimization problem: the bound-constrained Lagrangian (BCL) method and the linearly constrained Lagrangian (LCL) method. Both the BCL and LCL methods solve a sequence of subproblems minimizing a Lagrangian function, but the two approaches are quite different, both in their history and in their mathematical properties. The former is expected to converge from arbitrary starting points, but may converge slowly near a solution; the latter converges quickly near a solution, but may fail when started far from a solution. The algorithm proposed in Chapter 3 ties together these two approaches, which have previously been considered entirely disparate.

2.1 Bound-Constrained Lagrangian Methods

BCL methods are motivated by the deficiencies of their predecessors, quadratic-penalty function methods. The relationship between the quadratic-penalty function

$$P(x, \rho) = f(x) + \tfrac{1}{2}\rho\|c(x)\|^2,$$

first proposed by Courant [1943], and the augmented Lagrangian is made clear by the identity

$$P(x, \rho) \equiv \mathcal{L}(x, 0, \rho).$$

A quadratic-penalty function method for solving (GNP) consists of solving the sequence of problems

$$(\mathrm{PP}_k) \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & P(x, \rho_k) \\ \text{subject to} & x \geq 0. \end{array}$$

The sequence of subproblems is defined by a sequence of strictly increasing penalty parameters ρ_k satisfying $\rho_k \to \infty$. Let x_k^* denote a local minimizer of (PP_k). Under fairly mild assumptions, it can be shown that there exists a subsequence defined by $\mathcal{K}$ such that

$$\lim_{k \in \mathcal{K}} x_k^* = x_*$$

(see Proposition 2.2 of Bertsekas, 1982). Fiacco and McCormick [1968] provide an extensive analysis of these methods.

The numerical instabilities associated with penalty-function methods are well known. The convergence of the subproblem solutions to x_* requires increasing ρ_k without bound. As a result, the Hessian of $P(x, \rho_k)$ becomes increasingly ill-conditioned as ρ_k increases, and hence the bound-constrained subproblem (PP_k) becomes difficult to solve. There are methods for circumventing the ill-conditioning and computing accurate solutions of (PP_k); for example, see Murray [1971] and Gould [1989]. Indeed, Murray [1969] and Biggs [1972] show how to derive an SQP method from the penalty-function method. Fletcher [1974] summarizes these approaches. Using the augmented Lagrangian in place of the objective function of (PP_k) and updating a series of multiplier estimates leads to an algorithm that only requires that ρ_k be increased finitely many times.

The danger of ill-conditioning is not eliminated by avoiding large values of ρ_k, however. The Hessian of the augmented Lagrangian may be arbitrarily close to being singular if ρ_k is only slightly larger than the threshold value postulated by Debreu's Theorem (p. 7). BCL methods offer no safeguard against this possibility. LCL methods and the stabilized LCL method (presented in Chapter 3) do not suffer from this defect.

2.1.1 Structure

Like the quadratic-penalty function method, the BCL method solves a sequence of bound-constrained subproblems given by

$$(\mathrm{BC}_k) \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{L}_k(x) \\ \text{subject to} & x \geq 0, \end{array}$$

where the problem is parameterized by the current multiplier estimate y_k and penalty parameter ρ_k. Each solution of the subproblem generates an iterate x_k^*. If the iteration is "successful," an update to y_k is inferred from x_k^*. If the iteration is "unsuccessful," the multiplier estimates are not updated, and the penalty parameter is increased. The power of the BCL method is that an algorithm can be constructed so that the penalty parameter is increased only finitely many times. Under certain conditions, all iterations are eventually successful and the sequence $\{\rho_k\}$ remains bounded.

A version of the BCL algorithm, adapted from Conn et al. [1991b], is shown in Algorithm 1 on the following page. The salient features of this algorithm are that the bound-constrained subproblem needs to be solved only approximately and that no assumption need be made about the boundedness of the multiplier estimates, y_k. Comprehensive analyses of augmented Lagrangian methods are given in Tapia's [1977] paper and Bertsekas's [1982] book. Most of the analysis we rely on is given in more recent papers by Conn et al. [1991b, 1996].

The solver LANCELOT [Conn et al., 1991a] is a large-scale implementation of Algorithm 1. A dense implementation of a BCL method was available for over 10 years as part of the NAG library [Gill et al., 1977].

Convergence tests

The ability to solve the subproblems inexactly, without hindering convergence, is critical for good performance on large problems. Particularly during early major iterations it can yield substantial computational savings. In this section we outline a set of conditions, analogous to (1.6), for determining when a subproblem solution is "near enough" to optimality. Similar stopping criteria are used in the stabilized LCL algorithm presented in Chapter 3.

Definition 2.1 (First-order optimality conditions for (BC$_k$)). *A pair (x, z) is a first-order KKT point for* (BC$_k$) *if the following hold:*

$$x \geq 0 \tag{2.1a}$$

$$z \geq 0 \tag{2.1b}$$

$$\nabla \mathcal{L}_k(x) = z \tag{2.1c}$$

$$\min(x, z) = 0. \tag{2.1d}$$

Note that $\nabla \mathcal{L}_k(x)$ involves y_k and ρ_k.

Algorithm 1: BCL

Input: x_0, y_0, z_0
Output: x_*, y_*, z_*

[Initialize parameters]
Set the initial penalty parameters $\rho_0, \tau > 1$. Set positive convergence tolerances $\omega_*, \eta_* \ll 1$. Set constants $\alpha, \beta > 0$, with $\alpha < 1$;

$k \leftarrow 0$;
$\mathsf{converged} \leftarrow \mathsf{false}$;
repeat
  Choose $\omega_k \geq \omega_*$ such that $\lim_{k \to \infty} \omega_k = \omega_*$;
1.1 **[Solve the BC subproblem]**
    Solve (BC_k) to obtain a pair (x_k^*, z_k^*) that satisfies (2.2);
  if $\|c(x_k^*)\| \leq \max(\eta_k, \eta_*)$ **then**
    [Update solution estimates]
      $x_{k+1} \leftarrow x_k^*$;
1.2 $y_{k+1} \leftarrow y_k - \rho_k c(x_k^*)$; $[\, = \widehat{y}(x_k^*, y_k, \rho_k) \,]$
      $z_{k+1} \leftarrow z_k^*$;
1.3 **[Test convergence]**
      if $(x_{k+1}, y_{k+1}, z_{k+1})$ satisfies (1.6) **then** $\mathsf{converged} \leftarrow \mathsf{true}$;
    $\rho_{k+1} \leftarrow \rho_k$; [keep ρ_k]
1.4 $\eta_{k+1} \leftarrow \eta_k / \rho_{k+1}^{\beta}$; [decrease η_k]
  else
1.5 **[Increase penalty parameter]**
      $\rho_{k+1} \leftarrow \tau \rho_k$;
    [Keep solution estimates]
      $x_{k+1} \leftarrow x_k$;
      $y_{k+1} \leftarrow y_k$;
      $z_{k+1} \leftarrow z_k$;
1.6 $\eta_{k+1} \leftarrow \eta_0 / \rho_{k+1}^{\alpha}$; [may increase or decrease η_k]
  $k \leftarrow k + 1$;
until $\mathsf{converged}$;
$x_* \leftarrow x_k$;
$y_* \leftarrow y_k$;
$z_* \leftarrow z_k$;
return x_*, y_*, z_*;

A relaxed form of (2.1) is used in Algorithm 1. The approximate first-order optimality conditions are given by

$$x \geq 0 \tag{2.2a}$$

$$z \geq -\omega_k e \tag{2.2b}$$

$$\nabla \mathcal{L}_k(x) = z \tag{2.2c}$$

$$\min(x, z) \leq \omega_k e, \tag{2.2d}$$

where $\omega_k \geq 0$ is the kth optimality tolerance. Conditions (2.2) are equivalent to those used in the convergence proofs of Conn et al. [1991b].

At each major iteration of Algorithm 1, the iterate (x_k, y_k, z_k) is tested against conditions (1.6). The algorithm exits as soon as these conditions are satisfied.

Multiplier estimates

The multiplier update in the BCL method is critical for keeping the penalty parameter bounded, and also for accelerating the convergence rates, as discussed in §2.1.3. Powell [1969] and Hestenes [1969] proposed using $\widehat{y}(x_k^*, y_k, \rho_k)$ (see (1.2)) as an update to the multiplier estimates. This is often called the *first-order multiplier update*.

One perspective on this update is as follows. Suppose that (x_k^*, z_k^*) solves (BC_k) for a given y_k and ρ_k, and $\widehat{y}_k \equiv \widehat{y}(x_k^*, y_k, \rho_k)$. Then (x_k^*, z_k^*) satisfies (2.1), and from (1.3),

$$z_k^* = g(x_k^*) - J(x_k^*)^T \widehat{y}_k,$$

so that $(x_k^*, \widehat{y}_k, z_k^*)$ solves the problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & c(x) = c(x_k^*) \\ & x \geq 0. \end{array}$$

Hence if $c(x_k^*) \to 0$, we may expect $\widehat{y}_k$ to converge to y_*. Nash and Sofer [1996, §16.6.1] interpret the update $y_{k+1} = \widehat{y}_k \equiv y_k - \rho_k c(x_k^*)$ as applying the steepest-ascent method to a dual problem of (GNP): $-c(x_k^*)$ is the steepest-ascent direction for that dual, and ρ_k is a fixed step-length. In the stabilized LCL method of Chapter 3 we use a multiplier update analogous to the first-order estimate described here.

Let x_* be any limit point of the sequence $\{x_k^*\}$. At all points x for which $\widehat{J}(x)$ has full row rank (see §1.7 for the definitions of $\widehat{J}(x)$ and $\widehat{g}(x)$), we define the *least-squares multiplier estimate*, $\widetilde{y}(x)$, as the solution of the linear least-squares problem

$$\underset{y}{\text{minimize}} \ \|\widehat{g}(x) - \widehat{J}(x)^T y\|^2. \tag{2.3}$$

Note that the definitions of $\widehat{g}$, $\widehat{J}$, and hence $\widetilde{y}$, require a priori knowledge of the bounds active at x_*. We emphasize that $\widetilde{y}$ is used only as an analytical device—we never require its computation. Assumption 2.3 (stated below) guarantees the uniqueness of $\widetilde{y}$ at every limit point of the sequence $\{x_k^*\}$. Note that $y_* = \widetilde{y}(x_*)$ is the optimal Lagrange multiplier of (GNP) if the least-squares residual in (2.3) is zero, $c_* = 0$, $x_* \geq 0$, and $g_* - J_*^T y_* \geq 0$.

2.1.2 Global convergence properties

For the purposes of this section, we fix the convergence tolerances at zero: $\omega_* = 0$ and $\eta_* = 0$. We make the following assumptions.

Assumption 2.2. *The sequence of iterates $\{x_k^*\}$ lies in the closed and bounded set $\mathcal{B} \subset \mathbb{R}^n$.*

Assumption 2.3. *The matrix $\widehat{J}(x_*)$ has full row rank at every limit point x_* of the sequence $\{x_k^*\}$.*

The first assumption guarantees that any sequence of iterates generated by the algorithm always has some convergent subsequence. The second assumption is commonly known as the *linear independence constraint qualification* (LICQ) (see, for example, Mangasarian [1969], or for a more recent reference, Nocedal and Wright [1999]).

The global convergence properties of the bound-constrained Lagrangian method are readily evident from its relationship to the quadratic-penalty function method. At each failed iteration, the penalty parameter ρ_k is increased (Step 1.5). If this happens infinitely often and y_k remains bounded, the BCL method essentially reduces to a quadratic-penalty function method (cf. Proposition 2.2 of Bertsekas [1982]). As Bertsekas and others have shown, this is a globally convergent method. It is possible to relax the boundedness assumption on y_k. Global convergence can still be assured if we require instead that the quotient $\|y_k\|/\rho_k$ converge to 0. Thus, we require that $\|y_k\|$ not increase faster than the penalty parameter. The updates to η_k given in Steps 1.4 and 1.6 of Algorithm 1 are critical for this result to hold. If we let $\mathcal{K} = \{k_0, k_1, k_2, \ldots\}$ be the index set for all iterations where Step 1.5 is executed, then the updates to η_k

guarantee that the quantity $\sum_{l=1}^{\infty} \eta_{k_v+l}$ remains bounded as $v \to \infty$. This is required for Lemma 2.4 to hold.

Lemma 2.4 (Conn et al., 1991b). *Suppose that $\rho_k \to \infty$ as k increases when Algorithm 1 is executed. Then $\|y_k\|/\rho_k \to 0$.*

Observe that the forcing sequence $\{\eta_k\}$ may not decrease monotonically (see Steps 1.4 and 1.6 in Algorithm 1). However, the sequence is guaranteed to converge to zero, as required for the global convergence property of the algorithm to hold.

Following is the global convergence theorem for Algorithm 1. We prove an analogous theorem for the stabilized LCL algorithm in Chapter 3.

Theorem 2.5. *Let $\{(x_k^*, z_k^*)\}$ be the sequence of iterates generated by Algorithm 1. Let x_* be any limit point of the sequence $\{x_k^*\}$, and $\mathcal{K}$ be the infinite set of indices associated with that convergent subsequence. Suppose that $\{y_k\}$ is any sequence of m-vectors and that Assumptions 1.1, 2.2 and 2.3 hold. Set $y_* = \widetilde{y}(x_*)$. Then*

1. *$\{\widehat{y}_k(x_k^*, y_k, \rho_k)\}$ converges to y_* and $\{z_k^*\}$ converges to $z_* \stackrel{\mathrm{def}}{=} \nabla \mathcal{L}_k(x_*, y_*, 0)$, for $k \in \mathcal{K}$;*

2. *(x_*, y_*, z_*) is a first-order KKT point for (GNP).*

Proof. See Lemma 4.3 and Theorem 4.4 of Conn et al. [1991a]. ■

Finite termination

Note that Algorithm 1 exits *only if* the test in Step 1.3 is executed successfully. This is in contrast to the algorithms proposed by Conn et al. [1991b, 1996], in which the convergence test takes place directly after Step 1.1—i.e., convergence is tested at every iteration—and the algorithm may be terminated at that point. However, this leads to the possibility that the algorithm terminates before the multiplier estimates y_k are updated. Therefore, we choose to test convergence only after y_k is updated.

The following theorem ensures that the BCL algorithm will eventually exit (when the convergence tolerances are positive).

Theorem 2.6. *Suppose that the convergence tolerances ω_* and η_* are positive. Then under the assumptions of Theorem 2.5, Algorithm 1 terminates after a finite number of iterations.*

Proof. Set $\widehat{y}_k = \widehat{y}_k(x_k^*, y_k, \rho_k)$ (see (1.2) for the definition of $\widehat{y}$). Let $\{(x_k^*, z_k^*)\}$, x_*, and $\mathcal{K}$ be as defined in Theorem 2.5. By that theorem,

$$\lim_{k\in\mathcal{K}} \widehat{y}_k = y_*$$
$$\lim_{k\in\mathcal{K}} z_k^* = z_* \stackrel{\text{def}}{=} \nabla_x \mathcal{L}(x_*, y_*, 0),$$

and (x_*, y_*, z_*) is a first-order KKT point for (GNP). Then, (x_*, y_*, z_*) must satisfy (1.6). By the continuity of c, $\lim_{k\in\mathcal{K}} \|c(x_k^*)\| \to c_* = 0$, and because $\eta_* > 0$,

$$\|c(x_k^*)\| < \eta_* \le \max(\eta_k, \eta_*)$$

for all $k \in \mathcal{K}$ large enough. Consequently, Step 1.3 is executed infinitely often and

$$\lim_{k\in\mathcal{K}} (x_k, y_k, z_k) \to (x_*, y_*, z_*).$$

Because $\omega_* > 0$ and $\eta_* > 0$, (x_k, y_k, z_k) satisfies conditions (1.6) for some $k \in \mathcal{K}$ large enough. ■

2.1.3 Local convergence properties

Probably the greatest weakness of BCL methods is their poor local convergence characteristics. The rate of convergence of the sequence $\{x_k\}$ to x_* is determined by the rate of convergence of the sequence $\{y_k\}$ to y_*. The first-order multiplier update, while computationally practical, is responsible for the slow convergence.

Assume for the sake of argument that each bound-constrained subproblem is solved to full accuracy, i.e., $\omega_k \equiv 0$. Bertsekas [1982] shows that, near a solution, the error in the current solution estimate x_k is bounded by the error in the current multiplier estimate y_k according to

$$\|x_k - x_*\| \le M\|y_k - y_*\|/\rho_k,$$

for some $M > 0$. Under appropriate conditions, Bertsekas shows that if $\limsup_{k\to\infty} \rho_k = \rho_* < \infty$ and $y_k \ne y_*$ for all k, then there is some positive $L < 1$ such that

$$\limsup_{k\to\infty} \frac{\|y_{k+1} - y_*\|}{\|y_k - y_*\|} \le L, \tag{2.4}$$

implying a linear convergence rate. If we allow (or force) $\rho_k \to \infty$, then

$$\lim_{k\to\infty} \frac{\|y_{k+1} - y_*\|}{\|y_k - y_*\|} = 0,$$

and the convergence rate becomes superlinear. However, the higher convergence rate incurs the same ill-conditioning as penalty function methods—the very ill-conditioning that BCL methods were designed to avoid.

Improved convergence rates, without the need to drive $\rho_k \to \infty$, can be achieved by using a *second-order multiplier update*,

$$y_{k+1} \leftarrow y_k - B_k^{-1} c(x_k^*), \tag{2.5}$$

where we define $B_k = J(x_k^*)[\nabla^2 \mathcal{L}_k(x_k^*)]^{-1} J(x_k^*)^T$ (see, for example, Bertsekas [1982], Fletcher [1984], and Nash and Sofer [1996]). Bertsekas and Nash and Sofer interpret (2.5) as applying Newton's method to a dual problem of (GNP). This update, however, requires explicit second-derivative information, assumes the Hessian of the Lagrangian is nonsingular, and is much more expensive to compute.

2.2 Linearly Constrained Lagrangian Methods

There are several variations of the LCL method. In this section we outline a general form and discuss some of its properties.

2.2.1 Structure

The LCL method is based on solving the linearly constrained subproblems

$$(\mathrm{LC}_k) \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{M}_k(x) \\ \text{subject to} & \bar{c}_k(x) = 0 \\ & x \geq 0, \end{array}$$

which are parameterized by the latest estimates x_k and y_k, and the current penalty parameter ρ_k. The linear constraints $\bar{c}_k(x) = 0$ use the linearization of c at the point x_k. This is a crucial departure from the BCL method described in §2.1: the BCL subproblem contains only information regarding the latest multiplier estimate, but no information regarding the latest estimate of x_k. In a sense, the BCL method "forgets" at every

Algorithm 2: LCL

Input: x_0, y_0, z_0
Output: x_*, y_*, z_*

[Initialize parameters]
Set the penalty parameter $\rho_0 \geq 0$. Set positive convergence tolerances $\omega_*, \eta_* \ll 1$;

$k \leftarrow 0$;
converged $\leftarrow$ false;
repeat
Choose $\omega_k \geq \omega_*$ such that $\lim_{k\to\infty} \omega_k = \omega_*$;
2.1 **[Solve the LC subproblem]**
Solve (LC_k) to obtain a point (x_k^*, y_k^*, z_k^*) that satisfies (2.8). If there is more than one such point, choose (x_k^*, y_k^*, z_k^*) closest in norm to (x_k, y_k, z_k);
[Update solution estimates]
$x_{k+1} \leftarrow x_k^*$;
$y_{k+1} \leftarrow y_k^*$; [or $y_{k+1} \leftarrow y_k^* - \rho_k c(x_k^*)$]
$z_{k+1} \leftarrow z_k^*$; [or $z_{k+1} \leftarrow g_{k+1} - J_{k+1}^T y_{k+1}$]
2.2 **[Test convergence]**
if $(x_{k+1}, y_{k+1}, z_{k+1})$ satisfies (1.6) **then** converged $\leftarrow$ true;
2.3 $\rho_{k+1} \leftarrow \rho_k$; [leave ρ_k unchanged]
$k \leftarrow k+1$;
until converged;
$x_* \leftarrow x_k$;
$y_* \leftarrow y_k$;
$z_* \leftarrow z_k$;
return x_*, y_*, z_*;

iteration about the previous subproblem solution and searches the entire nonnegative orthant for a new solution. In contrast, the LCL method restricts its search to a subspace (the linearized constraints) defined by the latest estimate of x_k.

Algorithm 2 outlines what we regard to be a canonical LCL method. We note an important relationship between the BCL and LCL subproblem objectives: If $\bar{x}$ satisfies the linearized constraints, $\bar{c}_k(\bar{x}) = 0$, it is evident from the definitions of $\mathcal{L}_k$ and $\mathcal{M}_k$ (see (1.13)) that $\mathcal{L}_k(\bar{x}) \equiv \mathcal{M}_k(\bar{x})$. So it is clear that the BCL and LCL methods are minimizing the same subproblem objectives, but LCL minimizes them over a reduced space.

Note that Step 2.3 of Algorithm 2 leaves the penalty parameter ρ_k unchanged, and so $\rho_k \equiv \rho_0$. The original method introduced by Robinson [1972] sets $\rho_0 = 0$. Each subproblem thus minimizes

$$\mathcal{M}_k(x, y_k, 0) = f(x) - y_k^T d_k(x). \tag{2.6}$$

This is the modified Lagrangian defined by Robinson. We use the augmented modified Lagrangian as the subproblem objective in Algorithm 2 in order to highlight the similarities between the BCL and LCL methods, and also because the stabilized LCL algorithm we develop makes use of this subproblem objective. Moreover, a positive penalty parameter could be used by the LCL method (as it is in MINOS [Murtagh and Saunders, 1982]) and may help convergence from difficult starting points.

Early termination of the LC subproblems

Unlike the BCL subproblem, (LC_k) has a set of constraints and associated multipliers. The solution of the subproblem yields a triple (x_k^*, y_k^*, z_k^*) that can be used to update the major iteration estimates.

Definition 2.7 (First-order optimality conditions for $(\mathbf{LC}_k)$). *A triple (x, y, z) is a first-order KKT point for (LC_k) if the following hold:*

$$\begin{aligned}
x &\geq 0 && \text{(2.7a)}\\
z &\geq 0 && \text{(2.7b)}\\
\bar{c}_k(x) &= 0 && \text{(2.7c)}\\
\nabla\mathcal{M}_k(x) - J_k^T y &= z && \text{(2.7d)}\\
\min(x, z) &= 0. && \text{(2.7e)}
\end{aligned}$$

Note that $\nabla\mathcal{M}_k(x)$ involves x_k, y_k, and ρ_k.

As in the BCL method, we wish to solve the subproblems inexactly. Consistent with relaxed first-order conditions for BCL, we use a relaxed form of (2.7) in Algorithm 2. The approximate first-order optimality conditions are

$$
\begin{aligned}
x &\geq 0 && \text{(2.8a)}\\
z &\geq -\omega_k e && \text{(2.8b)}\\
\bar{c}_k(x) &= 0 && \text{(2.8c)}\\
\nabla\mathcal{M}_k(x) - J_k^T y &= z && \text{(2.8d)}\\
\min(x, z) &\leq \omega_k e, && \text{(2.8e)}
\end{aligned}
$$

where $\omega_k \geq 0$ is the kth optimality tolerance.

Note that we require that the subproblem solver always satisfy the bounds and linear constraints, (2.8a) and (2.8c), exactly. As mentioned with regard to (1.6) (see p. 6), in practice condition (2.8a) might also be relaxed to $x \geq -\delta e$, for some $\delta \geq 0$, and similarly for (2.8c). MINOS and SNOPT [Gill et al., 2002] enforce feasibility of the subproblem in this manner. Normally δ is kept small throughout all major iterations, as bounds and linear constraints can help restrict iterates to the domains of the nonlinear functions. Bounds and linear constraints appear unchanged in the subproblem, and can similarly safeguard the evaluation of the nonlinear functions. If an interior-point method were used for the LC subproblems, only (2.8c) would be relaxed by δ.

2.2.2 Global convergence properties

Robinson notes that his proposed LCL method can be "regarded as a kind of Newton process, although it is not equivalent to a straightforward linearization of the Kuhn-Tucker conditions" [Robinson, 1972]. The LCL algorithm "behaves" like Newton's method: it involves the linearization of (some of) the nonlinear functions; it enjoys a local quadratic convergence rate (in terms of major iterations) under mild conditions; and it may not converge if initialized with a point too far from a solution. This last issue is the focus and subject of this dissertation.

We consider here two particular causes of failure for the LCL method:

- The subproblem constraints may be infeasible, so that the LCL iterations are not defined;
- A near-singular Jacobian J_k (we only assume non-singularity of the Jacobian at

limit points—cf. Assumption 2.3) might lead to an arbitrarily large value of $\|x_k^* - x_k\|$, regardless of the values of y_k and ρ_k in the subproblem objective.

This discussion motivates our stabilized LCL method and modified subproblem, presented in Chapter 3.

Subproblem infeasibility

The LCL method maintains feasibility with respect to the linear constraints and bounds, but any given iteration may be infeasible with respect to the nonlinear constraints. Hence, there is no guarantee that the constraint linearizations are feasible. We consider a simple example. Suppose the constraints for (GNP) are

$$x^2 - x - s = 0 \tag{2.9}$$

$$x, s \geq 0. \tag{2.10}$$

Let $(\bar{x}, \bar{s})$ be the point at which (2.9) is linearized. Enforcing (2.10) implies that the linearized constraint set is feasible if and only if the constraints

$$x(2\bar{x} - 1) \geq \bar{x}^2, \qquad x \geq 0$$

are feasible. The linearized system is inconsistent for all $\bar{x} \in (0, \frac{1}{2}]$.

As shown by Robinson [1972, 1974], feasibility of the subproblem constraints is only guaranteed near a solution. This is in contrast to algorithms such as CONOPT [Drud, 1994], in which (near) feasibility of the nonlinear constraints is maintained at every iteration. In that case, there is a guarantee that the constraint linearizations will always be feasible.

Implied step-length

For some optimization problems, no value of the penalty parameter is sufficient to ensure that MINOS converges to a solution. This may be due in part to the nature of the LC subproblem and the inflexible requirement of satisfying the linearized constraints exactly at every iteration. Let us examine this more closely.

Let the QR factorization of J_k^T be

$$\begin{pmatrix} Y^T \\ Z^T \end{pmatrix} J_k^T = \begin{pmatrix} R \\ 0 \end{pmatrix},$$

so that $J_k Y = R^T$ and $J_k Z = 0$. The solution of the current subproblem defines x_k^*. Define $\Delta x = x_k^* - x_k$ and decompose it as

$$\Delta x = Y \Delta x_Y + Z \Delta x_Z, \tag{2.11}$$

for some vectors Δx_Y and Δx_Z. Because x_k^* must satisfy the linearized constraints,

$$0 = \bar{c}_k(x_k^*) = c_k + J_k \Delta x,$$

or, after rearranging terms, $-c_k = J_k \Delta x$. Premultiplying (2.11) by J_k,

$$-c_k = J_k \Delta x = J_k Y \Delta x_Y = R^T \Delta x_Y. \tag{2.12}$$

If J_k has full row rank, then R is square and nonsingular, and Δx_Y is uniquely determined by c_k. We thus observe that the range space component, $Y \Delta x_Y$, of any feasible step is completely determined by the constraints. Because Y and Z are orthogonal,

$$\|\Delta x\| \geq \|\Delta x_Y\| = \|R^{-T} c_k\|. \tag{2.13}$$

Requiring strict feasibility of the subproblem means that the generated step Δx might be very large and could interfere with global convergence. The only way to control $\|\Delta x\|$ is to modify the constraints (e.g., include a trust-region constraint), or as in certain SQP methods, for example, to employ a merit function to control the magnitude of a step along the direction Δx. We know of no merit function that could control step lengths of the LCL iterations systematically. MINOS makes an effort to control steps of excessive magnitude by heuristically changing Δx to $\alpha \Delta x$ (for some $\alpha < 1$) to ensure that $\|\Delta x\|$ remains reasonable. MINOS applies the same step length α to $\Delta y \equiv y_k^* - y_k$ to control $\|\Delta y\|$ [Murtagh and Saunders, 1982]. In §3.1.1 we show how the stabilized LCL subproblem is able to control $\|\Delta y\|$ systematically by *relaxing* the subproblem linearizations. This is in contrast to trust-region methods, which effectively tighten the linearizations.

To allow the subproblem some degree of flexibility, particularly when the current linearizations are very poor approximations of the true constraints, we define an *elastic* LC subproblem in which an ℓ_1-penalty function on a set of auxiliary elastic variables can be used to relax or enforce the constraint linearizations. Our use of elastic variables coupled with an ℓ_1-penalty function is reminiscent of the method Sℓ_1QP [Fletcher, 1984] and the SQP method SNOPT [Gill et al., 2002]. The elastic variables are integral to the Sℓ_1QP subproblem: it uses them to avoid infeasible subproblems and bound the subproblem multipliers. SNOPT only introduces the elastic variables if it encounters an infeasible subproblem or if $\|y_k\|$ becomes large.

Section 3.1 introduces the elastic LC subproblem and a mechanism for manipulating its parameters in order to achieve global convergence.

2.2.3 Local convergence properties

Robinson [1972] analyzed the local convergence properties of Algorithm 2 under the special case in which $\rho_k \equiv 0$ (cf. (2.6)) and each subproblem is solved to full accuracy (i.e., $\omega_k \equiv 0$). He proved that we can expect fast convergence from a good enough starting point. In particular, under Assumptions 1.1, 1.5, and 2.3, Robinson shows that we can expect an R-quadratic rate of convergence (see Ortega and Rheinboldt [1970] for an in-depth discussion of root-convergence rates). The issues outlined in the previous section do not apply for a sufficiently good starting point. Robinson [1974] proves that the subproblems are always well defined. He also shows that near a solution, the solutions to the LC subproblems, if parameterized appropriately, form a continuous path converging to (x_*, y_*, z_*).

In a later paper, Bräuninger [1977] shows how the fast local convergence rate can be preserved with only *approximate* solutions of the subproblems (again, with $\rho_k \equiv 0$). The subproblems are solved to a tolerance that is tightened at a rate that matches the decrease in the square of the primal and dual infeasibilities. Our proposed LCL algorithm uses a similar strategy.

The local convergence characteristics of the algorithm are not changed when ρ_0 is set to some positive constant, say $\bar{\rho}$. This can be seen by considering the following optimization problem:

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) + \tfrac{1}{2}\bar{\rho}\|c(x)\|^2 \\ \text{subject to} & c(x) = 0, \quad x \geq 0. \end{array} \tag{2.14}$$

The solutions of (2.14) are identical to the solutions of (GNP). The Robinson subproblem objective corresponding to (2.14) is given by

$$R_k(x) \stackrel{\text{def}}{=} f(x) + \tfrac{1}{2}\bar{\rho}\|c(x)\|^2 - y_k^T d_k(x).$$

Then, $\mathcal{M}_k(x) \equiv R_k(x)$ for all k because $\rho_k \equiv \bar{\rho}$. We then observe that applying Algorithm 2 to (GNP), with a penalty parameter $\rho_0 = \bar{\rho}$, is *equivalent* to applying Robinson's original method to problem (2.14). The convergence characteristics of Algorithm 2 are therefore the same as those demonstrated by Robinson [1972]. (However, while the asymptotic convergence rate remains R-quadratic, we expect a different asymptotic error constant.)

We summarize the convergence results in Theorem 2.8 below. Note that the function

$$F(x,y,z) = \begin{bmatrix} [x]^- \\ [z]^- \\ c(x) \\ \nabla_x \mathcal{L}(x,y,\rho) - z \\ \min(x,z) \end{bmatrix}$$

captures the first-order optimality conditions of (GNP), in the sense that $F(x_*,y_*,z_*) = 0$ if and only if (x_*,y_*,z_*) is a first-order KKT point for (GNP). Thus, $\|F(x,y,z)\|$ is a measure of the deviation from optimality. For the next theorem only, define

$$r = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad \text{and} \quad F(r) = F(x,y,z).$$

Theorem 2.8 (Robinson, 1972; Bräuninger, 1977). *Suppose Assumptions* 1.1–1.5 *and* 2.3 *hold at* r_*. *Moreover, suppose* $\omega_k = O(\|F(r_k)\|^2)$ *for all* $k \geq 0$. *Then there is a positive constant* δ *such that if*

$$\|r_0 - r_*\| < \delta,$$

the sequence $\{r_k\}$ *generated by Algorithm* 2 *converges to* r_*. *Moreover, the sequence converges R-quadratically, so that for all* $k \geq 0$,

$$\|r_k - r_*\| \leq Q(\tfrac{1}{2})^{2^k}, \tag{2.15}$$

for some positive constant Q. *Also,*

$$\|r_{k+1} - r_k\| \leq M\|F(r_k)\|, \tag{2.16}$$

for some positive constant M.

Robinson does not state (2.16) as part of a theorem, but it is found in the proof of (2.15). We state it explicitly here because we make use of this relationship in Chapter 3.

Chapter 3

A Stabilized LCL Method

Chapter 2 outlines two methods that form the cornerstones for the algorithm developed in this chapter. In this new algorithm, the linearized constraints of the LCL method are relaxed by a set of elastic variables, and an ℓ_1-penalty function controls the degree of elasticity. The subsequent relaxed LC problem is embedded in an algorithm that parallels the BCL method. We prove that the resulting method—*stabilized LCL*—is globally convergent, and has a local superlinear convergence rate.

3.1 An Elastic LC Subproblem

As suggested in §2.2.2, we modify the linearized constraints used by the LCL method to allow some degree of flexibility in their satisfaction. We introduce a set of nonnegative elastic variables, v and w, into the constraints, and introduce a penalty on these variables into the subproblem objective. The reformulated LC subproblem is

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \mathcal{M}_k(x) + \sigma_k e^T(v+w) \\ \text{subject to} \quad & \bar{c}_k(x) + v - w = 0 \\ & x, v, w \geq 0, \end{aligned} \tag{3.1}$$

where $\mathcal{M}_k$ is as defined on p. 8 and e is a vector of ones. *This elastic subproblem is always feasible.* The point (x_k^*, y_k^*, z_k^*) satisfies the first-order optimality conditions for (3.1) if

$$\|y_k^*\|_\infty \leq \sigma_k \tag{3.2}$$

and (3.16a)–(3.16e) hold (see p. 33). The term $\sigma_k e^T(v+w)$ is the ℓ_1-penalty function, and together with the nonnegativity constraints $v, w \geq 0$ it is equivalent to a penalty on the one-norm of $(v - w)$. However, (3.2) imposes a bound on the size of the subproblem multiplier, and we find later that it is necessary to bound the difference $(y_k^* - y_k)$. We accomplish this by "shifting" the subproblem objective and defining the subproblem as

$$
\boxed{
\begin{array}{lll}
(\mathrm{ELC}_k) \qquad & \underset{x,v,w}{\text{minimize}} & \mathcal{M}_k(x) + y_k^T(v-w) + \sigma_k e^T(v+w) \\
& \text{subject to} & \bar{c}_k(x) + v - w = 0 \\
& & x, v, w \ge 0.
\end{array}
}
$$

The same point (x_k^*, y_k^*, z_k^*) satisfies the first-order optimality conditions for (ELC_k), but the conditions now include

$$
\|y_k^* - y_k\|_\infty \le \sigma_k \tag{3.3}
$$

in place of (3.2). Now σ_k bounds the required quantity.

3.1.1 The ℓ_1-penalty function

The elastic penalty parameter σ_k plays a pivotal role in the stabilized LCL algorithm. It ties together the BCL and LCL methods of the previous chapter by introducing a "dial" that can be used to transform (ELC_k) continuously along a spectrum. At the two extremes of this spectrum lie the BCL and LCL subproblems.

With the definitions of $\mathcal{L}_k$ and $\mathcal{M}_k$ (see (1.13)), the objective of (ELC_k) can be rewritten as

$$
\begin{aligned}
&\mathcal{M}_k(x) + y_k^T(v-w) + \sigma_k e^T(v+w) \\
&\qquad\qquad = \mathcal{L}_k(x) + y_k^T\big(\bar{c}_k(x) + v - w\big) + \sigma_k e^T(v+w).
\end{aligned} \tag{3.4}
$$

Using the linearized constraints $\bar{c}_k(x) + v - w = 0$ to eliminate the middle term on the right-hand side of (3.4), we can write the stabilized LCL subproblem equivalently as

$$
\boxed{
\begin{array}{lll}
(\mathrm{ELC}'_k) \qquad & \underset{x,v,w}{\text{minimize}} & \mathcal{L}_k(x) + \sigma_k e^T(v+w) \\
& \text{subject to} & \bar{c}_k(x) + v - w = 0 \\
& & x, v, w \ge 0.
\end{array}
}
$$

Observe that $(x_k^*, \Delta y_k^*, z_k^*)$, with $\Delta y_k^* \equiv y_k^* - y_k$, satisfies the first-order optimality conditions for (ELC'_k). For any given subproblem of the stabilized LCL method, the penalty term $\sigma_k e^T(v+w)$ may or may not equal zero, indicating that the linearized constraints may not always be satisfied. In contrast, the MINOS or the Robinson LCL subproblems must always satisfy the linearized constraints. Thus, the set of active linearized constraints of the stabilized LCL subproblem is always a subset (though not necessarily strict) of the the usual LCL subproblem. Fletcher [1984] makes the same observation in connection with his Sℓ_1QP method. The global convergence properties of the stabilized LCL method do not require independent constraint gradients or bounded multipliers for each subproblem.

Recovering the BCL subproblem

Set $\sigma_k = 0$. Then (ELC'_k) (which is equivalent to (ELC_k)) reduces to the equivalent bound-constrained minimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{L}_k(x) \\ \text{subject to} & x \geq 0, \end{array} \tag{3.5}$$

where the bounds on the variables v and w have been eliminated because they no longer appear in the objective. The solutions of (3.5) are identical to the solutions of (BC_k), the BCL subproblem (see p. 12).

Recovering the LCL subproblem

The ℓ_1-penalty function is *exact*. For values of σ_k over a certain threshold, v and w will be zero and the minimizers of the elastic problem (ELC_k) will coincide with the minimizers of the inelastic problem (LC_k) (see p. 19). Exact penalty functions have been studied by, among others, Bertsekas [1982], Fletcher [1984], and Luenberger [1984]. See the book by Conn et al. [2000] for a more recent reference.

We are particularly interested in this feature when the iterates generated by the stabilized LCL algorithm are approaching a solution (x_*, y_*, z_*). Recovering the LCL subproblem as the iterates approach a solution ensures that the stabilized LCL method inherits LCL's fast local convergence properties.

In order to prove the existence of some value of σ_k large enough to force the elastic variables to zero, we require two conditions: (i) the inelastic subproblem (LC_k) must satisfy the second-order sufficiency conditions at a solution x_k^*; and (ii) x_k^* must be a regular point. Assumptions 1.5 and 2.3 guarantee that both these conditions are met. For x_k^* close to x_*, Assumption 1.5 guarantees that (LC_k) satisfies the second-order conditions because $\nabla^2_{xx}\mathcal{M}_k \equiv \nabla^2_{xx}\mathcal{L}_k$ (see (1.4) and (1.12)). Assumption 2.3 guarantees the regularity of x_k^* when it is near x_*.

In order to derive a threshold value for σ_k, we first need a technical lemma. The conclusion of Lemma 3.1 is closely related to the usual sensitivity of the optimal objective value to constraint perturbations.

Lemma 3.1. *Suppose that the constraints of* (LC$_k$) *(p. 19) are parameterized by the vector u so that $\bar{c}_k(x) = u$, and that for $u = 0$, (x_k^*, y_k^*, z_k^*) is a local solution satisfying the second-order sufficiency conditions. Moreover, suppose that $J(x_k^*)$ has full row rank. Consider the problem*

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \mathcal{L}_k(x) \\ \text{subject to} & \bar{c}_k(x) = u \\ & \quad x \geq 0, \end{array} \tag{3.6}$$

also parameterized by the vector u. The following properties hold:

1. *There exists some $\epsilon > 0$ such that for every $u \in B(0,\epsilon)$ there is a continuous function $x_*(u)$ that is a solution of* (3.6) *such that $x_*(0) = x_k^*$;*

2. $\nabla_u \mathcal{L}_k(x_*(0)) = y_k^* - y_k$.

Proof. Part 1 follows from Luenberger [1984, p. 313]. Part 2 is proved as follows. From (1.13),

$$\begin{aligned} \nabla_u \mathcal{L}_k(x_*(0)) &= \nabla_u \big(\mathcal{M}_k(x_*(0)) - y_k^T \bar{c}_k(x_*(0))\big) \\ &= \nabla_u \mathcal{M}_k(x_*(0)) - \nabla_u \bar{c}_k(x_*(0)) y_k. \end{aligned} \tag{3.7}$$

Apply the chain rule to derive $\nabla_u \bar{c}_k(x_*(0)) = \nabla_u x_*(0) J_k^T$. Luenberger shows that $\nabla_u \mathcal{M}_k(x_*(0)) = y_k^*$. From (3.7) we then find that

$$\nabla_u \mathcal{L}_k(x_*(0)) = y_k^* - \nabla_u x_*(0) J_k^T y_k. \tag{3.8}$$

For all $u \in B(0,\epsilon)$, $x_*(u)$ is a solution of (3.6), and so

$$u = \bar{c}_k(x_*(u)). \tag{3.9}$$

Differentiating both sides of (3.9) with respect to u and setting $u = 0$ yields

$$I = \nabla_u \bar{c}_k(x_*(0)) = \nabla_u x_*(0) J_k^T. \tag{3.10}$$

Substituting (3.10) into (3.8) gives $\nabla_u \mathcal{L}_k(x_*(0)) = y_k^* - y_k$, thus deriving Part 2 of the lemma, as required. ■

The next lemma establishes the threshold value of σ_k. Lemma 3.2 is almost entirely based on the Exact Penalty Function Theorem of Luenberger [1984, p. 389], but modified to account for the additional term $y_k^T(v - w)$ in the objective of (ELC$_k$).

Lemma 3.2. *Suppose that (x_k^*, y_k^*, z_k^*) satisfies the second-order sufficiency conditions for* (LC$_k$) *(p. 19). Then if $\sigma_k > \|y_k^* - y_k\|_\infty$, (x_k^*, y_k^*, z_k^*) also solves* (ELC$_k$) *(p. 28).*

Proof. Define

$$\begin{aligned} p(u) &= \min_x \{\mathcal{L}_k(x) \mid \bar{c}_k(x) = u,\ x \geq 0\} \\ p_{\sigma_k}(u) &= p(u) + \sigma_k \|u\|_1. \end{aligned}$$

Then,

$$\begin{aligned} &\min_x \Big\{ \mathcal{L}_k(x) + \sigma_k \|\bar{c}_k(x)\|_1 \ \Big|\ x \geq 0 \Big\} \\ &\qquad = \min_{x,u} \Big\{ \mathcal{L}_k(x) + \sigma_k \|u\|_1 \ \Big|\ \bar{c}_k(x) = u,\ x \geq 0 \Big\} \\ &\qquad = \min_u \Big\{ p(u) + \sigma_k \|u\|_1 \Big\} \\ &\qquad = \min_u p_{\sigma_k}(u). \end{aligned} \tag{3.11}$$

Note that $p(0) = p_{\sigma_k}(0)$. If $u = 0$ solves

$$\min_u p_{\sigma_k}(u), \tag{3.12}$$

then (3.11) implies that

$$\min_x \{\mathcal{L}_k(x) + \sigma_k \|\bar{c}_k(x)\|_1 \mid x \geq 0\} = \min_x \{\mathcal{L}_k(x) \mid \bar{c}_k(x) = 0,\ x \geq 0\}. \tag{3.13}$$

By hypothesis, x_k^* solves the right-hand side of (3.13). Now suppose that x_k^* does *not* also solve the left-hand side of (3.13)—i.e., x_k^* does not solve (ELC$'_k$) (see p. 28). Then

$$\mathcal{L}_k(x_k^*) + \sigma_k \|\bar{c}_k(x_k^*)\|_1 > \mathcal{L}_k(x_k^*).$$

But $\bar{c}_k(x_k^*) = 0$, and so the above leads to the contradictory statement that $\mathcal{L}_k(x_k^*) > \mathcal{L}_k(x_k^*)$. Therefore, x_k^* must also solve (ELC$'_k$). Because (ELC$'_k$) is equivalent to (ELC$_k$),

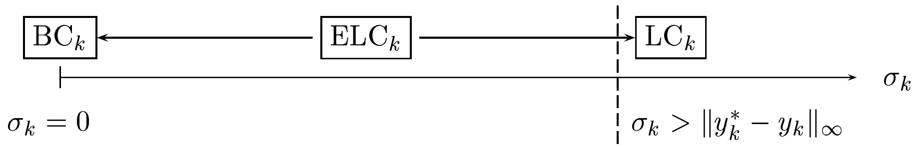

FIGURE 3.1 *The elastic penalty parameter σ_k relates* (ELC_k) *to* (BC_k) *and* (LC_k)

x_k^* also solves (ELC_k). It only remains to show that $u = 0$ solves (3.12). Lemma 3.1 implies that $p(u)$ is continuously differentiable within some neighborhood of 0, and so we can apply the Mean Value Theorem to derive that

$$p(u) = p(0) + \nabla p(\alpha u)^T u$$

for some $\alpha \in [0, 1]$. Then

$$p_{\sigma_k}(u) = p(0) + \nabla p(\alpha u)^T u + \sigma_k \|u\|_1. \tag{3.14}$$

Let δ be any small positive scalar. Because $\nabla p(\cdot)$ is continuous at 0, Lemma 3.1 implies that

$$\|\nabla p(\alpha u)\|_\infty < \|y_k^* - y_k\|_\infty + \delta$$

for every $u \in B(0, \epsilon)$ and $\epsilon > 0$ small enough. Applying the Hölder inequality,

$$\nabla p(\alpha u)^T u \geq -\|\nabla p(\alpha u)\|_\infty \|u\|_1 \geq -(\|y_k^* - y_k\|_\infty + \delta)\|u\|_1.$$

Using this inequality in (3.14),

$$p_{\sigma_k}(u) \geq p(0) + (\sigma_k - \|y_k^* - y_k\|_\infty - \delta)\|u\|_1. \tag{3.15}$$

If $\sigma_k > \|y_k^* - y_k\|_\infty + \delta$, it follows that $u = 0$ minimizes $p_{\sigma_k}(u)$, and because this result holds for any arbitrarily small $\delta > 0$, then it is sufficient that $\sigma_k > \|y_k^* - y_k\|_\infty$. ■

Figure 3.1 summarizes the relationship between (ELC_k) and the BCL and LCL subproblems.

3.1.2 Early termination of the subproblems

The first-order optimality conditions for the stabilized LCL subproblem (ELC_k) (p. 28), are very close to those for the LCL subproblem (LC_k) (see (2.7)). The elastic variables v

and w appear in the linear constraints, and their objective coefficients lead to first-order conditions

$$\begin{aligned}(\sigma_k e + y_k) - y_k^* &\geq 0\\ (\sigma_k e - y_k) + y_k^* &\geq 0,\end{aligned}$$

as already summarized in (3.3). Each solution of the stabilized LCL subproblem yields a 5-tuple $(x_k^*, y_k^*, z_k^*, v_k^*, w_k^*)$.

Definition 3.3 (First-order optimality conditions for (ELC$_k$)). *A 5-tuple (x, y, z, v, w) is a first-order KKT point for* (ELC$_k$) *if the following hold:*

$$x, v, w \geq 0 \tag{3.16a}$$
$$z \geq 0 \tag{3.16b}$$
$$\bar{c}_k(x) + v - w = 0 \tag{3.16c}$$
$$\nabla\mathcal{M}_k(x) - J_k^T y = z \tag{3.16d}$$
$$\min(x, z) = 0 \tag{3.16e}$$
$$\|y - y_k\|_\infty \leq \sigma_k. \tag{3.16f}$$

Note that $\nabla\mathcal{M}_k(x)$ involves x_k, y_k, and ρ_k.

As in the BCL and LCL methods, we wish to solve the subproblems inexactly. We allow (3.16b), (3.16e), and (3.16f) to be violated by an amount ω_k:

$$x, v, w \geq 0 \tag{3.17a}$$
$$z \geq -\omega_k e \tag{3.17b}$$
$$\bar{c}_k(x) + v - w = 0 \tag{3.17c}$$
$$\nabla\mathcal{M}_k(x) - J_k^T y = z \tag{3.17d}$$
$$\min(x, z) \leq \omega_k e \tag{3.17e}$$
$$\|y - y_k\|_\infty \leq \sigma_k + \omega_k. \tag{3.17f}$$

As in the LCL method, each subproblem is required to return a solution satisfying the linear and nonnegativity constraints, and as discussed in connection with (2.8), (3.17a) and/or (3.17c) in practice would be relaxed by a fixed tolerance δ.

3.2 Structure of the Stabilized LCL Method

Algorithm 3 on the next page outlines the stabilized LCL method. Its structure closely parallels the BCL method. Based on the current primal infeasibility, each iteration of the algorithm is regarded as either "successful" or "unsuccessful." In the "successful" case, the solution estimates are updated using information from the current subproblem solution. If the iteration is "unsuccessful," the subproblem solutions are discarded, the current solution estimates are held fixed, and the penalty parameter ρ_k is increased in an effort to reduce the primal infeasibility in the next iteration. In order for the linearized constraints not to continue interfering with the penalty parameter's ability to reduce the primal infeasibility, the algorithm relaxes the linearizations by reducing the elastic penalty parameter σ_k.

The two salient features of this algorithm are that it is globally convergent, and that it is asymptotically equivalent to the LCL method. In the following section we demonstrate the global convergence properties of the algorithm by proving a result similar to Theorem 2.5. In §3.4 we demonstrate that the algorithm eventually reduces to the LCL method, and so inherits that method's asymptotic convergence properties.

3.3 Global Convergence Properties

We need the following lemma to bound the errors in the least-squares multiplier estimates relative to the error in x_k. It simply demonstrates that $\widetilde{y}(x)$ (see (2.3) on p. 16) is Lipschitz continuous in a neighborhood of x_*.

Lemma 3.4. *Let $\{x_k\}$, $k \in \mathcal{K}$ be a subsequence converging to x_* and suppose that Assumptions* 1.1 *and* 2.3 *hold. Then there exists a positive constant α such that $\|\widetilde{y}(x_k) - \widetilde{y}(x_*)\| \le \alpha \|x_k - x_*\|$ for all $k \in \mathcal{K}$ sufficiently large.*

Proof. See Lemmas 2.1 and 4.4 of Conn et al. [1996]. ■

To prove the global convergence properties of Algorithm 3, it is useful to describe first the properties of any limit point it generates. We are not claiming (yet!) that the algorithm is globally convergent. Only that if it *does* converge, then the set of limit points generated must satisfy some desirable properties. The following lemma is adapted from Lemma 4.4 of Conn et al. [1996].

Algorithm 3: Stabilized LCL.

Input: x_0, y_0, z_0
Output: x_*, y_*, z_*

[Initialize parameters]

Set $\overline{\sigma} > \underline{\sigma} > 0$. Set constants $\tau_\rho, \tau_\sigma > 1$. Set the initial penalty parameters $\rho_0 > 1$ and $\sigma_0 \gg 1$. Set positive convergence tolerances $\omega_*, \eta_* \ll 1$ and initial tolerances $\omega_0 > \omega_*$ and $\eta_0 > \eta_*$. Set constants $\alpha, \beta > 0$ with $\alpha < 1$;

$k \leftarrow 0$;
converged $\leftarrow$ false;
repeat

3.1 Choose $\omega_k \geq \omega_*$ such that $\lim_{k\to\infty} \omega_k = \omega_*$;

3.2 **[Solve the LC subproblem]**

Solve (ELC$_k$) to obtain a point (x_k^*, y_k^*, z_k^*) satisfying (3.17). If there is more than one such point, compute the point that is closest in norm to (x_k, y_k, z_k);

3.3 **if** $\|c(x_k^*)\| \leq \max(\eta_*, \eta_k)$ **then**

3.4 **[Update solution estimates]**

$x_{k+1} \leftarrow x_k^*$;
3.5 $y_{k+1} \leftarrow y_k^* - \rho_k c(x_k^*)$; [or $y_{k+1} \leftarrow y_k^*$]
$z_{k+1} \leftarrow z_k^*$; [or $z_{k+1} \leftarrow g_{k+1} - J_{k+1}^T y_{k+1}$]

3.6 **[Update penalty parameter and elastic weight]**

$\rho_{k+1} \leftarrow \rho_k$; [keep ρ_k]
$\sigma_{k+1} \leftarrow \max\{\underline{\sigma}, \min(\|y_k^* - y_k\|_\infty, \overline{\sigma})\}$; [reset σ_k]

3.7 **[Test convergence]**

if $(x_{k+1}, y_{k+1}, z_{k+1})$ satisfies (1.6) **then** converged $\leftarrow$ true;

3.8 $\eta_{k+1} \leftarrow \eta_k / \rho_{k+1}^\beta$; [decrease η_k]

else

3.9 **[Keep solution estimates]**

$x_{k+1} \leftarrow x_k$;
$y_{k+1} \leftarrow y_k$;
$z_{k+1} \leftarrow z_k$;

3.10 **[Update penalty parameter and elastic weight]**

$\rho_{k+1} \leftarrow \tau_\rho \rho_k$; [increase ρ_k]
$\sigma_{k+1} \leftarrow \sigma_k / \tau_\sigma$; [decrease σ_k]

3.11 $\eta_{k+1} \leftarrow \eta_0 / \rho_{k+1}^\alpha$; [may increase or decrease η_k]

$k \leftarrow k + 1$;

until converged;
$x_* \leftarrow x_k$;
$y_* \leftarrow y_k$;
$z_* \leftarrow z_k$;
return x_*, y_*, z_*;

Lemma 3.5. *Let* $\{\omega_k\}$ *and* $\{\rho_k\}$ *be sequences of positive scalars, where* $\omega_k \to 0$. *Let* $\{x_k\}$ *be any sequence of* n*-vectors and* $\{y_k\}$ *be any sequence of* m*-vectors. Let* $\{(x_k^*, y_k^*, z_k^*)\}$ *be a sequence of vectors satisfying* (3.17a), (3.17b), (3.17d), (3.17e). *Let* x_* *be any limit point of the sequence* $\{x_k^*\}$, *and let* $\mathcal{K}$ *be the infinite set of indices associated with that convergent subsequence. Suppose that Assumptions* 1.1, 2.2, *and* 2.3 *hold. Set* $\widehat{y}_k = \widehat{y}(x_k^*, y_k^*, \rho_k)$ *and* $y_* = \widetilde{y}(x_*)$. *The following properties then hold:*

1. *There are positive constants* α_1, α_2, *and* M *such that*

$$\|\widehat{y}_k - y_*\| \le \alpha_1\omega_k + M\|x_k^* - x_k\|\,\|y_k^* - y_k\| + \alpha_2\|x_k^* - x_*\|, \tag{3.18}$$

$$\rho_k\|c(x_k^*)\| \le \alpha_1\omega_k + \|y_k^* - y_k\|\big(M\|x_k^* - x_k\| + 1\big) + \alpha_2\|x_k^* - x_*\| + \|y_k - y_*\|, \tag{3.19}$$

   for all $k \in \mathcal{K}$ *sufficiently large.*

2. *If* $\|y_k^* - y_k\| \to 0$ *as* $k \in \mathcal{K}$ *gets large, or if* $\|y_k^* - y_k\|$ *is bounded and* $\|x_k^* - x_k\| \to 0$ *as* $k \in \mathcal{K}$ *gets large, then*

$$\widehat{y}_k \to y_* \quad \textit{and} \quad z_k^* \to z_* \stackrel{\text{def}}{=} \nabla_x \mathcal{L}(x_*, y_*, 0)$$

   as $k \in \mathcal{K}$ *gets large.*

3. *If, in addition,* $c_* = 0$, *then* (x_*, y_*, z_*) *is a first-order KK point for* (GNP).

Proof. From the definition of $\widetilde{y}(x_k^*)$, the least-squares multiplier estimates,

$$\begin{aligned}
\|\widetilde{y}(x_k^*) - \widehat{y}_k\| &= \|(\widehat{J}(x_k^*)\widehat{J}(x_k^*)^T)^{-1}\widehat{J}(x_k^*)\widehat{g}(x_k^*) - \widehat{y}_k\| \\
&= \|(\widehat{J}(x_k^*)\widehat{J}(x_k^*)^T)^{-1}\widehat{J}(x_k^*)(\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T\widehat{y}_k)\| \\
&\le \|(\widehat{J}(x_k^*)\widehat{J}(x_k^*)^T)^{-1}\widehat{J}(x_k^*)\| \cdot \|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T\widehat{y}_k\|.
\end{aligned} \tag{3.20}$$

By assumption, $\widehat{J}(x_*)$ has full row rank. Continuity of J then implies that

$$(\widehat{J}(x_k^*)\widehat{J}(x_k^*)^T)^{-1}\widehat{J}(x_k^*)$$

exists for all $k \in \mathcal{K}$ large enough. Then there exists some positive scalar α_1 such that

$$\|(\widehat{J}(x_k^*)\widehat{J}(x_k^*)^T)^{-1}\widehat{J}(x_k^*)\| \le \frac{\alpha_1}{\sqrt{n}}, \tag{3.21}$$

where n is the dimension of the vector x. Substituting (3.21) into (3.20),

$$\|\widehat{y}(x_k^*) - \widehat{y}_k\| \le \frac{\alpha_1}{\sqrt{n}} \|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T \widehat{y}_k\|. \tag{3.22}$$

We now show that $\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T \widehat{y}_k\|$ is bounded. By hypothesis, (x_k^*, y_k^*, z_k^*) satisfies (3.17d). Together with (1.11),

$$\begin{aligned} z_k^* &= \nabla \mathcal{M}_k(x_k^*) - J_k^T y_k^* \\ &= g(x_k^*) - J(x_k^*)^T (y_k - \rho_k c(x_k^*)) + J_k^T y_k - J_k^T y_k^* \\ &= g(x_k^*) - J(x_k^*)^T \big(y_k^* - \rho_k c(x_k^*)\big) + \big(J(x_k^*) - J_k\big)^T (y_k^* - y_k) \\ &= g(x_k^*) - J(x_k^*)^T \widehat{y}_k + \big(J(x_k^*) - J_k\big)^T (y_k^* - y_k), \end{aligned} \tag{3.23}$$

where $\widehat{y}_k \overset{\text{def}}{=} \widehat{y}(x_k^*, y_k^*, \rho_k) = y_k^* - \rho_k c(x_k^*)$. For $k \in \mathcal{K}$ large enough, x_k^* is sufficiently close to x_* so that

$$\|[z_k^*]_{\mathcal{I}}\| \le \|\min(x_k^*, z_k^*)\|, \tag{3.24}$$

where $\mathcal{I}$ is the index set of inactive bounds at x_k^*, as defined on p. 9. Because x_k^* and z_k^* both satisfy (3.17e), (3.24) implies that

$$\|[z_k^*]_{\mathcal{I}}\| \le \sqrt{n}\ \omega_k. \tag{3.25}$$

Combining (3.23) and (3.25),

$$\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T \widehat{y}_k + \big(\widehat{J}(x_k^*) - \widehat{J}_k\big)^T (y_k^* - y_k)\| \le \sqrt{n}\ \omega_k. \tag{3.26}$$

But, from the triangle and Cauchy-Schwartz inequalities, we have

$$\begin{aligned} \|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T \widehat{y}_k\| \le \|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T \widehat{y}_k + \big(\widehat{J}(x_k^*) - \widehat{J}_k\big)^T (y_k^* - y_k)\| \\ + \|\widehat{J}(x_k^*) - \widehat{J}_k\| \|y_k^* - y_k\|. \end{aligned} \tag{3.27}$$

Also, the continuity of J implies that there exists a positive constant M such that $\|\widehat{J}(x_k^*) - \widehat{J}_k\| \le M \frac{\sqrt{n}}{\alpha_1} \|x_k^* - x_k\|$. Together, (3.27) and (3.26) imply that

$$\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T \widehat{y}_k\| \le \sqrt{n}\ \omega_k + M \frac{\sqrt{n}}{\alpha_1} \|x_k^* - x_k\| \|y_k^* - y_k\|, \tag{3.28}$$

and so we have derived a bound on $\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T \widehat{y}_k\|$, as required.

We now derive (3.18). From the triangle inequality,

$$\|\widehat{y}_k - y_*\| \le \|\widetilde{y}(x_k^*) - \widehat{y}_k\| + \|\widetilde{y}(x_k^*) - y_*\|. \tag{3.29}$$

Using inequality (3.28) in (3.22), we deduce that

$$\|\widetilde{y}(x_k^*) - \widehat{y}_k\| \le \alpha_1\omega_k + M\|x_k^* - x_k\|\|y_k^* - y_k\|, \tag{3.30}$$

and Lemma 3.4 implies that there exists a constant α_2 such that

$$\|\widetilde{y}(x_k^*) - y_*\| \le \alpha_2\|x_k^* - x_*\|, \tag{3.31}$$

for all $k \in \mathcal{K}$ large enough (recall that $y_* \equiv \widetilde{y}(x_*)$). Substituting (3.30) and (3.31) into (3.29), we derive

$$\|\widehat{y}_k - y_*\| \le \alpha_1\omega_k + M\|x_k^* - x_k\|\,\|y_k^* - y_k\| + \alpha_2\|x_k^* - x_*\|, \tag{3.32}$$

as required. We now prove (3.19). From the definition of the first-order multiplier estimates, rearranging terms yields

$$\rho_k c(x_k^*) = y_k^* - \widehat{y}_k. \tag{3.33}$$

Taking norms of both sides of (3.33) and using (3.32) yields

$$\begin{aligned}\rho_k\|c(x_k^*)\| &= \|y_k^* - \widehat{y}_k\| \\ &= \|y_k - y_* + y_* - \widehat{y}_k + y_k^* - y_k\| \\ &\le \|\widehat{y}_k - y_*\| + \|y_k - y_*\| + \|y_k^* - y_k\| \\ &\le \alpha_1\omega_k + M\|x_k^* - x_k\|\,\|y_k^* - y_k\| + \alpha_2\|x_k^* - x_*\| + \|y_k - y_*\| + \|y_k^* - y_k\|,\end{aligned} \tag{3.34}$$

and so Part 1 of Lemma 3.5 is proved.

Now suppose that $\|y_k^* - y_k\| \to 0$ as $k \in \mathcal{K}$ goes to infinity. Because $\{x_k^*\}$ and $\{x_k\}$ are in the compact set $\mathcal{B}$, $\|x_k^* - x_k\|$ is bounded. We can conclude from (3.18) that $\widehat{y}_k \to y_*$ as $k \in \mathcal{K}$ goes to infinity. We can also conclude from the continuity of J that $\|J(x_k^*) - J_k\|$ is bounded, so that

$$\lim_{k\in\mathcal{K}} \|(J(x_k^*) - J_k)^T(y_k^* - y_k)\| = 0. \tag{3.35}$$

On the other hand, suppose that $\|y_k^* - y_k\|$ is uniformly bounded and that $\lim_{k\in\mathcal{K}} \|x_k^* - x_k\| = 0$. We then conclude from (3.18) that $\widehat{y}_k \to y_*$ as $k \in \mathcal{K}$ goes to infinity and (3.35) holds. Because $\lim_{k\in\mathcal{K}}(x_k^*, \widehat{y}_k) = (x_*, y_*)$,

$$g(x_k^*) - J(x_k^*)^T\widehat{y}_k \to g_* - J_*^T y_*,$$

and so (3.23) and (3.35) together imply that

$$z_k^* \to z_* \equiv \nabla_x\mathcal{L}(x_*, y_*, 0) \tag{3.36}$$

as $k \in \mathcal{K}$ goes to infinity. Thus we have proved Part 2 of Lemma 3.5.

Now suppose that

$$c_* = 0. \tag{3.37}$$

Each x_k^* and z_k^* satisfies (3.17a), (3.17b), and (3.17e). Then $\lim_{k\in\mathcal{K}}(x_k^*, z_k^*) = (x_*, z_*)$ and $\omega_k \to 0$ implies that

$$\begin{aligned} x_* &\geq 0 \\ z_* &\geq 0 \\ \min(x_*, z_*) &= 0. \end{aligned} \tag{3.38}$$

Therefore, (3.36)–(3.38) imply that (x_*, y_*, z_*) satisfies (1.5) and so is a first-order KKT point for (GNP). Part 3 is thus proved, and the proof of Lemma 3.5 is complete. ■

The conclusions of Lemma 3.5 pertain to any sequence $\{x_k^*, y_k^*, z_k^*\}$ satisfying the approximate first-order conditions (3.17). Algorithm 3 generates such a sequence, and also generates auxiliary sequences of scalars $\{\omega_k\}$, $\{\rho_k\}$, and $\{\sigma_k\}$ in such a way as to guarantee that the hypotheses of Lemma 3.5 hold. Every limit point of the sequence $\{x_k^*, y_k^*, z_k^*\}$ is therefore a first-order KKT point for (GNP).

Before laying out the global convergence properties of the stabilized LCL method, we need to show that Lemma 2.4 also applies to Algorithm 3.

Lemma 3.6. *Suppose that $\rho_k \to \infty$ as k increases when Algorithm 3 is executed. Then $\|y_k\|/\rho_k \to 0$.*

Proof. Lemma 2.4 relies only on the multiplier update (Step 3.5) and the construction of the forcing sequence η_k (Steps 1.4 and 1.6) found in Algorithm 1. It guarantees that $\|y_k\|/\rho_k \to 0$ when the multiplier update of Algorithm 1 is used. Note that Algorithm 3 constructs η_k in the same manner. Moreover, note that the norm of the difference between the multiplier updates used in the two algorithms is given by $\|y_k^* - y_k\|$ (see

Steps 1.2 and 3.5). This difference is bounded because y_k^* and y_k satisfy (3.17f). Therefore, $\|y_k\|/\rho_k \to 0$ in Algorithm 3 as $\rho_k \to \infty$. ■

Set the convergence tolerances $\omega_* = 0$ and $\eta_* = 0$.

Theorem 3.7. *Let $\{(x_k^*, y_k^*, z_k^*)\}$ be the sequence of vectors generated by Algorithm* 3. *Let x_* be any limit point of the sequence $\{x_k^*\}$ and let $\mathcal{K}$ be the infinite set of indices associated with that convergent subsequence. Then, under the assumptions of Lemma* 3.5, *Parts* 1, 2, *and* 3 *of that lemma hold.*

Proof. Algorithm 3 generates positive scalars ρ_k, and by Step 3.1 and Steps 3.8 and 3.11, positive scalars $\omega_k \to 0$ and $\eta_k \to 0$. Step 3.2 of the algorithm generates a sequence $\{x_k^*, y_k^*, z_k^*\}$ satisfying (3.17). Then, the hypotheses of Lemma 3.5 hold, and Part 1 of the lemma follows immediately.

First, note that each x_k^* satisfies (3.17), and so $x_k^* \geq 0$ for all k. Thus, $x_* \geq 0$. Moreover, because $\tau_\sigma > 1$ and $\overline{\sigma}$ is finite, Steps 3.6 and 3.10 of Algorithm 3 ensure that σ_k is uniformly bounded. We then need to consider the four possible cases. For all k,

1. ρ_k is uniformly bounded, and $\sigma_k \to 0$ as k gets large;
2. ρ_k is uniformly bounded, and σ_k is uniformly bounded away from zero;
3. $\rho_k \to \infty$ and $\sigma_k \to 0$ as k gets large;
4. $\rho_k \to \infty$ and σ_k is uniformly bounded away from zero.

We dismiss Case 1 because it cannot be generated by the algorithm. (As k gets large, $\sigma_k \to 0$ only if Step 3.10 is executed infinitely many times, contradicting the finiteness of ρ_k.)

Case 2 implies that Step 3.6 of Algorithm 3 is executed for all k large enough. Thus, $x_{k+1} = x_k^*$ for all large k, so that $x_k^* \to x_*$ implies $x_k \to x_*$. Therefore, $\|x_k^* - x_k\| \to 0$. Because each y_k^* satisfies (3.17f),

$$\|y_k^* - y_k\|_\infty \leq \omega_k + \sigma_k. \tag{3.39}$$

Because σ_k and ω_k are uniformly bounded, Part 2 of Lemma 3.5 holds. In addition, $\|c(x_k^*)\| \leq \eta_k$ for all k large enough, and so $\eta_k \to 0$ implies that $c(x_k^*) \to 0$. By continuity of c, $c_* = 0$. Thus, Part 3 of Lemma 3.5 holds.

Now consider Case 3. Because $\sigma_k \to 0$ and $\omega_k \to 0$, (3.39) implies that $\|y_k^* - y_k\| \to 0$ as $k \in \mathcal{K}$ increases. Then Part 2 of the lemma holds. To show that $c(x_k^*) \to 0$, divide both sides of (3.19) by ρ_k to obtain

$$\|c(x_k^*)\| \leq \underbrace{\frac{\alpha_1 \omega_k}{\rho_k}}_{(a)} + \underbrace{\frac{1}{\rho_k}\|y_k^* - y_k\|\big(M\|x_k^* - x_k\| + 1\big)}_{(b)} + \underbrace{\frac{\alpha_2}{\rho_k}\|x_k^* - x_*\|}_{(c)} + \underbrace{\frac{1}{\rho_k}\|y_k - y_*\|}_{(d)}.$$

Term (a) clearly goes to zero as ρ_k increases. Because $\|y_k^* - y_k\| \leq \sigma_k + \omega_k$, and because x_k^* and x_k belong to the compact set B, (b) and (c) go to zero as ρ_k increases. By Lemma 3.6, $\|y_k\|/\rho_k \to 0$, and so (d) goes to 0. We conclude that $\|c(x_k^*)\| \to 0$ as $k \in \mathcal{K}$ increases, as required.

Case 4 is a situation in which both Steps 3.6 and 3.10 are executed infinitely often. But because σ_k is uniformly bounded, so is $\|y_k^* - y_k\|$. Using the same arguments as in Case 2, $\|x_k^* - x_k\| \to 0$ as $k \in \mathcal{K}$ get large, and so Part 2 of Lemma 3.5 holds. The rest of the analysis for this Case is the same as for Case 3. ■

Note that the convergence test takes place only if Step 3.3 of Algorithm 3 tests true; i.e., if $\|c(x_k^*)\| \leq \eta_k$ (because $\eta_* = 0$). In order to guarantee that the algorithm will eventually terminate as the iterates x_k, y_k, and z_k converge, we need to guarantee that Steps 3.4 and 3.7 execute infinitely often. The forcing sequence η_k is intimately tied to this occurrence. For example, if $\eta_k \equiv 0$, then we would not normally expect Step 3.3 to evaluate true (except in rare occasions when $c(x_k^*) = 0$). The forcing sequence defined by Steps 3.8 and 3.11 of Algorithm 3 is suggested by Conn et al. [1991b, 1996]. The following corollaries show that this forcing sequence has the desired property and summarize the global convergence properties of Algorithm 3. Unlike for the previous results in this section, we now need to strengthen our assumptions and require that only a single limit point exist.

Corollary 3.8 (Global convergence). *Let $\{(x_k^*, y_k^*, z_k^*)\}$ and $\{(x_k, y_k, z_k)\}$ be the sequence of vectors generated by Algorithm 3. Let x_* be the single limit point of the sequence $\{x_k^*\}$. Suppose that Assumptions 1.1, 2.2, and 2.3 hold. Then*

$$\lim_{k \to \infty} (x_k, y_k, z_k) = (x_*, y_*, z_*)$$

and (x_*, y_*, z_*) is a first-order KKT point for (GNP).

Proof. Set $\widehat{y}_k = \widehat{y}(x_k^*, y_k^*, \rho_k)$. By Lemma 3.5 and Theorem 3.7,

$$\lim_{k\to\infty} \widehat{y}_k = y_* \quad \text{and} \quad \lim_{k\to\infty} z_k^* = z_*.$$

Moreover, (x_*, y_*, z_*) is a first-order KKT point for (GNP). Suppose Step 3.4 is executed infinitely often. The result then follows immediately because x_k, y_k, and z_k are updated infinitely often and form a convergent sequence from x_k^*, $\widehat{y}_k$, and z_k^*.

We now show by contradiction that Step 3.4 does occur infinitely often. Suppose instead that it does not. Then there exists a k_1 large enough so that Steps 3.9 and 3.10 are executed for all $k > k_1$. Consider only iterations $k > k_1$. Then $y_k \equiv \bar{y}$ and $\rho_k \to \infty$. From (3.33),

$$\begin{aligned} \rho_k \|c(x_k^*)\| &= \|y_k^* - \widehat{y}_k\| \\ &= \|(y_k^* - \bar{y}) + \bar{y} - \widehat{y}_k\| \\ &\le \|y_k^* - \bar{y}\| + \|\bar{y}\| + \|\widehat{y}_k\|. \end{aligned} \tag{3.40}$$

The vector y_k^* satisfies (3.17f), and so $\|y_k^* - \bar{y}\| \le \sigma_k + \omega_k$. Moreover, $\lim_{k\to\infty} \widehat{y}_k = y_*$ and y_* is bounded (see Assumption 2.3). Then, from (3.40), there exists some constant $L > 0$, independent of k, such that

$$\rho_k \|c(x_k^*)\| \le L \tag{3.41}$$

for all k. But the test at Step 3.3 fails at every iteration, so that

$$\eta_k < \|c(x_k^*)\|. \tag{3.42}$$

Combining (3.41) and (3.42), it must be that

$$\rho_k \eta_k < \rho_k \|c(x_k^*)\| \le L. \tag{3.43}$$

From Step 3.11, $\eta_{k+1} = \eta_0 / \rho_{k+1}^\alpha$, so that

$$\rho_k \eta_k = \rho_k \frac{\eta_0}{\rho_k^\alpha} = \eta_0 \rho_k^{1-\alpha}. \tag{3.44}$$

Substituting (3.44) into (3.43), we find that

$$\eta_0 \rho_k^{1-\alpha} < L,$$

for all k. This is a contradiction under the hypothesis that $\alpha < 1$ and $\rho_k \to \infty$.

Therefore, Step 3.4 must occur infinitely often. ■

The following result simply asserts that Algorithm 3 will eventually exit when, as in practice, ω_* and η_* are positive.

Corollary 3.9 (Finite Termination). *Suppose that the convergence tolerances ω_* and η_* are strictly positive. Then, under the assumptions of Corollary* 3.8, *Algorithm* 3 *terminates after a finite number of iterations.*

Proof. This result follows from Corollary 3.8. The proof is analogous to the proof of Theorem 2.6. ■

3.4 Local Convergence Properties

BCL methods retain global convergence while LCL methods can retain fast local convergence under inexact solutions to the subproblems. The stabilized LCL algorithm combines these two desirable features.

Bertsekas [1982] and Conn et al. [1991b, 1996] show how to construct the forcing sequence $\{\eta_k\}$ to guarantee that $\|c(x_k^*)\| \le \eta_k$ will eventually always be true so that the iterates x_k, y_k, and z_k are updated (see Step 3.4 of Algorithm 3) for all iterations after some k large enough. The penalty parameter ρ_k then remains uniformly bounded—an important property. These results rely on a relationship between $\|c(x_k^*)\|$ and ρ_k, namely (3.19). We know from the BCL convergence theory that the convergence rate approaches superlinear as ρ_k grows large. As long as η_k is reduced at a rate slower than superlinear, then $\|c(x_k^*)\|$ will eventually go to zero faster than η_k, at which point it is no longer necessary to increase ρ_k. Thus, we can be assured that the algorithm does not increase ρ_k without bound.

Bertsekas suggests constructing the sequence η_k as

$$\eta_{k+1} = \gamma\|c(x_k^*)\|, \tag{3.45}$$

for some $\gamma < 1$. Within Algorithm 3, this would lead to the following rule for updating ρ_k:

$$\rho_{k+1} = \begin{cases} \rho_k & \text{if } \|c(x_k^*)\| \le \gamma\|c(x_k)\| \\ \tau_\rho\rho_k & \text{if } \|c(x_k^*)\| > \gamma\|c(x_k)\|. \end{cases} \tag{3.46}$$

As ρ_k gets larger, the convergence rate gets arbitrarily close to superlinear, so that the first case of (3.46) is always satisfied, and ρ_k becomes constant for all k large enough.

We prefer not to use rule (3.45) because it may be too strict. Any intermediate (and non-optimal) iterate x_k^* could be feasible or nearly feasible for (GNP), so that $\|c(x_k^*)\|$ could be very small. Then η_{k+1} would be smaller than warranted on the following iteration. The forcing sequence suggested by Conn et al. [1991b, 1996] does not suffer from this defect, and has been proven by them to keep ρ_k bounded. We have used this update in Algorithm 3 (see Steps 3.8 and 3.11).

In this section we discuss some of the local convergence characteristics of the stabilized LCL method. For this analysis and the remainder of this chapter, we assume that ρ_k is uniformly bounded, so that $\rho_k = \bar{\rho}$ for all k greater than some $\bar{k}$. Hence, we drop the subscript on ρ_k and simply write $\bar{\rho}$. We only consider iterations $k > \bar{k}$.

We begin by discussing the local convergence rates of the algorithm under the assumption that the elastic variables are always zero—that is, the linearized constraints are always satisfied. Next, we show that after finitely many iterations the elastic penalty parameter σ_k will always be large enough to guarantee that this assumption holds. In this way, we demonstrate that stabilized LCL becomes equivalent to MINOS as it approaches the solution.

3.4.1 Convergence rates

Under the assumption that the elastic variables are always equal to 0 and that $\bar{\rho}$ is finite, the steps executed by Algorithms 2 and 3 are identical and the subproblems (ELC$_k$) and (LC$_k$) are also identical. The only difference is the multiplier update formulas:

$$\text{LCL update} \qquad y_{k+1} = y_k^* \tag{3.47}$$

$$\text{Stabilized LCL update} \qquad y_{k+1} = y_k^* - \bar{\rho}c(x_k^*), \tag{3.48}$$

which differ only by the vector $\bar{\rho}c(x_k^*)$. We may think of this vector as a perturbation of the LCL multiplier update. Moreover, by Theorem 2.8, this perturbation converges to 0 at the same rate as $\{x_k^*\}$ converges to x_*. Therefore, it does not interfere with the convergence rate of the stabilized LCL iterates. Theorem 2.8 then applies to the stabilized LCL method.

3.4.2 Asymptotic equivalence to MINOS

Much of the efficiency of LCL methods, including MINOS, derives from the fact that they eventually identify the correct active set, and each subproblem restricts its search to the subspace defined by a linear approximation of the constraints. This approximation can be very accurate near the solution. The stabilized LCL subproblems do *not* restrict themselves to this subspace. In early iterations we do not expect, nor do we wish, the method to honor these linearizations (see the discussion in §2.2.2 on the perils of doing so). The elastic variables give the subproblems an opportunity to deviate from this subspace. In order to recover LCL's fast convergence rate, however, it is not desirable to allow deviation near the solution.

We show below that as the stabilized LCL iterations approach a solution of (GNP), the solutions of the stabilized LCL subproblems eventually always have the elastic variables equal to zero. Hence, $\bar{c}_k(x_k^*) = 0$ and $v_k^*, w_k^* = 0$, so that each x_k^* satisfies the constraints of the MINOS subproblem, and the objective of (ELC_k) at (x_k^*, v_k^*, w_k^*) is equivalent to the MINOS subproblem objective (see (1.10)).

As discussed in §3.4.1, Theorem 2.8 applies to Algorithm 3, and so for all k large enough, (2.16) yields

$$\|y_k^* - y_k\| \le M\|F(x_k, y_k, z_k)\| \tag{3.49}$$

for some positive constant M. By Corollary 3.8, $(x_k, y_k, z_k) \to (x_*, y_*, z_*)$, and because $\|F(x_*, y_*, z_*)\| = 0$ and F is continuous,

$$\|F(x_k, y_k, z_k)\| < \frac{\underline{\sigma}}{M} \tag{3.50}$$

for all k large enough ($\underline{\sigma}$ is defined in Algorithm 3). Combining (3.49) and (3.50), we conclude that

$$\|y_k^* - y_k\| < \underline{\sigma} \tag{3.51}$$

for all k large enough. However, Step 3.6 of Algorithm 3 guarantees that $\underline{\sigma} \le \sigma_k$ for all k, and so from (3.51), $\|y_k^* - y_k\| < \sigma_k$ for all k large enough. Lemma 3.2 then implies that σ_k will be sufficiently large that the optimal elastic variables will be equal to 0.

3.5 Infeasible Problems

Not all optimization problems are feasible. The user of an optimization algorithm may formulate a set of nonlinear constraints $c(x) = 0$ for which no nonnegative solution exists. Detecting infeasibility of the system $c(x) = 0$, $x \ge 0$ is equivalent to verifying that the *global* minimizer of

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|c(x)\|^2 \\ \text{subject to} & x \ge 0 \end{array} \tag{3.52}$$

yields a positive objective value. Detecting infeasibility of the nonlinear constraints is a useful feature, but it is a very difficult problem and is beyond the purview of this thesis.

We analyze the properties of the stabilized LCL algorithm when it is applied to an infeasible problem with convergence tolerances $\omega_* = \eta_* = 0$. We show that Algorithm 3 converges to a point that satisfies the first-order optimality conditions of the minimum-norm problem (3.52).

Theorem 3.10. *Let x_* be any limit point of the sequence of vectors $\{x_k^*\}$ generated by Algorithm* 3*, and let $\mathcal{K}$ be the infinite set of indices associated with that subsequence. Suppose that* (GNP) *is infeasible. Then, under the assumptions of Lemma* 3.5*,*

$$\lim_{k\in\mathcal{K}} J(x_k^*)^T c(x_k^*) = z_* \stackrel{\text{def}}{=} J_*^T c_*,$$

and (x_*, z_*) is a first-order KKT point for (3.52)*.*

Proof. The pair (x_*, z_*) satisfies the first-order KKT conditions of (3.52) if

$$x_*, z_* \geq 0 \tag{3.53a}$$

$$\min(x_*, z_*) = 0 \tag{3.53b}$$

$$J_*^T c_* = z_*. \tag{3.53c}$$

Because (GNP) is infeasible, there exists a constant $\delta > 0$ such that $\delta < \|c(x)\|$ for all $x \geq 0$. Moreover, Steps 3.8 and 3.11 of Algorithm 3 generate a sequence $\{\eta_k\}$ converging to 0, and so $\eta_k < \delta$ for all k large enough. Consider only such k. Then, $\eta_k < \delta < \|c(x_k^*)\|$ and Step 3.10 is executed at every k, so that $\rho_k \to \infty$ and $\sigma_k \to 0$. Moreover, x_k and y_k are not updated, so that for some n-vector $\bar{x}$ and m-vector $\bar{y}$,

$$x_k \equiv \bar{x} \quad \text{and} \quad y_k \equiv \bar{y}. \tag{3.54}$$

First, note that Algorithm 3 generates x_k^* satisfying (3.17). Therefore, $x_k^* \geq 0$ for all k, and so $\lim_{k\in\mathcal{K}} x_k^* = x_*$ implies $x_* \geq 0$. Then x_* satisfies (3.53a).

From (3.17b), (3.23), and (3.54),

$$g(x_k^*) - J(x_k^*)^T(\bar{y} - \rho_k c(x_k^*)) + J(\bar{x})^T(\bar{y} - y_k^*) \geq -\omega_k e, \tag{3.55}$$

or, after rearranging terms,

$$\underbrace{g(x_k^*) - J(x_k^*)^T \bar{y}}_{(a)} + \underbrace{J(\bar{x})^T(\bar{y} - y_k^*)}_{(b)} + \rho_k J(x_k^*)^T c(x_k^*) \geq -\omega_k e. \tag{3.56}$$

By hypothesis, all iterates x_k^* lie in a compact set, and so (a) is bounded because g and J are continuous and $\bar{y}$ is constant. Also, (b) is bounded because $\bar{x}$ and $\bar{y}$ are constant, and from (3.17f) we have $\|\bar{y} - y_k^*\|_\infty \leq \sigma_k + \omega_k$. Then, because $\omega_k \to 0$ and $\rho_k \to \infty$, (3.56) implies that $J(x_k^*)^T c(x_k^*) \geq 0$ for all k large enough. Otherwise, (3.56) would eventually be violated as ρ_k grew large. Then, $z_* \stackrel{\text{def}}{=} \lim_{k \in \mathcal{K}} J(x_k^*)^T c(x_k^*) = J_*^T c_* \geq 0$, and so (3.53a) and (3.53c) are satisfied.

It only remains to show that (x_*, z_*) satisfies (3.53b). Because all x_k^* lie in a compact set, there exists some constant $L > 0$ such that

$$\|x_k^* - \bar{x}\| \leq \frac{L\alpha_1}{M}, \tag{3.57}$$

where M and α_1 are as defined in Lemma 3.5. Substituting (3.57) into (3.28) and using (3.17f),

$$\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T y_k^* + \rho_k \widehat{J}(x_k^*)^T c(x_k^*)\| \leq \omega_k + L(\sigma_k + \omega_k). \tag{3.58}$$

Dividing (3.58) through by ρ_k,

$$\left\| \frac{1}{\rho_k}\left(\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T y_k^*\right) + \widehat{J}(x_k^*)^T c(x_k^*) \right\| \leq \frac{\omega_k + L(\sigma_k + \omega_k)}{\rho_k}. \tag{3.59}$$

The quantity $\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T y_k^*$ is bounded for the same reasons that (a) and (b) above are bounded. Taking limits of both sides of (3.59), $\rho_k \to \infty$ and $\omega_k, \sigma_k \to 0$ imply that $\widehat{J}(x_k^*)^T c(x_k^*) \to 0$. By continuity of J and c, $\widehat{J}_*^T c_* = 0$. Equivalently, we may write

$$[J_*^T c_*]_j = 0 \quad \text{if} \quad [x_*]_j > 0,$$

for $j = 1, \dots, n$, and so condition (3.53b) holds, as required. ∎

Theorem 3.10 describes a useful feature of Algorithm 3. When applied to an infeasible problem, the algorithm converges to a solution of (3.52)—or at least to a first-order point. One important caveat deserves mention: if the convergence tolerance η_* is small (it usually will be), Algorithm 3 may never terminate. We need to insert an additional test to provide for the possibility that (GNP) is infeasible. For example, the test could force the algorithm to exit if ρ_k is above a certain threshold value and $\|c(x_k^*)\|$ is no longer decreasing. However, any test we devise is necessarily heuristic: it is impossible to know for certain if a larger value of ρ_k would force $\|c(x_k^*)\|$ to be less than η_*. We discuss this further in §4.6.

3.6 Second-Order Optimality

The stabilized LCL method imposes few requirements on the manner in which the LC subproblems are solved. Our implementation (see Chapter 4) uses MINOS or SNOPT to solve the LC subproblems. These are active-set solvers suitable for optimization problems with few expected degrees of freedom at the solution, and in which only first derivatives are available. However, second derivatives might be readily available for some problems. Also, some problems are expected to have many degrees of freedom at the solution. In either case, an interior-point solver (requiring second derivatives) may be more appropriate for the solution of the subproblems.

Lemma 3.5 and Theorem 3.7 assert that iterates generated by the stabilized LCL algorithm converge to first-order KKT points. A subproblem solver that uses second-derivatives may be able to guarantee convergence to second-order points. If we augment the convergence criteria for the solution of each subproblem to include second-order conditions, we can show that Algorithm 3 generates iterates converging to points satisfying the second-order sufficiency conditions for (GNP). The following assumption strengthens the first-order conditions (3.16). Recall that $\mathcal{M}_k(x,y,\rho)$ depends on x_k.

Assumption 3.11. *Let x_* be any limit point of the sequence $\{x_k^*\}$, and let $\mathcal{K}$ be the infinite set of indices associated with that convergent subsequence. For all $k \in \mathcal{K}$ large enough, the following conditions hold at each x_k^* and z_k^*: For some $\delta > 0$, independent of k,*

1. *(Strict Complementarity) Either*

$$[x_k^*]_j > \delta \quad \text{or} \quad [z_k^*]_j > \delta, \tag{3.60}$$

   but not both, for each $j = 1, \ldots, n$;

2. *(Second-Order Condition) For any $\rho \geq 0$,*

$$p^T \nabla^2_{xx} \mathcal{M}_k(x_k^*, y_k^*, \rho) p > \delta \tag{3.61}$$

for all $p \neq 0$ *satisfying* $J(x_k^*)p = 0$ *and* $[p]_j = 0$ *for all* j *such that* $[x_k^*]_j = 0$.

Condition (3.61) implies that the reduced Hessian of $\mathcal{M}_k$ is uniformly positive definite at all x_k^*.

The following result extends Corollary 3.8 to consider the case in which iterates generated by Algorithm 3 satisfy Assumption 3.11. Conn et al. [1991b] show a similar result for their BCL method.

Theorem 3.12. *Suppose Assumptions* 1.1, 2.2, 2.3, *and* 3.11 *hold. Let* $\{(x_k^*, y_k^*, z_k^*)\}$ *and* $\{(x_k, y_k, z_k)\}$ *be the sequences of vectors generated by Algorithm* 3*. Let* x_* *be any limit point of the sequence* $\{x_k^*\}$*, and let* $\mathcal{K}$ *be the infinite set of indices associated with that convergent subsequence. Then*

$$\lim_{k \in \mathcal{K}} (x_k, y_k, z_k) = (x_*, y_*, z_*) \tag{3.62}$$

and (x_*, y_*, z_*) *is an isolated local minimizer of* (GNP)*.*

Proof. It follows immediately from Corollary 3.8 that (3.62) holds and that (x_*, y_*, z_*) is a first-order KKT point for (GNP). It only remains to show that (x_*, y_*, z_*) satisfies the second-order sufficiency conditions, laid out in Definition 1.4.

By hypothesis, x_k^* and z_k^* satisfy Part 1 of Assumption 3.11 for all $k \in \mathcal{K}$. Therefore, their limit points satisfy

$$[x_*]_j \geq \delta \quad \text{or} \quad [z_*]_j \geq \delta,$$

but not both, for each $j = 1 \ldots n$, and so x_* and z_* satisfy strict complementarity (Definition 1.3). We now show that x_* and y_* satisfy the second-order sufficiency conditions for (GNP). Because x_k^* and x_k converge to x_* as $k \in \mathcal{K}$ goes to infinity and because all x_k^* and y_k^* satisfy Part 2 of Assumption 3.11, the continuity of $\mathcal{M}_k$ and J implies that for any $\rho \geq 0$,

$$\lim_{k \in \mathcal{K}} p^T \nabla_{xx}^2 \mathcal{M}_k(x_k^*, y_k^*, \rho) p \geq \delta \tag{3.63}$$

for all $p \neq 0$ satisfying $J_* p = 0$ and $[p]_j = 0$ for all j such that $[x_*]_j = 0$. From Lemma 3.5, $\lim_{k \in \mathcal{K}} \widehat{y}(x_k^*, y_k^*, \rho) = y_*$. Using the definition of $\mathcal{M}_k$ (see (1.9)),

$$\begin{aligned}
\lim_{k \in \mathcal{K}} \nabla_{xx}^2 \mathcal{M}_k(x_k^*, y_k^*, \rho) &= \lim_{k \in \mathcal{K}} H(x_k^*) - \sum_{i=1}^{m} [\widehat{y}(x_k^*, y_k^*, \rho)]_i H_i(x_k^*) + \rho J(x_k^*)^T J(x_k^*) \\
&= H(x_*) - \sum_{i=1}^{m} [y_*]_i H_i(x_*) + \rho J(x_*)^T J(x_*) \\
&= \nabla_{xx}^2 \mathcal{L}(x_*, y_*, \rho).
\end{aligned} \tag{3.64}$$

Because $\delta > 0$, using (3.64) we can rewrite (3.63) as

$$p^T \nabla^2_{xx} \mathcal{L}(x_*, y_*, \rho) p > 0.$$

Therefore, (x_*, y_*, z_*) satisfies the second-order sufficiency conditions for (GNP), as required. ■

Chapter 4

Implementation

The practical implementation of an algorithm invariably requires many features that are not made explicit by its theory. In this chapter we discuss some important details of our implementation of the stabilized LCL method. The algorithm has been implemented in MATLAB [MathWorks, 1992] and is called LCLOPT. It makes use of the Fortran codes MINOS [Murtagh and Saunders, 1978, 1982] and SNOPT [Gill et al., 1997] to solve the linearly constrained subproblems. We now turn our attention back to the more general problem (NP), first presented in §1.1, and leave (GNP) behind.

4.1 Problem Formulation

LCLOPT does not solve (NP) directly, but rather solves the equivalent problem

$$\text{(NPi)} \qquad \begin{array}{ll} \underset{x,s}{\text{minimize}} & f(x) \\ \text{subject to} & \begin{pmatrix} c(x) \\ Ax \end{pmatrix} - s = 0, \quad l \le \begin{pmatrix} x \\ s \end{pmatrix} \le u. \end{array}$$

The vector of slack variables s is subdivided into nonlinear and linear slack variables, s_N and s_L. The general constraints of (NPi) are then given by the equations

$$\begin{aligned} c(x) - s_N \quad\quad &= 0 \\ Ax \quad\quad - s_L &= 0. \end{aligned}$$

The formulation of problem (NPi) is chosen to match the problem formulation used by SNOPT. It is also closely related to that used by MINOS. As in those methods, our implementation distinguishes between variables in the vector x that appear and do not appear nonlinearly in the objective or the constraints; variables that appear only linearly are treated specially. The following discussion ignores this detail in order to keep the notation concise.

The linearly constrained subproblems corresponding to (NPi) take the form

$$
\text{(ELCi}_k\text{)} \qquad \begin{array}{ll} \underset{x,s,v,w}{\text{minimize}} & \mathcal{M}_k(x) + y_k^T(v-w) + \sigma_k e^T(v+w) \\ \text{subject to} & \begin{pmatrix} c_k + J_k(x-x_k) + v - w \\ Ax \end{pmatrix} - s = 0, \quad l \le \begin{pmatrix} x \\ s \end{pmatrix} \le u, \\ & \qquad\qquad\qquad\qquad\qquad\qquad 0 \le v, w. \end{array}
$$

4.2 The Main Algorithm

LCLOPT is coded in MATLAB, version 6. The structure of the implementation is illustrated by Figure 4.1, in which `LCLOPT.m` represents the "outer" level of the LCL algorithm. The main tasks of the outer level are to form the subproblems, update solution estimates, update parameters, and test for convergence or errors. The computational kernel of LCLOPT resides in the solution of each LC subproblem, and the efficiency of the implementation ultimately relies on the efficiency of the subproblem solver.

4.3 Solving the LC Subproblems

LCLOPT can use either MINOS or SNOPT to solve (ELCi$_k$). For linearly constrained problems, MINOS uses a reduced-gradient method, coupled with a quasi-Newton approximation of the reduced Hessian of the the problem objective. SNOPT implements a sparse SQP method and maintains a limited-memory, quasi-Newton approximation of the Hessian of the problem objective. (In both cases, the problem objective will be the objective of (ELCi$_k$).) For linearly constrained problems, SNOPT avoids performing an expensive Cholesky factorization of the reduced Hessian for the quadratic programming subproblem in each of its own major iterations, and thus realizes considerable computational savings over problems with nonlinear constraints [Gill et al., 2002].

Both MINOS and SNOPT are available as libraries of Fortran 77 routines. We implemented MEX interfaces [MathWorks, 1995] written in C to make each of the routines from the MINOS and SNOPT libraries accessible from within MATLAB. The subproblem solvers evaluate the nonlinear objective function (there are no nonlinear constraints in (ELCi$_k$)) through a generic MEX interface, `funObj.c`. This routine makes calls to a MATLAB routine to evaluate the nonlinear objective $\mathcal{M}_k$. In turn, the routine

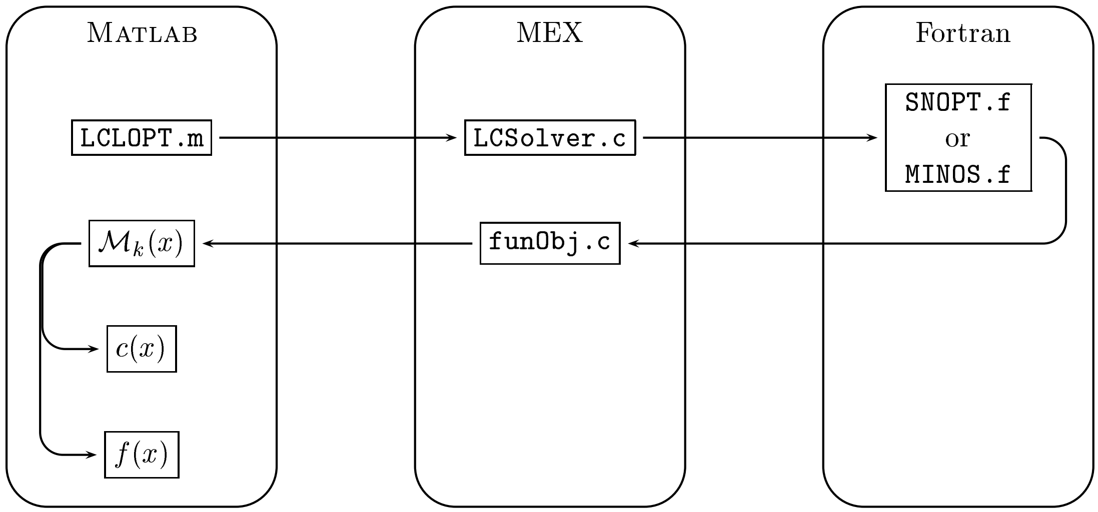

FIGURE 4.1 *The sequence of subroutine calls for the solution of each stabilized LCL subproblem. The constraints c are always evaluated before the objective f.*

for $\mathcal{M}_k$ makes calls to routines (available as MATLAB or MEX routines) to evaluate the original nonlinear functions f and c. Figure 4.1 summarizes the calling sequence.

4.4 Computing an Initial Point

MINOS and SNOPT both ensure that all iterates remain feasible (to within a small tolerance) with respect to the bounds and linear constraints in (ELCi$_k$), which includes the bounds and linear constraints in (NPi). LCLOPT is therefore able to restrict the evaluation of the nonlinear functions f and c to points in the latter region. A user of LCLOPT may thus introduce bounds and linear constraints into (NPi) to help guard against evaluation of the nonlinear functions at points where they are not defined.

Before entering the first iteration of the stabilized LCL method, LCLOPT solves the following quadratic *proximal-point* (PP) problem:

$$\text{(PP2)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|x - \widetilde{x}\|_2^2 \\ \text{subject to} & l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u, \end{array}$$

where $\widetilde{x}$ is a vector provided by the LCLOPT user. The solution of (PP2) is used as the initial point x_0 for the algorithm. The objective function of the PP problem helps find an x_0 reasonably close to $\widetilde{x}$, while the constraints ensure that x_0 is feasible with respect to the bounds and linear constraints of (NPi). If (PP2) proves infeasible, (NPi) is declared infeasible and LCLOPT exits immediately with an error message.

An alternative PP problem is based on the one-norm deviation from $\widetilde{x}$:

$$\text{(PP1)} \qquad \begin{array}{ll} \underset{x}{\text{minimize}} & \|x - \widetilde{x}\|_1 \\ \text{subject to} & l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u. \end{array}$$

An advantage of (PP1) is that it can be reformulated and solved as a linear program, and its solution is therefore expected to lie on more constraint vertices. It is a correspondingly easier problem to solve. SNOPT provides the option of solving either (PP1) or (PP2). LCLOPT can take advantage of this by initializing $x = \widetilde{x}$ and passing SNOPT the optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & 0 \\ \text{subject to} & l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u, \end{array}$$

with the parameter `Proximal Point` set to either 1 or 2. (For MINOS, the constraints would have to be reformulated and a set of elastic variables introduced.)

The computational results presented in Chapter 5 were derived using (PP2) to compute x_0. As suggested by Gill et al. [2002], a loose optimality tolerance on (PP2) is used to limit the computational expense of its solution: reducing the number of iterations and (typically) the number of superbasic variables.

4.5 Early Termination of the LC Subproblems

The global convergence results for the stabilized LCL algorithm (cf. Lemma 3.5 and Theorem 3.7) assume that the optimality tolerance for each of the subproblem solutions, ω_k, converges to 0. This loose requirement allows much flexibility in constructing the sequence $\{\omega_k\}$.

The solution estimates may be quite poor during early iterations. We expect slow progress during those iterations, even if they are solved to tight optimality tolerances. A loose tolerance may help limit the computational work performed by the subproblem solver during these early iterations. Near a solution, however, we wish to reduce the optimality tolerance quickly in order to take advantage of the fast local convergence rate predicted by Theorem 2.8.

To construct the sequence $\{\omega_k\}$, we replace Step 3.1 of Algorithm 3 on p. 35 with

$$\begin{aligned} \tilde{\omega}_k &\leftarrow \min(\omega_k, \|F(x_k, y_k, z_k)\|_\infty^2) \\ \omega_{k+1} &\leftarrow \max(0.5\tilde{\omega}_k, \omega_*), \end{aligned} \tag{4.1}$$

where ω_0 can be set by a user to any value between 0.5 and ω_*. The update (4.1) guarantees that $\omega_k \to \omega_*$, as required.

Following the prescription outlined in §3.1.2, we fix at a small value the feasibility tolerance for satisfying the linearized constraints. The feasibility and optimality tolerances for each major iteration are passed to the subproblem solver as run-time parameters.

4.6 Detecting Infeasibility and Unboundedness

As discussed in §3.5, Algorithm 3 will *not* exit if the optimization problem is infeasible and the infeasibility tolerance η_* is small. We declare (NPi) infeasible if at any given iteration k, x_k is infeasible with respect to the nonlinear constraints and the penalty parameter is greater than some threshold value. In particular, at Step 3.10, Algorithm 3 exits and (NPi) is declared infeasible if

$$\begin{aligned} \max(\|[l_c - c_k]^+\|_\infty, \|[u_c - c_k]^+\|_\infty) &> \eta_* \\ \rho_k &> \bar{\rho}, \end{aligned}$$

where l_c and u_c are the lower and upper bounds for the nonlinear constraints. For the computational results in Chapter 5 the threshold value was set at $\bar{\rho} = 10^8$.

We also need to consider the possibility that (NPi) is unbounded—i.e., that the objective function f is unbounded below in the feasible region, or that $\|x\| \to \infty$. As with tests for infeasibility, any test for unboundedness must be ad hoc. We rely on the LC solver to help detect infeasibility. Problem (NPi) is declared unbounded and LCLOPT exits if the point x_k is feasible and the LC solver reports (ELCi$_k$) as unbounded.

4.7 Summary of the Stabilized LCL Method

Following is a summary of the stabilized LCL method as implemented in LCLOPT. We assume that $\widetilde{x}$ is given and that the starting tolerances, ω_0 and η_0, and parameters, ρ_0 and σ_0, are set.

1. Apply the LC solver to (PP1) or (PP2) to obtain a starting point x_0 that is feasible with respect to the bounds and linear constraints and reasonably close to $\widetilde{x}$. If the PP problem is infeasible, declare (NPi) infeasible and exit. Otherwise, set $k = 0$.

2. Evaluate the functions and gradients at x_k. Linearize the constraints and form (ELCi$_k$).

3. Apply the LC solver to (ELCi$_k$) and compute (x_k^*, y_k^*, z_k^*) with optimality tolerance ω_k.

4. If (ELCi$_k$) is unbounded and x_k^* is *feasible*, declare (NPi) unbounded and exit. If (ELCi$_k$) is unbounded and x_k^* is *infeasible*, go to Step 8. Otherwise, continue.

5. If x_k^* meets the current nonlinear feasibility threshold η_k, continue. Otherwise, go to Step 8.

6. Update the solution estimate: $(x_{k+1}, y_{k+1}, z_{k+1}) \leftarrow (x_k^*, y_k^*, z_k^*)$. Keep the penalty parameter ρ_k fixed and reset the infeasibility weight σ_k.

7. Test convergence: If $(x_{k+1}, y_{k+1}, z_{k+1})$ satisfies the optimality conditions for (NPi), declare the current solution estimate optimal, return $(x_{k+1}, y_{k+1}, z_{k+1})$, and exit. Otherwise, go to Step 9.

8. If $\rho_k > \bar{\rho}$, declare (NPi) infeasible, return (x_k^*, y_k^*, z_k^*), and exit. Otherwise, discard the subproblem solution (i.e., $(x_{k+1}, y_{k+1}, z_{k+1}) \leftarrow (x_k, y_k, z_k)$), increase the penalty parameter ρ_k, and reduce the elastic weight σ_k.

9. Set the nonlinear feasibility threshold, η_{k+1}. Reduce the LC subproblem optimality tolerance, ω_{k+1}.

10. Set $k \leftarrow k + 1$. Return to Step 2.

Chapter 5

Numerical Results

This chapter summarizes the results of applying LCLOPT, our implementation of the stabilized LCL method, to a subset of nonlinearly constrained test problems from the COPS 2.0 [Dolan and Moré, 2000], Hock-Schittkowski [Hock and Schittkowski, 1981], and CUTE [Bongartz et al., 1995] test suites. Two versions of LCLOPT are applied to each test problem: The first version uses MINOS to solve the sequence of linearly constrained subproblems; the second version uses SNOPT.

We made use of the AMPL versions of all problems, as formulated by Vanderbei [2002]. A MEX interface to the AMPL libraries makes functions and gradients available in MATLAB (see Gay [1997] for details on interfacing external routines to AMPL). All runs were conducted on an AMD Athlon 1700XP using 384 MB of RAM, running Linux 2.4.18.

For each test problem, the performances of the LCLOPT routines are compared with the performance of AMPL/MINOS 5.5 (version 19981015) [Fourer et al., 1993]. Tables 5.2, 5.4, and 5.7 summarize the performance of the two versions of LCLOPT (shown by the first two groups of columns) and MINOS (shown by the last group of columns). Table 5.1 defines the column heads used in the summary tables, as well as the symbols used to identify special exit conditions. These exit flags appear in the last column of each group.

5.1 Default Parameters

Figure 5.1 shows the options files that LCLOPT uses for the LC solvers. These are fixed for all subproblems. Separately, at each major iteration, LCLOPT sets the parameter `Optimality Tolerance` in MINOS and the parameter `Major Optimality Tolerance` in SNOPT. These are equivalent to the subproblem optimality tolerance ω_k (§4.5 outlines the method for choosing this parameter).

Each test problem supplies a default starting point. This point is used as $\widetilde{x}$ in the

TABLE 5.1
Column heads and symbols used in the tables of numerical results

Head/ Symbol	Meaning
Mnr	No. of minor iterations (i.e., total subproblem iterations)
Mjr	No. of major iterations (i.e., total subproblems solved)
Fcn	Max. of the no. of nonlinear objective and constraint evaluations
$f(x_*)$	Final objective value
Viol	Final constraint violation norm
i	Nonlinear constraints locally infeasible
f	Final point could not be improved
u	Unbounded or badly scaled problem
t	Terminated by iteration limit
s	Terminated by superbasics limit

proximal-point problem (see §4.4). The initial vector of multiplier estimates y_0 is set to zero.

Both MINOS and SNOPT provide the option to reuse a quasi-Newton approximation of a Hessian from a previous solve: MINOS approximates the reduced Hessian; SNOPT approximates the full Hessian. We take advantage of this feature for all iterations $k = 2, 3, 4, \ldots$ by setting the MINOS and SNOPT option `Start = 'Hot'`.

The parameters used by Algorithm 3 (p. 35) are set as follows. The upper and lower bounds of the elastic penalty parameters are $\underline{\sigma} = 1$ and $\overline{\sigma} = 10^4$. The initial infeasibility weight is $\sigma_0 = 10^2$. (Normally, LCLOPT scales this quantity by $1 + \|y_0\|_\infty$, but the scaling has no effect for these test runs because $y_0 \equiv 0$.) The penalty scaling factors are $\tau_\rho = 100^{0.5}$ and $\tau_\sigma = 10$. As suggested by Conn et al. [1991b], we set $\alpha = 0.1$ and $\beta = 0.9$. The initial penalty parameter is $\rho_0 = 10^{5/2}/m_c$, where m_c is the number of nonlinear constraints. The final optimality and feasibility tolerances are $\omega_* = \eta_* = 10^{-6}$. The initial optimality and feasibility tolerances are $\omega_0 = 10^{-3}$ $(= \sqrt{\omega_*})$ and $\eta_0 = 1$.

In all cases, default options, with the exception of `Major Iterations 500` and `Superbasics Limit 2000`, are used for the MINOS benchmarks.

5.2 The COPS Test Problems

There are 17 problems in the COPS 2.0 collection [Dolan and Moré, 2000]. Five problems are excluded for the following reasons:

- 3 problems are unconstrained: *bearing*, *minsurf*, and *torsion*;

```
BEGIN LCL SUBPROBLEM
   Scale option                 0
   Superbasics limit         2000
   Iterations                5000
   Feasibility tol         1.0e-6
END LCL SUBPROBLEM
```

The MINOS specs file

```
BEGIN LCL SUBPROBLEM
   Scale option                   0
   Superbasics limit           2000
   Iterations                  5000
   Major iterations            1000
   Minor iterations             500
   Minor feasibility tol     1.0e-6
   Minor optimality  tol     2.5e-7
END LCL SUBPROBLEM
```

The SNOPT specs file

FIGURE 5.1 *The fixed optional parameters for every subproblem solve. The optimality tolerance* ω_k *is specified by LCLOPT for each* k*.*

- 2 problems cause system errors when called using the AMPL MEX interface: *glider* and *marine*.

The dimensions of the COPS test problems can be adjusted. In all cases, the solvers were applied to the largest version of the problem (as specified by the AMPL model) that would not cause the system to page memory to disk. Table A.1 on p. 73 summarizes the dimensions of the selected problems, and Table 5.2 on the next page shows the numerical results.

The version of LCLOPT using MINOS for the subproblems solved all 12 problems to first-order optimality. The version using SNOPT solved 11 problems to first-order optimality; the exception was *robot*, which it declared as having infeasible nonlinear constraints. MINOS solved 10 of the 12 problems to optimality; it declared *steering* an infeasible problem, and it terminated the solution of *elec* because of excessive iterations. Feasible points exist for all of the test problems chosen, so we consider all declarations of infeasibility to be errors. Table 5.3 on p. 61 summarizes these results. Note that different local optima appear to have been found for problems *camshape, methanol, polygon*, and *rocket*.

TABLE 5.2
Numerical results: The 12 selected COPS test problems

	LCLOPT (MINOS)					LCLOPT (SNOPT)					MINOS				
Problem	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol
camshape	1338	8	1401	4.071686e+00	3.98e-16	1147	5	16	4.071686e+00	2.91e-14	1518	17	596	4.274258e+00	4.10e-13
catmix	1563	8	2717	-4.805427e-02	5.14e-17	59262	88	2715	-4.805550e-02	1.14e-15	181	8	754	-4.744486e-02	2.20e-16
chain	3757	10	4297	5.068917e+00	1.24e-10	3046	9	1342	5.068918e+00	3.85e-11	1286	35	2954	5.068918e+00	1.90e-07
channel	1305	4	640	1.000000e+00	5.68e-13	633	4	26	1.000000e+00	5.62e-13	301	5	6	1.000000e+00	2.30e-13
elec	5900	16	7568	1.843899e+04	6.61e-07	21476	16	5554	1.843905e+04	9.17e-08	39427	500	41610	1.546797e+04	^{t}2.00e+00
gasoil	4536	10	2612	5.236596e-03	6.32e-08	5348	10	92	5.236598e-03	1.84e-07	1373	45	1189	5.236596e-03	1.90e-08
methanol	8161	20	5338	1.727307e-02	4.12e-14	6838	14	622	9.022290e-03	3.99e-11	1496	11	245	9.022290e-03	3.00e-13
pinene	7886	6	3706	1.987217e+01	1.10e-14	12554	5	111	1.987217e+01	1.31e-14	3030	11	192	1.987217e+01	4.10e-13
polygon	1832	5	1916	-3.414973e-14	5.36e-17	830	4	55	-4.574389e-16	0.00e+00	7742	171	9661	6.884045e-01	2.10e-13
robot	8575	11	11983	9.121409e+00	5.53e-13	25538	2	293	3.922458e+04	^{i}3.80e+02	1369	37	3722	9.141090e+00	5.30e-13
rocket	3576	12	6110	1.000006e+00	6.22e-14	7131	14	115	9.853280e-01	6.70e-08	1461	12	1367	1.011302e+00	2.90e-13
steering	5521	8	4793	5.545724e-01	1.30e-09	3715	8	73	5.545739e-01	1.52e-07	2204	20	1405	2.177778e-12	^{i}5.00e+00

TABLE 5.3
Summary: The 12 selected COPS test problems

	LCLOPT (MINOS)	LCLOPT (SNOPT)	MINOS
Optimal	12	11	10
False Infeasibility		1	1
Terminated			1

Note the excessive number of minor iterations required by LCLOPT on *catmix*, *elec*, and *robot* with SNOPT as its subproblem solver. Especially during early major iterations, SNOPT was unable to solve the LC subproblems to the required optimality tolerance within the 5000 iteration limit. Rather than terminate with an error message, LCLOPT forces SNOPT to keep working on the same subproblem until it returns a solution within the required optimality tolerance. In practice, a different strategy would be adopted, but our goal here is to test the robustness of the outer iterations (the stabilized LCL method), not the robustness of the subproblem solvers.

As compared with MINOS, LCLOPT tends to require more minor iterations (a measure of total computational work), but fewer major iterations, to reach a solution. We comment further on this fact in §6.1.

5.3 The Hock-Schittkowski Test Problems

The HS test suite contains 86 nonlinearly constrained problems [Hock and Schittkowski, 1981]. These are generally small and dense problems. We exclude 5 problems from this set for the following reasons:

- 3 problems are not smooth: *hs67*, *hs85*, and *hs87*;
- 2 problems require external functions: *hs68* and *hs69*.

Table A.2 on p. 74 summarizes the dimensions of the selected problems, and Table 5.4 on the following pages presents the numerical results.

Both versions of LCLOPT solved the same 80 problems to first-order optimality, but both declared *hs109* infeasible. MINOS solved 80 problems to first-order optimality, but declared *hs93* infeasible. Table 5.5 on p. 65 summarizes these results.

Table 5.4
Numerical results: The 81 selected Hock-Schittkowski test problems

	LCLOPT (MINOS)					LCLOPT (SNOPT)					MINOS				
Problem	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol
hs6	70	14	146	1.084384e-18	5.60e-16	100	13	115	1.390971e-23	1.82e-08	27	18	87	4.083108e-20	3.60e-12
hs7	119	12	212	-1.732051e+00	0.00e+00	98	11	102	-1.732051e+00	3.12e-11	22	15	62	-1.732051e+00	3.20e-12
hs8	19	4	43	-1.000000e+00	6.55e-12	22	4	24	-1.000000e+00	1.59e-15	1	7	7	-1.000000e+00	3.60e-15
hs10	109	13	195	-1.000000e+00	3.08e-08	170	12	153	-1.000000e+00	7.85e-09	22	15	58	-1.000000e+00	1.20e-13
hs11	25	7	54	-8.498464e+00	3.10e-12	62	7	61	-8.498464e+00	4.48e-11	18	11	45	-8.498464e+00	2.40e-13
hs12	32	6	68	-3.000000e+01	1.48e-07	44	7	59	-3.000000e+01	9.58e-11	61	23	154	-3.000000e+01	4.90e-10
hs13	11	17	88	9.949892e-01	2.55e-08	20	31	78	1.006705e+00	6.85e-08	12	23	50	1.000665e+00	0.00e+00
hs14	3	5	24	1.393465e+00	9.29e-16	5	5	14	1.393465e+00	2.32e-15	1	6	6	1.393465e+00	3.10e-11
hs15	2	3	14	3.065000e+02	0.00e+00	13	3	14	3.065000e+02	0.00e+00	37	17	79	3.603798e+02	0.00e+00
hs16	16	5	38	2.500000e-01	0.00e+00	37	5	33	2.500000e-01	0.00e+00	1	5	6	2.314466e+01	0.00e+00
hs17	10	5	36	1.000000e+00	4.56e-15	12	5	21	1.000000e+00	0.00e+00	2	7	8	1.000000e+00	0.00e+00
hs18	75	8	128	5.000000e+00	4.84e-12	101	8	97	5.000000e+00	1.26e-12	34	19	90	5.000000e+00	2.70e-12
hs19	30	6	62	-6.961814e+03	4.22e-08	110	8	75	-6.961814e+03	6.31e-08	16	11	54	-6.961814e+03	2.60e-13
hs20	202	33	434	4.019873e+01	0.00e+00	126	33	174	4.019873e+01	0.00e+00	0	5	5	4.019873e+01	1.10e-16
hs22	1	4	17	1.000000e-00	2.63e-12	8	4	12	1.000000e-00	2.64e-12	14	8	44	1.000000e+00	1.90e-14
hs23	15	7	54	2.000000e+00	0.00e+00	25	7	39	2.000000e+00	0.00e+00	15	12	50	2.000000e+00	0.00e+00
hs26	56	8	104	1.397957e-12	3.59e-07	56	8	69	9.279517e-11	1.71e-12	31	10	73	1.432758e-11	0.00e+00
hs27	83	8	136	4.000000e-02	1.03e-12	117	7	119	3.999992e-02	9.78e-07	65	13	133	4.000000e-02	1.10e-16
hs29	108	11	181	-2.262742e+01	2.16e-13	121	10	125	-2.262742e+01	7.15e-10	66	22	172	-2.262742e+01	7.10e-15
hs30	7	8	42	1.000000e+00	0.00e+00	16	8	29	1.000000e+00	0.00e+00	7	10	27	1.000000e+00	2.20e-16
hs31	13	4	37	6.000001e+00	0.00e+00	20	4	23	6.000001e+00	0.00e+00	9	6	25	6.000000e+00	0.00e+00
hs32	7	2	16	1.000000e-00	0.00e+00	10	2	11	1.000000e+00	0.00e+00	6	3	11	1.000000e+00	0.00e+00
hs33	5	5	34	-4.000000e+00	0.00e+00	13	5	28	-4.000000e+00	0.00e+00	11	9	36	-4.000000e+00	0.00e+00
hs34	11	6	46	-8.340324e-01	3.92e-11	22	6	33	-8.340324e-01	7.98e-10	9	10	32	-8.340324e-01	1.80e-15
hs39	57	9	103	-1.000000e+00	1.31e-07	104	9	91	-1.000000e+00	1.03e-07	31	11	74	-1.000000e+00	2.20e-13
hs40	11	3	27	-2.500002e-01	1.64e-07	21	4	20	-2.500000e-01	1.12e-11	6	5	18	-2.500000e-01	5.80e-14
hs42	11	4	35	1.385786e+01	2.14e-12	24	4	25	1.385786e+01	2.30e-10	7	5	18	1.385786e+01	7.50e-15
hs43	65	7	108	-4.400000e+01	2.14e-07	79	8	75	-4.400000e+01	9.03e-12	41	12	99	-4.400000e+01	9.00e-13
hs46	108	10	194	2.353883e-10	2.86e-07	126	10	139	1.439046e-09	1.00e-07	44	10	118	4.125292e-13	8.90e-16
hs47	144	13	255	-2.671418e-02	1.36e-08	362	12	246	1.557436e-10	2.62e-10	66	17	164	-2.671418e-02	6.70e-14
hs56	73	9	136	-3.456000e+00	5.15e-10	57	6	48	-3.456000e+00	1.59e-13	21	8	49	-3.456000e+00	2.00e-13
hs57	18	6	58	2.845967e-02	9.56e-12	31	6	50	2.845967e-02	2.05e-12	10	6	25	2.845967e-02	2.80e-17
hs59	75	9	134	-6.749505e+00	0.00e+00	106	9	103	-6.749505e+00	0.00e+00	48	22	124	-6.749505e+00	0.00e+00
hs60	77	9	128	3.256820e-02	2.86e-08	98	7	96	3.256820e-02	6.79e-12	44	10	117	3.256820e-02	1.50e-12
hs61	47	9	116	-1.436461e+02	3.98e-10	92	9	94	-1.436461e+02	7.85e-12	22	13	65	-1.436461e+02	0.00e+00
hs63	50	8	97	9.617152e+02	3.49e-10	54	8	67	9.617152e+02	2.76e-11	45	21	116	9.617152e+02	3.60e-14
hs64	101	8	146	6.299787e+03	1.19e-07	201	8	170	6.299784e+03	1.25e-07	38	14	94	6.299842e+03	2.10e-11

Table 5.4
Numerical results: The 81 Hock-Schittkowski test problems (continued)

	LCLOPT (MINOS)					LCLOPT (SNOPT)					MINOS				
Problem	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol
hs65	60	7	106	9.535288e-01	8.98e-08	73	7	73	9.535288e-01	8.99e-08	128	42	316	9.535289e-01	2.10e-14
hs66	11	4	29	5.181633e-01	1.24e-10	16	4	19	5.181633e-01	3.84e-11	8	6	20	5.181633e-01	7.10e-15
hs70	32	3	50	9.401973e-03	0.00e+00	40	3	39	9.401973e-03	0.00e+00	31	6	63	9.401973e-03	0.00e+00
hs71	32	6	61	1.701402e+01	1.40e-08	55	6	50	1.701402e+01	1.91e-08	23	9	52	1.701402e+01	1.10e-07
hs72	114	13	182	7.252304e+02	2.78e-07	280	11	196	7.272718e+02	7.11e-08	52	17	149	7.276794e+02	4.10e-10
hs73	1	2	9	2.989438e+01	5.76e-10	1	2	5	2.989438e+01	5.76e-10	8	5	15	2.989438e+01	5.30e-09
hs74	32	5	52	5.126498e+03	1.11e-16	107	5	57	5.126498e+03	2.21e-16	11	8	24	5.126498e+03	4.50e-13
hs75	26	4	39	5.174413e+03	2.45e-16	96	4	44	5.174413e+03	0.00e+00	8	6	15	5.174413e+03	2.30e-13
hs77	133	10	208	9.908760e+00	3.23e-11	220	13	213	2.415051e-01	6.96e-11	51	11	120	2.415051e-01	1.30e-08
hs78	29	5	61	-2.919700e+00	3.41e-09	68	7	67	-2.919700e+00	7.48e-11	20	8	54	-2.919700e+00	8.90e-16
hs79	58	6	92	7.877682e-02	6.71e-09	77	6	65	7.877682e-02	2.10e-09	21	8	50	7.877682e-02	9.80e-10
hs80	44	7	85	5.394985e-02	6.92e-09	98	9	94	5.394985e-02	6.04e-13	24	9	57	5.394985e-02	4.50e-13
hs81	45	7	86	5.394985e-02	8.22e-09	97	9	93	5.394985e-02	7.49e-09	24	8	56	5.394985e-02	1.80e-10
hs83	4	4	20	-3.066554e+04	1.35e-10	6	4	19	-3.066554e+04	1.35e-10	4	5	8	-3.066554e+04	0.00e+00
hs84	12	4	39	-5.280335e+06	0.00e+00	367	4	177	-5.280335e+06	5.73e-12	16	7	42	-5.280335e+06	0.00e+00
hs88	41	13	112	1.362657e+00	1.81e-11	86	13	81	1.362656e+00	3.88e-10	29	15	65	1.362657e+00	7.00e-15
hs89	111	16	214	1.362657e+00	8.55e-12	85	11	90	1.362656e+00	1.87e-10	94	18	222	1.362657e+00	3.30e-12
hs90	55	10	111	1.362657e+00	2.56e-11	105	11	116	1.362657e+00	2.49e-12	52	12	116	1.362657e+00	3.00e-13
hs91	167	12	258	1.362651e+00	2.67e-09	211	12	185	1.362644e+00	6.47e-09	105	13	223	1.362657e+00	1.50e-11
hs92	159	10	237	1.362639e+00	9.31e-09	83	10	66	1.362657e+00	5.29e-12	56	11	112	1.362657e+00	5.20e-12
hs93	49	5	81	1.350760e+02	1.72e-09	60	5	60	1.350760e+02	3.51e-12	20	8	43	0.000000e+00	[i]2.10e+00
hs95	1	1	5	1.561953e-02	0.00e+00	22	1	13	1.561953e-02	0.00e+00	1	2	3	1.561953e-02	8.90e-16
hs96	1	1	5	1.561953e-02	0.00e+00	22	1	13	1.561953e-02	0.00e+00	1	2	3	1.561953e-02	8.90e-16
hs97	7	2	16	4.071246e+00	0.00e+00	124	2	48	4.071246e+00	0.00e+00	63	10	105	3.135809e+00	0.00e+00
hs98	7	2	16	4.071246e+00	0.00e+00	124	2	48	4.071246e+00	0.00e+00	49	10	88	3.135809e+00	0.00e+00
hs99	190	5	231	-8.310799e+08	5.69e-13	1255	5	282	-8.310799e+08	9.73e-12	39	6	57	-8.310799e+08	2.00e-09
hs99exp	380	7	499	-1.008063e+09	2.67e-16	2576	2	741	-2.500387e+12	9.83e-02	106	8	201	-1.008062e+09	3.90e-08
hs100lnp	88	7	137	6.806301e+02	2.33e-10	148	7	126	6.806301e+02	7.32e-11	52	10	130	6.806301e+02	1.40e-14
hs100mod	96	6	151	6.787547e+02	2.26e-13	170	6	140	6.787547e+02	4.15e-14	49	10	117	6.787547e+02	4.00e-13
hs100	114	7	170	6.806301e+02	5.30e-15	194	6	137	6.806301e+02	2.35e-10	51	10	127	6.806301e+02	3.60e-15
hs101	948	13	1259	1.809754e+03	5.78e-07	4383	14	2343	1.809765e+03	1.04e-10	5602	151	15940	1.809765e+03	1.60e-15
hs102	356	13	523	9.118806e+02	1.99e-11	3485	13	1833	9.118806e+02	1.17e-13	374	16	969	9.118806e+02	3.30e-14
hs103	525	15	740	5.436680e+02	6.18e-12	3441	14	1784	5.436679e+02	5.16e-09	955	32	2549	5.436680e+02	1.50e-12
hs104	175	7	264	3.951163e+00	9.73e-09	170	7	113	3.951163e+00	1.60e-09	38	9	81	3.951163e+00	2.20e-16
hs106	221	39	524	7.049248e+03	1.35e-15	527	28	560	7.049318e+03	0.00e+00	238	67	652	7.049248e+03	0.00e+00
hs107	34	5	57	5.055012e+03	5.03e-11	54	5	26	5.055012e+03	2.60e-10	12	6	17	5.055012e+03	4.90e-15
hs108	135	10	212	-8.660254e-01	2.39e-09	507	17	303	-8.660259e-01	3.18e-07	75	11	156	-8.660254e-01	5.80e-15

Table 5.4

Numerical results: The 81 Hock-Schittkowski test problems (continued)

	LCLOPT (MINOS)					LCLOPT (SNOPT)					MINOS				
Problem	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol
hs109	15	5	22	5.519560e+03	[i]2.98e+01	1697	6	491	-7.540990e+09	[i]7.78e+00	175	60	412	5.326851e+03	1.50e-11
hs111lnp	372	8	557	-4.776117e+01	8.10e-07	441	9	463	-4.776109e+01	6.20e-11	158	15	417	-4.776109e+01	3.50e-11
hs111	311	9	437	-4.776109e+01	4.75e-09	437	9	370	-4.776109e+01	1.63e-10	158	15	417	-4.776109e+01	3.50e-11
hs113	98	6	142	2.430621e+01	7.77e-11	161	7	115	2.430621e+01	9.32e-14	56	10	137	2.430621e+01	2.80e-14
hs114	40	8	84	-1.768807e+03	8.48e-09	63	6	48	-1.768805e+03	2.36e-14	29	12	79	-1.768807e+03	1.50e-09
hs116	139	15	217	9.758752e+01	4.45e-08	239	6	98	9.759103e+01	1.38e-12	52	7	96	9.758751e+01	3.60e-06
hs117	81	5	123	3.234868e+01	0.00e+00	136	6	76	3.234868e+01	0.00e+00	83	10	157	3.234868e+01	0.00e+00

TABLE 5.5
Summary: The 81 selected Hock-Schittkowski test problems

	LCLOPT		
	(MINOS)	(SNOPT)	MINOS
Optimal	80	80	80
False Infeasibility	1	1	1

On *hs13*, all the solvers reached different solutions. However, the linear independence constraint qualification does not hold at the solution of this problem—this violates the required assumptions for both LCLOPT and MINOS.

Recall that LCLOPT and MINOS use only first derivatives and so may not necessarily converge to local solutions of a problem. For example, LCLOPT (in both versions) converged to a known local solution of *hs16*, but MINOS converged to some other first-order point. In contrast, MINOS converged to the known local solutions of *hs97* and *hs98*, while LCLOPT (in both versions) converged to other first-order points. There are similar differences for problems *hs47* and *hs77*.

5.4 A Selection of CUTE Test Problems

With the `select` utility [Bongartz et al., 1995], we extracted from the CUTE test suite dated September 7, 2000, problems with the following characteristics (`*` is a wild-card character):

```
Objective function type         : *
Constraint type                 : Q O (quadratic, general nonlinear)
Regularity                      : R   (smooth)
Degree of available derivatives : 1   (first derivatives, at least)
Problem interest                : M R (modeling, real applications)
Explicit internal variables     : *
Number of variables             : *
Number of constraints           : *
```

These criteria yield 108 problems. We exclude 66 problems from this set of 108 for the following reasons:

- 33 problems do not have AMPL versions: *car2, c-reload, dembo7, drugdis, durgdise, errinbar, junkturn, leaknet, lubrif, mribasis, nystrom5, orbit2, reading4, reading5,* *reading6, reading7, reading8, reading9, rotodisc, saromm, saro, tenbars1, tenbars2, tenbars3, tenbars4, trigger, truspyr1, truspyr2, zamb2, zamb2-8, zamb2-9, zamb2-10*, and *zamb2-11*;

- 21 problems cause system errors when evaluated by either the AMPL MEX interface or by MINOS (when invoked from AMPL): *brainpc2, brainpc3, brainpc4, brainpc5, brainpc6, brainpc7, brainpc8, brainpc9, bratu2dt, cresc132, csfi1, csfi2, drcav1lq, drcav2lq, drcav3lq, kissing, lakes, porous1, porous2, trainf*, and *trainh*;
- The AMPL versions of 12 problems are formulated with no nonlinear constraints: *drcavty1, drcavty2, drcavty3, flosp2hh, flosp2hl, flosp2hm, flosp2th, flosp2tl, flosp2tm, methanb8, methanl8*, and *res*.

The dimensions of 17 of the remaining 42 problems can be adjusted. In all cases, the solvers were applied to the largest problem versions that would not cause the system to page memory to disk. Table A.3 on p. 76 summarizes the dimensions of the selected problems and indicates problems that can vary in size. Table 5.7 on the facing page shows the numerical results.

The version of LCLOPT using MINOS solved 36 of 42 problems to first-order optimality, while the version using SNOPT solved 34 problems to first-order optimality. MINOS solved 35 problems to first-order optimality. Table 5.6 summarizes these results. We note that LCLOPT, in one of its two versions, solves every problem except for *heart6*, which it declares infeasible. With the exception of *cresc50*, LCLOPT with SNOPT does not seem to suffer (on successful solves) from excessive minor iterations due to subproblem restarts as it does on the COPS problems.

TABLE 5.6
Summary: The 42 selected CUTE test problems

	LCLOPT		
	(MINOS)	(SNOPT)	MINOS
Optimal	36	34	35
False Infeasibility	4	3	2
Terminated	1	4	1
Cannot be improved	1	1	1
Unbounded/badly scaled			3

Table 5.7

Numerical results: The 42 selected CUTE test problems

	LCLOPT (MINOS)					LCLOPT (SNOPT)					MINOS				
Problem	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol	Mnr	Mjr	Fcn	$f(x_*)$	Viol
aircrfta	4	3	16	0.000000e+00	3.30e-12	4	3	7	0.000000e+00	3.30e-12	1	5	7	0.000000e+00	4.50e-13
airport	463	8	715	4.795270e+04	1.33e-09	929	9	153	4.795270e+04	7.06e-12	220	13	516	4.795270e+04	3.00e-13
bdvalue	1	1	5	0.000000e+00	1.25e-08	84	1	4	5.404680e-16	1.25e-08	1	2	2	0.000000e+00	[u]1.50e-08
brainpc0	7202	8	386	1.499639e-03	5.75e-08	7058	8	110	1.499639e-03	5.01e-08	179	10	235	1.499639e-03	1.50e-13
brainpc1	7205	4	1713	6.267181e-07	9.99e-08	7230	3	9	6.081161e-05	4.31e-07	1373	45	2724	3.961479e-04	[f]9.30e-10
bratu2d	60	2	9	0.000000e+00	1.12e-06	135	2	8	3.069767e-14	1.49e-07	0	1	1	0.000000e+00	7.90e-04
bratu3d	64	3	13	0.000000e+00	5.05e-09	136	3	10	0.000000e+00	5.05e-09	64	5	6	0.000000e+00	4.40e-10
cantilvr	193	12	286	1.339956e+00	1.14e-09	433	12	343	1.339956e+00	1.15e-09	71	16	180	1.339956e+00	1.40e-13
cbratu2d	227	1	7	-3.995068e-15	4.17e-07	228	1	4	-1.235990e-16	4.53e-08	21	3	3	0.000000e+00	2.20e-16
cbratu3d	516	1	5	0.000000e+00	5.05e-07	518	1	4	1.925543e-16	1.01e-07	320	3	4	0.000000e+00	1.60e-17
chandheq	200	9	140	0.000000e+00	7.25e-07	200	9	22	1.035179e-13	7.25e-07	0	11	11	0.000000e+00	7.00e-07
chemrcta	2527	6	55	5.471814e+03	[i]2.19e+03	2556	6	13	8.316266e+00	[i]1.11e+04	1689	4	264	0.000000e+00	1.90e-10
chemrctb	1080	6	67	1.973342e+02	[i]6.73e+02	44451	6	794	1.320199e+01	[f]1.07e+01	64	33	150	0.000000e+00	4.50e-13
clnlbeam	499	1	5	3.500000e+02	1.47e-09	3024	3	70	3.448768e+02	3.21e-08	4404	112	9528	3.481482e+02	7.00e-13
coolhans	4	6	28	-1.247266e-15	6.66e-13	7	7	17	-1.451530e-15	1.29e-10	4	4	6	0.000000e+00	5.70e-21
cresc100	2197	25	3365	5.676027e-01	5.64e-09	60000	1	3665	8.840751e+05	[t]0.00e+00	131	24	315	5.676027e-01	5.60e-16
cresc4	746	18	1037	8.718975e-01	1.28e-09	18226	1	10708	1.955855e+05	[t]5.02e-03	464	42	1149	8.718975e-01	4.40e-16
cresc50	1171	12	1654	7.326584e+00	[f]0.00e+00	19296	65	9337	5.940539e-01	5.17e-16	322	190	814	5.932301e-01	3.50e-10
deconvc	45	4	89	5.652501e-15	3.42e-12	451	4	86	2.293333e-12	7.83e-10	39	4	84	6.469070e-18	1.90e-12
disc2	1005	25	1296	1.677243e+00	1.47e-07	1473	19	781	15625000e+00	5.97e-11	493	32	826	0.000000e+00	[u]2.10e+09
discs	14734	25	17910	1.664163e+01	1.56e-08	24219	2	13142	4.004581e+04	[t]9.24e-02	1141	18	653	8.566530e+02	[i]7.80e+01
dnieper	144	4	149	1.874401e+04	8.59e-11	435	5	70	1.874401e+04	3.08e-11	39	4	33	1.874401e+04	6.30e-14
grouping	42	1	48	1.385040e+01	6.08e-16	42	1	3	1.385040e+01	0.00e+00	0	2	2	1.385040e+01	0.00e+00
hadamard	1	2	9	1.000000e+00	0.00e+00	0	1	3	1.000000e+00	0.00e+00	0	6	6	0.000000e+00	[i]8.00e+00
heart6	910	18	1278	2.529364e+01	[i]8.89e-05	4024	15	2450	3.545563e+03	[i]2.14e-04	49	37	137	0.000000e+00	7.10e-15
heart8	393	12	560	1.064338e+06	[i]2.57e-01	444	6	192	-2.163292e+03	[i]1.49e+08	51	24	129	0.000000e+00	8.90e-15
himmelbk	164	9	228	5.181434e-02	7.98e-11	243	8	88	5.181434e-02	2.16e-11	109	9	171	5.181434e-02	2.40e-10
launch	754	103	1558	9.004903e+00	1.50e-11	2877	97	3091	9.004903e+00	6.20e-14	32	2	45	3.927354e-01	[u]5.90e+08
manne	19	1	24	-9.745726e-01	0.00e+00	0	1	3	-9.745726e-01	0.00e+00	37	2	10	-9.745726e-01	0.00e+00
prodpl0	43	8	77	6.091924e+01	5.59e-12	349	7	186	6.091924e+01	7.48e-09	42	10	40	6.091924e+01	6.00e-14
prodpl1	65	6	95	5.303702e+01	5.03e-11	350	5	173	5.303702e+01	2.39e-13	56	9	69	5.303702e+01	2.20e-16
reading1	13000	2	10068	-5.049306e-02	[t]1.30e-01	7483	1	42	-1.812989e-04	[s]1.76e-03	4420	117	10578	-1.604743e-01	4.50e-13
reading3	101	1	333	1.033976e-27	8.32e-25	101	1	3	0.000000e+00	0.00e+00	1	2	4	0.000000e+00	0.00e+00
rk23	51	8	97	8.333333e-02	6.38e-11	205	8	114	8.333333e-02	7.64e-11	27	9	55	8.333333e-02	7.70e-11
robot	228	12	343	1.307186e+02	1.87e-12	125	10	90	5.462841e+00	5.48e-10	107	16	279	5.462841e+00	1.50e-13
sreadin3	3637	1	642	-2.227114e-05	2.56e-07	3936	1	7	-2.227115e-05	1.88e-10	1062	3	3	-1.340287e-05	6.80e-13

Table 5.7
Numerical results: The 42 selected CUTE test problems (continued)

Problem	LCLOPT (MINOS) Mnr	Mjr	Fcn	$f(x_*)$	Viol	LCLOPT (SNOPT) Mnr	Mjr	Fcn	$f(x_*)$	Viol	MINOS Mnr	Mjr	Fcn	$f(x_*)$	Viol
ssebnln	229	3	233	1.617060e+07	0.00e+00	4048	3	291	1.617060e+07	1.73e-12	94	3	49	1.617060e+07	0.00e+00
ssnlbeam	141	3	162	3.377725e+02	1.45e-08	102	4	33	3.377725e+02	1.03e-13	58	6	120	3.377725e+02	2.90e-13
svanberg	3409	10	5263	1.671433e+03	9.81e-11	27923	10	549	1.671433e+03	2.28e-10	3706	93	7280	1.671433e+03	2.70e-14
swopf	170	6	168	6.786017e-02	1.42e-08	472	8	142	6.785978e-02	5.90e-07	84	11	115	6.786018e-02	7.30e-13
twobars	22	6	48	1.508652e+00	1.81e-08	31	6	33	1.508652e+00	1.97e-08	11	7	28	1.508652e+00	6.20e-09
ubh5	6810	4	9032	1.116013e+00	1.60e-12	34314	4	10882	1.115192e+00	2.13e-08	8035	200	16438	5.556909e+00	^{t}9.30e-05

Chapter 6

Conclusions and Future Work

The stabilized LCL method developed in this dissertation is a generalization of the augmented Lagrangian methods discussed in Chapter 3. From (ELC'_k), the stabilized LCL subproblem can be recast as

$$\boxed{(\mathrm{ELC}''_k) \qquad \underset{x}{\text{minimize}} \quad \mathcal{L}_k(x) + \sigma_k \|\bar{c}_k(x)\|_1 \qquad \text{subject to} \quad x \geq 0,}$$

with solution (x_k^*, z_k^*). This immediately reveals the stabilized LCL method's intimate connection with both the augmented Lagrangian function and the BCL method. Near a solution, the ℓ_1-penalty term $\sigma_k \|\bar{c}_k(x)\|_1$ serves to keep the iterates close to the current constraint linearizations. Far from a solution, it gives the method an opportunity to deviate from the linearizations. For values of σ_k over a threshold value, the linearized constraints are satisfied exactly, as required by the LCL method.

The stabilized LCL method shares the strengths of its predecessors: it is globally convergent (the BCL advantage) and it has fast local convergence (the LCL advantage). The ℓ_1-penalty function brings the two together. Because the stabilized LCL method operates in a reduced space given by the linearized constraints (like the LCL method), it does not suffer from the ill-conditioning effects that can plague BCL methods.

6.1 The Importance of Early Termination

The numerical results presented in Chapter 5 demonstrate that MINOS successfully solved many of the test problems using relatively few minor iterations. MINOS terminates its progress on each of its subproblems after 40 iterations (to avoid a refactorization of the current basis, which by default occurs every 50 iterations). In contrast, LCLOPT attempts to constrain the subproblem iterations via an initially loose optimality tolerance (we set $\omega_0 = \sqrt{\omega_*}$ for the runs shown in Chapter 5). A potential weakness of this approach vis à vis MINOS is that there is no a priori bound on the number of subproblem iterations. MINOS's aggressive (but heuristic) strategy seems effective in keeping the total minor iteration counts low. This property is particularly important during the early major iterations, when the current solution estimates are poor.

It may be possible to emulate the MINOS strategy and still satisfy the LCLOPT requirement that the subproblem optimality tolerances ω_k converge to zero (cf. Lemma 3.5 on p. 34). For example, LCLOPT might truncate the subproblem solutions after a fixed number of iterations, and only gradually increase the iteration limit on successive major iterations. Especially during early major iterations, such a strategy may keep the accumulated number of subproblem iterations small. During later major iterations, the strategy would still ensure that the subproblem solver returns solutions within the prescribed tolerance ω_k.

On the other end of the performance spectrum lies the issue of recovering LCL's fast local convergence rate under inexact solves (cf. §3.4.1). Bräuninger [1981] proves that the quadratic convergence rate of Robinson's method is retained when ω_k is reduced at a rate $O(\|F(x_k, y_k, z_k)\|^2)$ (cf. Theorem 2.8). The first-order KKT conditions (3.16) for the LCL subproblem can be expressed as

$$\begin{pmatrix} \nabla^2_{xx}\mathcal{L}_k(x_k) & J_k^T \\ J_k & \end{pmatrix} \begin{pmatrix} p \\ -y \end{pmatrix} + O(\|p\|^2) = \begin{pmatrix} -g_k + J_k^T y_k \\ -c_k \end{pmatrix}, \tag{6.1}$$

where $p \stackrel{\text{def}}{=} x - x_k$, and a first-order Taylor expansion was used to derive the residual term $O(\|p\|^2)$. (We have ignored bound constraints for the moment. Robinson [1972, 1974] shows that the correct active set is identified by the subproblems near a solution.) The nonlinear equations (6.1) are closely related to the linear equations that would be derived from applying Newton's method to (3.16) (again, ignoring bound constraints). In that case, the theory from inexact Newton methods [Dembo et al., 1982] predicts that the quadratic convergence rate is recovered when the residual error is reduced at the rate $O(\|F(x_k, y_k, z_k)\|)$. The similarity between (6.1) and the Newton equations hints at the possibility of recovering the quadratic convergence rate of the LCL and stabilized LCL methods by reducing ω_k at the rate $O(\|F(x_k, y_k, z_k)\|)$. We note, however, that stronger assumptions may be needed on the smoothness of the nonlinear functions. This deserves more study.

6.2 Keeping the Penalty Parameter Small

Preliminary experimentation reveals that a small penalty parameter ρ_k can significantly reduce the difficulty of each subproblem solve. BCL methods require that ρ_k be larger than some threshold value $\bar{\rho}$—partly a consequence of Debreu's Theorem (see §1.5). In contrast, LCL methods can converge when $\rho_k \equiv 0$ if they are started near a solution (see §2.8).

The challenge here is to find a strategy that can keep ρ_k small or reduce it without destabilizing the method. A tentative strategy might be to reduce ρ_k only finitely many times. This approach does not violate the hypotheses of Lemma 3.5, and may be effective in practice. A form of this strategy was used for the runs shown in Chapter 5.

6.3 A Second Derivative LC Solver

We prove in §3.6 that the stabilized LCL method will converge to second-order stationary points if the LC subproblems are solved to second-order points (for example, by using a second-derivative LC solver). In practice, however, a second-derivative LC solver may be most useful as a means of reducing the overall computational work required by the stabilized LCL method.

The stabilized LCL method is largely independent of the method in which its subproblems are solved. An LC solver using second derivatives is likely to require fewer iterations (and hence less computational work) for the solution of each of the subproblem. We would expect the number of required major iterations to remain constant if each subproblem solution is computed to within the prescribed tolerance ω_k. However, we would expect to *reduce* the number of required major iterations if a MINOS-like strategy is used to terminate the subproblems (see §6.1). Over the same number of iterations, a subproblem solver using second derivatives may make more progress towards a solution than a first-derivative solver.

Any future implementation of the stabilized LCL method would ideally be flexible enough to allow for a variety of solvers to be used for the LC subproblems. The choice of the subproblem solver could then be guided by the characteristics of the optimization problem at hand. In particular, the advent of automatic differentiation makes second derivatives increasingly available for certain problem classes (e.g., within recent versions of GAMS and AMPL) and for more general functions defined by Fortran or C code (notably ADIFOR and ADIC [Bischof and Roh, 1997; Bischof et al., 1998]). These may be used by SQP and interior methods for nonlinearly constrained (NC) problems (e.g., LOQO [Shanno and Vanderbei, 1999]). Certain theoretical challenges might be avoided, however, by developing specialized second-derivative LC solvers. Such LC solvers could be extended readily to general NC problems by incorporating them into the stabilized LCL algorithm.

Appendix A

Test-Problem Set Description

The tables in this Appendix show the dimensions of each test problem used in the numerical results of Chapter 5. Each table uses the following heads:

Head	Dimension
m	Constraints (linear and nonlinear)
m_c	Nonlinear constraints
n	Variables
n_c	Variables appearing nonlinearly in c
n_f	Variables appearing nonlinearly in f

TABLE A.1
Dimensions: The 12 selected COPS test problems

Problem	m	m_c	n	n_c	n_f
camshape	1604	801	800	800	0
catmix	1603	1600	2403	2403	0
chain	204	1	402	201	402
channel	800	400	800	800	0
elec	201	200	600	600	600
gasoil400	4004	3200	4003	4003	202
marine	1208	800	1215	1215	344
methanol	2406	1800	2405	1605	1670
pinene	4006	3000	4005	2405	2469
polygon	1377	1225	100	100	100
robot	2414	2400	3611	3209	0
rocket	2409	1200	1605	1605	0
steering	2011	1600	2007	1204	0

Table A.2
Dimensions: The 81 selected Hock-Schittkowski test problems

Problem	m	m_c	n	n_c	n_f
hs6	2	1	2	1	1
hs7	2	1	2	2	1
hs8	2	2	2	2	0
hs10	2	1	2	2	0
hs11	2	1	2	1	2
hs12	2	1	2	2	2
hs13	2	1	2	1	2
hs14	3	1	2	2	2
hs15	3	2	2	2	2
hs16	3	2	2	2	2
hs17	3	2	2	2	2
hs18	3	2	2	2	2
hs19	3	2	2	2	2
hs20	4	3	2	2	2
hs22	3	1	2	1	2
hs23	6	4	2	2	2
hs26	2	1	3	3	3
hs27	2	1	3	1	3
hs29	2	1	3	3	3
hs30	2	1	3	2	3
hs31	2	1	3	2	3
hs32	3	1	3	1	3
hs33	3	2	3	3	1
hs34	3	2	3	2	0
hs39	3	2	4	3	0
hs40	4	3	4	3	4
hs42	2	1	3	2	3
hs43	4	3	4	4	4
hs46	3	2	5	4	5
hs47	4	3	5	4	5
hs56	5	4	7	4	7
hs57	2	1	2	2	2
hs59	4	3	2	2	2
hs60	2	1	3	3	3
hs61	3	2	3	2	3
hs63	3	1	3	3	3
hs64	2	1	3	3	3
hs65	2	1	3	3	3
hs66	3	2	3	2	0
hs70	2	1	4	2	4
hs71	3	2	4	4	4
hs72	3	2	4	4	0
hs73	4	1	4	4	0

Table A.2
Dimensions: The 81 selected Hock-Schittkowski test problems (continued)

Problem	m	m_c	n	n_c	n_f
hs74	5	3	4	2	4
hs75	5	3	4	2	4
hs77	3	2	5	4	5
hs78	4	3	5	5	5
hs79	4	3	5	4	5
hs80	4	3	5	5	5
hs81	4	3	5	5	5
hs83	4	3	5	5	3
hs84	4	3	5	5	5
hs88	2	1	2	2	2
hs89	2	1	3	3	3
hs90	2	1	4	4	4
hs91	2	1	5	5	5
hs92	2	1	6	6	6
hs93	3	2	6	6	6
hs95	5	4	6	5	0
hs96	5	4	6	5	0
hs97	5	4	6	5	0
hs98	5	4	6	5	0
hs99	15	14	19	7	7
hs99exp	22	21	28	7	8
hs100lnp	3	2	7	4	7
hs100mod	5	4	7	5	7
hs100	5	4	7	5	7
hs101	7	6	7	7	7
hs102	7	6	7	7	7
hs103	7	6	7	7	7
hs104	6	5	8	8	4
hs106	7	3	8	8	0
hs107	7	6	9	5	7
hs108	14	13	9	9	9
hs109	11	8	9	9	2
hs111lnp	4	3	10	10	10
hs111	4	3	10	10	10
hs113	9	5	10	5	10
hs114	12	6	10	7	8
hs116	16	10	13	10	0
hs117	6	5	15	5	5

TABLE A.3

Dimensions: The 42 selected CUTE test problems. Problem names marked with a † are variable in dimension.

Problem	m	m_c	n	n_c	n_f
aircrfta	5	5	5	5	0
airport	43	42	84	84	84
bdvalue†	1000	1000	1000	1000	0
brainpc0	6901	6898	6905	6901	6904
brainpc1	6901	6898	6905	6901	6904
bratu2d†	4900	4900	4900	4900	0
bratu3d†	512	512	512	512	0
cantilvr	2	1	5	5	0
cbratu2d†	882	882	882	882	0
cbratu3d†	1024	1024	1024	1024	0
chandheq†	100	100	100	100	0
chemrcta†	2000	1996	2000	1996	0
chemrctb†	1000	998	1000	998	0
clnlbeam†	1001	500	1499	499	1000
coolhans	9	3	9	9	0
cresc100	201	200	6	6	3
cresc4	9	8	6	6	3
cresc50	101	100	6	6	3
deconvc	2	1	51	11	51
disc2	24	23	28	28	0
discs	67	66	33	33	0
dnieper	25	24	57	48	49
grouping	126	100	100	100	100
hadamard†	257	128	65	64	65
heart6	6	6	6	6	0
heart8	8	6	8	8	0
himmelbk	15	12	24	24	0
launch	30	11	25	25	21
manne†	731	364	1094	364	729
prodpl0	30	4	60	10	0
prodpl1	30	4	60	10	0
reading1†	5001	5000	10001	10000	10000
reading3†	103	101	202	202	202
rk23	12	7	17	7	0
robot	3	2	7	7	7
sreadin3†	5001	5000	10000	9998	9998
ssebnln	97	24	192	48	0
ssnlbeam†	21	10	31	11	22
svanberg†	1001	1000	1000	1000	1000
swopf	92	49	82	69	0
twobars	3	2	2	2	2
ubh5†	14001	2000	19997	6003	0

Bibliography

Arrow, K. J., and R. M. Solow, Gradient methods for constrained maxima, with weakened assumptions, in *Studies in Linear and Nonlinear Programming*, edited by K. J. Arrow, L. Hurwicz, and H. Uzawa, pp. 166–176, Stanford University Press, Stanford, California, 1958.

Bertsekas, D. P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, 1982.

Best, M. J., J. Bräuninger, K. Ritter, and S. M. Robinson, A globally and quadratically convergent algorithm for general nonlinear programming problems, *Computing*, *26*, 141–155, 1981.

Biggs, M. C., Constrained minimization using recursive equality quadratic programming, in *Numerical Methods for Nonlinear Optimization*, edited by F. A. Lootsma, Academic Press, London, 1972.

Bischof, C., and L. Roh, ADIC: An extensible automatic differentiation tool for ANSI-C, *Software: Practice and Experience*, *27*, 1427–1456, 1997.

Bischof, C., A. Carle, P. Hovland, P. Khademi, and A. Mauer, ADIFOR 2.0 Users' Guide, *Tech. Rep. 192*, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, Illinois 60439, 1998.

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

Bräuninger, J., A modification of Robinson's algorithm for general nonlinear programming problems requiring only approximate solutions of subproblems with linear equality constraints, in *Optimization Techniques. Proceedings of the 8th IFIP Conference, Part 2*, pp. 33–41, Würzburg, 1977.

Bräuninger, J., A globally convergent version of Robinson's algorithm for general nonlinear programming problems without using derivatives, *J. Opt. Theory Applic.*, *35*, 195–216, 1981.

Brooke, A., D. Kendrick, and A. Merraus, *GAMS: A User's Guide*, Scientific Press, South San Francisco, 1988.

Conn, A. R., N. I. M. Gould, and P. L. Toint, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer-Verlag, Berlin, 1991a.

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

Conn, A. R., N. I. M. Gould, A. Sartenaer, and P. L. Toint, Convergence properties of an augmented Lagrangian algorithm for optimization with a combination of general equality and linear constraints, *SIAM J. Optim.*, *6*, 674–703, 1996.

Conn, A. R., N. I. M. Gould, and P. L. Toint, *Trust-Region Methods*, MPS-SIAM Series on Optimization, SIAM Publications, Philadelphia, 2000.

Courant, R., Variational methods for the solution of problems with equilibrium and variation, *Bull. Amer. Math. Soc.*, *49*, 1–23, 1943.

Debreu, G., Definite and semidefinite quadratic forms, *Econometrica*, *20*, 295–300, 1952.

Dembo, R. S., S. C. Eisenstat, and T. Steihaug, Inexact Newton methods, *SIAM J. Numer. Anal.*, *19*, 400–408, 1982.

Dolan, E. D., and J. J. Moré, Benchmarking optimization software with COPS, *Tech. Rep. ANL/MCS-246*, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, Illinois 60439, 2000, revised January 2, 2001.

Drud, A. S., A large scale GRG code, *ORSA J. Computing*, *6*, 207–216, 1994.

Fiacco, A. V., and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York, 1968.

Fletcher, R., Methods related to lagrangian functions, in *Numerical Methods for Constrained Optimization*, edited by P. E. Gill and W. Murray, pp. 219–239, Academic Press, London, 1974.

Fletcher, R., An ℓ_1 penalty method for nonlinear constraints, in *Numerical Optimization*, edited by P. T. Boggs, R. H. Byrd, and R. B. Schnabel, pp. 26–40, SIAM, Boulder, Colorado, 1984.

Fletcher, R., *Practical Methods of Optimization*, second ed., John Wiley and Sons, New York, 1987.

Fourer, R., D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Scientific Press, San Francisco, 1993.

Gay, D. M., Hooking your solver to AMPL, *Tech. Rep. 97-4-06*, Computing Sciences Research Center, Bell Laboratories, Murray Hill, New Jersey, 07974, 1997.

Gill, P. E., W. Murray, and S. M. Picken, E04UAF, E04VAF, E04VBF, E04WAF, constrained optimization using sequential augmented Lagrangian methods, in *The NAG Fortran Library Manual Mark 6*, edited by Numerical Algorithms Group Limited, chap. E04, pp. 170–220, Numerical Algorithms Group Limited, Wilkinson House, Jordan Hill Road, Oxford, England, 1977, withdrawn March 13, 1988.

Gill, P. E., W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, San Diego, California, 1981.

Gill, P. E., W. Murray, and M. A. Saunders, User's guide for SNOPT 5.3: A fortran package for large-scale nonlinear programming, *Tech. Rep. 98-1*, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California 94305, 1997.

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

Gould, N. I. M., On the convergence of a sequential penalty function method for constrained minimization, *SIAM J. Numer. Anal.*, *26*, 107–128, 1989.

Griffith, R. E., and R. A. Stewart, A nonlinear programming technique for the optimization of continuous processing systems, *Mgmnt Sci.*, *7*, 379–392, 1961.

Hestenes, M. R., Multiplier and gradient methods, *J. Opt. Theory Applic.*, *4*, 303–320, 1969.

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

Luenberger, D. G., *Linear and Nonlinear Programming*, second ed., Addison-Wesley, 1984.

Mangasarian, O. L., *Nonlinear Programming*, McGraw-Hill, New York, 1969.

MathWorks, *MATLAB User's Guide*, The MathWorks, Inc., Natick, Massachussetts, 1992.

MathWorks, *MATLAB: External Interfaces*, Natick, Massachussetts, 1995.

Murray, W., An algorithm for constrained minimization, in *Optimization*, edited by R. Fletcher, Academic Press, London and New York, 1969.

Murray, W., Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions, *J. Opt. Theory Applic.*, *7*, 189–196, 1971.

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

Murtagh, B. A., and M. A. Saunders, A projected Langrangian algorithm and its implementation for sparse nonlinear constraints, *Mathematical Programming Study*, *16*, 84–117, 1982.

Nash, S. G., and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, New York, 1996.

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

Ortega, J. M., and W. C. Rheinboldt, *Iterative Solutions of Nonlinear Equations in Several Variables*, Academic Press, London, 1970.

Palacios-Gomez, F., L. Lasdon, and E. Engquist, Nonlinear optimization by successive linear programming, *Mgmnt Sci.*, *28*, 1106–1120, 1982.

Powell, M. J. D., A method for nonlinear constraints in minimization problems, in *Optimization*, edited by R. Fletcher, chap. 19, Academic Press, London and New York, 1969.

Robinson, S. M., A quadratically-convergent algorithm for general nonlinear programming problems, *Math. Prog.*, *3*, 145–156, 1972.

Robinson, S. M., Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms, *Math. Prog.*, *7*, 1–16, 1974.

Rosen, J. B., Two-phase algorithm for nonlinear constraint problems, in *Nonlinear Programming 3*, edited by O. Mangasarian, R. Meyer, and S. Robinson, pp. 97–124, Academic Press, New York, 1978.

Rosen, J. B., and J. Kreuser, A gradient projection algorithm for non-linear constraints, in *Numerical Methods for Nonlinear Optimization*, edited by F. A. Lootsma, pp. 297–300, Academic Press, London, 1972.

Shanno, D., and R. Vanderbei, An interior-point algorithm for nonconvex nonlinear programming, *Computational Optimization and Applications*, *13*, 231–252, 1999.

Tapia, R. A., Diagonalized multiplier methods and quasi-Newton methods for constrained optimization, *J. Opt. Theory Applic.*, *22*, 135–194, 1977.

Van Der Hoek, G., Asymptotic properties of reduction methods applying linearly equality constrained reduced problems, *Mathematical Programming Study*, *16*, 162–189, 1982.

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

Author Index