Preprint ANL/MCS-P1015-1202, Dec 2002 (Revised Nov 2003, Mar 2004)
Mathematics and Computer Science Division
Argonne National Laboratory

A GLOBALLY CONVERGENT LINEARLY CONSTRAINED LAGRANGIAN METHOD FOR NONLINEAR OPTIMIZATION

MICHAEL P. FRIEDLANDER* AND MICHAEL A. SAUNDERS†

Abstract. For optimization problems with nonlinear constraints, linearly constrained Lagrangian (LCL) methods solve a sequence of subproblems of the form "minimize an augmented Lagrangian function subject to linearized constraints". Such methods converge rapidly near a solution but may not be reliable from arbitrary starting points. The well known software package MINOS has proven effective on many large problems. Its success motivates us to propose a variant of the LCL method that possesses three important properties: it is globally convergent, the subproblem constraints are always feasible, and the subproblems may be solved inexactly.

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 of the COPS, HS, and CUTE test problems, which include many large examples. The results demonstrate the robustness and efficiency of the stabilized LCL procedure.

Key words. large-scale optimization, nonlinear programming, nonlinear inequality constraints, augmented Lagrangian

AMS subject classifications. 49M37, 65K05, 90C30

1. Introduction.

For optimization problems with nonlinear constraints, *linearly constrained Lagrangian methods* (LCL methods) solve a sequence of subproblems that minimize an augmented Lagrangian function subject to linearizations of the problem constraints. (Typically some of the constraints are already linear, and some are simple bounds on the variables. They are included verbatim in the subproblems.) Existing LCL methods converge rapidly near a solution but sometimes may not converge from arbitrary starting points. (They might not be *globally convergent*.) Nevertheless, the well known software package MINOS [35] employs an LCL method and has proven effective on many problems (large and small), especially within the GAMS [8] and AMPL [19] environments. It is widely used in industry and academia. Its success motivates us to propose an LCL-like method for which global convergence to a local minimizer or a stationary point can be proved under standard assumptions.

Our globally convergent LCL algorithm, henceforth referred to as *stabilized* LCL (sLCL), solves a sequence of linearly constrained (LC) subproblems as just described. Each subproblem minimizes an augmented Lagrangian function within a linear manifold that describes a current approximation to the nonlinear constraints (including any linear constraints and bounds). 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.

*Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439-4844 (michael@mcs.anl.gov). This work was supported by the U.S. National Science Foundation grant CCR-9988205 and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy contract W-31-109-Eng-38.

†Department of Management Science and Engineering, Stanford University, Stanford, CA 94305-4026 (saunders@stanford.edu). This work was supported by the U.S. National Science Foundation grants CCR-9988205 and CCR-0306662, and the U.S. Office of Naval Research grants N00014-96-1-0274 and N00014-02-1-0076. Version: 1.60, April 6, 2004

In addition to global convergence, the sLCL method possesses two important properties: the subproblems are always feasible, and they may be solved inexactly. In the method used by MINOS, there is no guarantee that the subproblems will be feasible, and the criteria for their early termination is heuristic. Our method may be regarded as a generalization of sequential augmented Lagrangian methods (see, for example, [25, 1, 18]). The theory we develop provides a framework that unifies Robinson's LCL method [39] with the bound-constrained Lagrangian (BCL) method used, for example, by LANCELOT [11]. In the context of our theory, the proposed algorithm is actually a continuum of methods, with Robinson's LCL method and the BCL method at opposite ends of a spectrum. The sLCL method exploits this relationship, preserving the fast local convergence properties of LCL methods while inheriting the global convergence properties of BCL methods. This connection is explored in more detail by Friedlander [20].

Our intent is to develop a method that is effective in practice for large-scale problems, and is also based on sound theory. We implemented the sLCL method as a MATLAB program that calls either the reduced-gradient part of MINOS [34] or the sequential quadratic programming code SNOPT [24] 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. Following the theoretical development, we give computational results and a comparison with MINOS on a set of nontrivial problems.

1.1. The optimization problem. The proposed method solves nonlinearly constrained optimization problems of the form

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

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a linear or nonlinear objective function and $c : \mathbb{R}^n \mapsto \mathbb{R}^m$ is a vector of nonlinear constraint functions. We assume that the constraints of (GNP) are feasible. In section 5.6 we explain how the proposed algorithm reveals infeasible constraints and discuss properties of the points to which it converges. In section 7 we consider a formulation with more general constraints (for which our implementation and other solvers are designed).

One of the strengths of our method 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. Second derivatives could be used by the subproblem solver if they were available, thus accelerating the subproblem solution process and changing the properties of the solutions obtained for (GNP). We discuss this further in section 5.7.

1.2. The LCL approach. The acronyms LCL and BCL are new. Methods belonging to the LCL class have previously been referred to as sequential linearized constraint (SLC) methods (cf. [25, 36]). The term SLC was chosen for compatibility with the terms sequential quadratic programming (SQP) and sequential linear programming (SLP). Those methods also sequentially linearize the constraints. The term *linearly constrained Lagrangian* emphasizes that the Lagrangian itself (really the augmented Lagrangian) is used in the subproblems and not a linear or quadratic approximation. Moreover, there is a useful relationship (which we exploit) between LCL and BCL methods, and this is hinted at by the nomenclature.

The first LCL methods were proposed independently in 1972. Robinson [39] and Rosen and Kreuser [42] 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 (GNP). A practical strength is that efficient large-scale methods exist for the solution of the linearly constrained subproblems formed at each iteration. Any suitable subproblem solver may be called as a black box.

1.3. Other work on stabilizing LCL methods. Other approaches to stabilizing LCL algorithms include two-phase methods proposed by Rosen [41] and Van Der Hoek [43]. In these approaches, a Phase 1 problem is formed by moving the nonlinear constraints into the objective by means of 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. These two-phase methods choose the penalty parameter arbitrarily, however, and do not deal methodically with infeasible linearizations.

In 1981, Best et al. [2] 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 involve only those 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 is that it requires a fourth-order penalty term to ensure continuous second derivatives of the penalty objective. This requirement 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 all linearized constraints as part of each subproblem (one advantage being that it keeps the number of degrees of freedom small). We also allow the subproblems to predict the set of active constraints, as in Robinson's 1972 method. In particular, the active set for the final subproblem is taken to be the active set for the solution of the original problem.

1.4. Definitions. Unless otherwise specified, the function $\|v\|$ represents the Euclidean norm of a vector v. With vector arguments, the functions $\min(\cdot,\cdot)$ and $\max(\cdot,\cdot)$ are defined component-wise.

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

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

where x, the m-vector y, and the scalar ρ are independent variables. This function is particularly important for our analysis. When y_k and ρ_k are fixed, we often use the shorthand notation $\mathcal{L}_k(x) \equiv \mathcal{L}(x, y_k, \rho_k)$.

Let $g(x)$ denote the gradient of the objective function $f(x)$, and let $J(x)$ denote the Jacobian of $c(x)$: a matrix whose ith row is the gradient of $[c(x)]_i$, where $[\,\cdot\,]_i$ refers to the ith component of a vector. Let $H(x)$ and $H_i(x)$ be the Hessian matrices of $f(x)$ and $[c(x)]_i$ respectively, and define the *first-order multipler estimate*

$$\widetilde{y}(x, y, \rho) = y - \rho c(x). \tag{1.1}$$

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

$$\nabla_x \mathcal{L}(x, y, \rho) = g(x) - J(x)^T \widetilde{y}(x, y, \rho) \tag{1.2}$$

$$\nabla^2_{xx} \mathcal{L}(x, y, \rho) = H(x) - \sum_{i=1}^{m} [\widetilde{y}(x, y, \rho)]_i H_i(x) + \rho J(x)^T J(x). \tag{1.3}$$

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) conditions.

DEFINITION 1.1 (First-Order KKT Conditions). *A triple (x_*, y_*, z_*) is a first-order KKT point for* (GNP) *if for any $\rho \geq 0$ all of the following hold:*

$$c(x_*) = 0 \tag{1.4a}$$

$$\nabla_x \mathcal{L}(x_*, y_*, \rho) = z_* \tag{1.4b}$$

$$\min(x_*, z_*) = 0. \tag{1.4c}$$

The vector of Lagrange multipliers z_* corresponds to the bound constraints $x \geq 0$. Its elements are referred to as *reduced costs* in the linear programming literature.

KKT conditions are normally defined in terms of the Lagrangian function, not the augmented Lagrangian function used in (1.4b). We take the perspective that the Lagrangian is just a special case of the augmented Lagrangian (with $\rho = 0$). Our definition is not more restrictive: if (1.4) holds for $\rho = 0$ (the usual case) then it holds for all $\rho \geq 0$ because $c(x_*) = 0$.

Also, (1.4c) is shorthand for nonnegativity and complementarity conditions that are normally defined explicitly:

$$x_* \geq 0, \;\; z_* \geq 0 \qquad \text{and} \qquad [x_*]_j = 0 \quad \text{or} \quad [z_*]_j = 0. \tag{1.5}$$

Let $\eta_* > 0$ and $\omega_* > 0$ be specified as primal and dual convergence tolerances. We regard the point (x, y, z) as an acceptable approximate solution of (GNP) if it satisfies (1.4) to within these tolerances:

$$\|c(x)\| \leq \eta_* \tag{1.6a}$$

$$\nabla_x \mathcal{L}(x, y, \rho) = z \tag{1.6b}$$

$$\|\min(x, z)\|_\infty \leq \omega_*, \tag{1.6c}$$

where (1.6c) is shorthand for

$$x \geq -\omega_* e, \;\; z \geq -\omega_* e \qquad \text{and} \qquad [x]_j \leq \omega_* \quad \text{or} \quad [z]_j \leq \omega_*,$$

where e is a vector of ones. In practice, we might use a different tolerance for the nonnegativity of x, replacing the first inequality by $x \geq -\delta_* e$ for some $\delta_* > 0$.

The algorithms we discuss are structured around *major* and *minor* iterations. Each major iteration solves a linearly constrained subproblem to generate 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 iterates converging to $(x_k^*, \Delta y_k^*, z_k^*)$, a local solution of the current subproblem. In our development and analysis of a stabilized LCL method we are primarily concerned with the "outer"-level algorithm, comprising the major iterations. Unless stated otherwise, "iterations" refers to major iterations.

For the theoretical development of our method, it is convenient to use the normal augmented Lagrangian for the objective of the subproblems. The dual solutions of the

subproblems then converge to 0 (not to y_*) as $k \to \infty$. Hence our notation Δy_k^*. (As mentioned in section 9.2, a *modified* augmented Lagrangian is desirable in practice, and the dual solutions then converge to y_*.)

Let $\mathcal{I}$ be the index set $\mathcal{I} = \{j \in 1, \ldots, n \mid [x_*]_j > 0\}$ corresponding to inactive bounds at x_*. We define

$$\widehat{g}(x) \equiv [g(x)]_{\mathcal{I}} \quad \text{and} \quad \widehat{J}(x) \tag{1.7}$$

to be the elements of $g(x)$ and the columns of $J(x)$ corresponding to the indices in $\mathcal{I}$. Further notation follows, including some already introduced:

(x_*, y_*, z_*)	optimal primal and dual variables for (GNP)
(x_k, y_k, z_k)	the kth estimate of (x_*, y_*, z_*)
$(x_k^*, \Delta y_k^*, z_k^*)$	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_*
$\bar{c}_k(x)$	$c_k + J_k(x - x_k)$ the linearization of $c(x)$ at x_k
$\mathcal{L}_k(x)$	$\mathcal{L}(x, y_k, \rho_k)$, the augmented Lagrangian with fixed y_k and ρ_k
$\widetilde{y}(x, y, \rho)$	$y - \rho c(x)$, the first-order multiplier estimate in (1.1)–(1.3)
$\widehat{y}(x)$	the least-squares multiplier estimate (5.1).

The symbol x_* (used above) also denotes a limit point of the sequence $\{x_k^*\}$. For a single limit point, $x_k^* \to x_*$ for all k sufficiently large. When there are multiple limit points, we use $(x_*, \mathcal{K})$ to denote the limit point and the infinite set of indices associated with the corresponding convergent subsequence.

1.5. Assumptions. The first assumption holds implicitly throughout.

ASSUMPTION 1.2. *The functions f and c are twice continuously differentiable.*

The following assumptions are used in sections 5–6.

ASSUMPTION 1.3. *The sequence $\{x_k^*\}$ lies in a closed and bounded set $\mathcal{B} \subset \mathbb{R}^n$.* (This ensures at least one convergent subsequence and at least one limit point x_*.)

ASSUMPTION 1.4. *The matrix $\widehat{J}(x_*)$ has full row rank at every limit point x_* of the sequence $\{x_k^*\}$.* (This is commonly known as the *linear independence constraint qualification* (LICQ); see Mangasarian [30] or Nocedal and Wright [36]. For infeasible problems, section 5.6 does not need this assumption.)

2. The canonical LCL method. Algorithm 1 below outlines what we regard to be the canonical LCL method for solving problem (GNP). It solves a sequence of linearly constrained subproblems that are parameterized by the latest estimates (x_k, y_k) and a fixed penalty parameter $\rho_k \equiv \bar{\rho}$ (which may be zero):

(LC_k)	$\underset{x}{\text{minimize}} \quad \mathcal{L}_k(x)$
	subject to $\quad \bar{c}_k(x) = 0$
	$x \geq 0.$

A local solution of (LC_k) is a triple $(x_k^*, \Delta y_k^*, z_k^*)$ that satisfies the first-order KKT conditions

$$\bar{c}_k(x) = 0 \tag{2.1a}$$

$$\nabla \mathcal{L}_k(x) - J_k^T \Delta y = z \tag{2.1b}$$

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

Algorithm 1: Canonical LCL.

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

[**Initialize parameters**]
  Set the penalty parameter $\rho_0 = \bar{\rho} \geq 0$.
  Set positive convergence tolerances $\omega_*, \eta_* \ll 1$;
$k \leftarrow 0$;
converged $\leftarrow$ false;
repeat
  [**Solve the linearly constrained subproblem** (LC_k)]
    Find a point $(x_k^*, \Delta y_k^*, z_k^*)$ that satisfies (2.1);
1   If there is more than one such point, choose $(x_k^*, \Delta y_k^*, z_k^*)$ closest in norm to $(x_k, 0, z_k)$;
  [**Update solution estimates**]
    $x_{k+1} \leftarrow x_k^*, \;\; y_{k+1} \leftarrow y_k + \Delta y_k^*, \;\; z_{k+1} \leftarrow z_k^*$;
  [**Test convergence**]
    if $(x_{k+1}, y_{k+1}, z_{k+1})$ satisfies (1.6) **then** converged $\leftarrow$ true;
  $\rho_k \leftarrow \bar{\rho}$; [keep ρ_k fixed]
  $k \leftarrow k + 1$;
until converged;
$x_* \leftarrow x_k, \;\; y_* \leftarrow y_k, \;\; z_* \leftarrow z_k$;
return x_*, y_*, z_*;

The original LCL method of Robinson [39] is equivalent to Algorithm 1 with $\bar{\rho} = 0$. A positive penalty parameter was introduced in the software package MINOS [35] to improve convergence from difficult starting points. Empirically, it has proven a helpful addition to Robinson's method for many problems, but sometimes it is ineffective. A theoretical understanding of when and how to modify ρ_k has been lacking.

For any set of parameters x_k, y_k, and ρ_k there may be many points $(x_k^*, \Delta y_k^*, z_k^*)$ that satisfy (2.1). Step 1 of Algorithm 1 (the line labeled 1) requires that the chosen point be closest in norm to $(x_k, 0, z_k)$. This step is found in the original formulation of Robinson's method, where it is used in the local convergence proof [39, Theorem 2], and we will continue to use it in our formulation of the sLCL method. Such a requirement cannot be verified in practice, but it may be argued that, depending on the choice of subproblem solver, the requirement may often be satisfied. Note that the requirement is used only in the local analysis (section 6) of the sLCL method, and that the point $(x_k^*, \Delta y_k^*, z_k^*)$ is locally unique under the assumptions of that analysis. However, in general the point may not be unique.

3. An elastic LC subproblem. We recognize two particular causes of failure in the canonical LCL method:

- The linear constraints in subproblem (LC_k) may be infeasible for some x_k, so that x_{k+1} may not be well defined.
- Even if the linearized constraints are feasible, $\|x_k^* - x_k\|$ may be arbitrarily large regardless of the values of y_k and ρ_k in the subproblem objective.

To remedy both deficiencies we modify the linearized constraints, allowing some degree of flexibility in their satisfaction. We introduce nonnegative *elastic variables* v and w into the constraints, with a penalty on these variables in the subproblem

objective. The resulting elastic subproblem is *always feasible*:

$$\begin{array}{ll} (\mathrm{ELC}_k) & \underset{x,v,w}{\text{minimize}} \quad \mathcal{L}_k(x) + \sigma_k e^T(v+w) \\ & \text{subject to} \quad \bar{c}_k(x) + v - w = 0 \\ & \qquad\qquad\qquad\qquad x, v, w \geq 0. \end{array}$$

The solution of (ELC_k) is a 5-tuple $(x_k^*, v_k^*, w_k^*, \Delta y_k^*, z_k^*)$ that satisfies the first-order KKT conditions

$$\bar{c}_k(x) + v - w = 0 \tag{3.1a}$$
$$\nabla \mathcal{L}_k(x) - J_k^T \Delta y = z \tag{3.1b}$$
$$\min(x, z) = 0 \tag{3.1c}$$
$$\min(v, \sigma_k e - \Delta y) = 0 \tag{3.1d}$$
$$\min(w, \sigma_k e + \Delta y) = 0. \tag{3.1e}$$

The last two conditions (3.1d)–(3.1e) imply that their arguments are nonnegative, and so $\sigma_k e \geq \Delta y \geq -\sigma_k e$. This pair of inequalities can be conveniently restated as

$$\|\Delta y\|_\infty \leq \sigma_k. \tag{3.2}$$

The sLCL method is based on subproblem (ELC_k) with judicious changes to ρ_k and σ_k for each k. We find later that the bound (3.2)—relaxed slightly in (3.4) below—is crucial for the global convergence analysis of the method. In particular, setting σ_k small allows us to counteract the effect of a large $\|x_k^* - x_k\|$ (see (5.2) below).

3.1. The ℓ_1 penalty function. The term $\sigma_k e^T(v+w)$ is the ℓ_1 penalty function. Together with the constraints $v, w \geq 0$, it is equivalent to a penalty on $\|v - w\|_1$ (see [27]). Eliminating $v - w$, we see that the elastic subproblem (ELC_k) can be stated as

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

with solution (x_k^*, z_k^*). This immediately reveals the stabilized LCL method's intimate connection with the augmented Lagrangian function and BCL methods. Far from a solution, the ℓ_1 penalty term $\sigma_k \|\bar{c}_k(x)\|_1$ gives the method an opportunity to deviate from the constraint linearizations and reduce $\mathcal{L}_k(x)$ more than otherwise. Near a solution, it keeps the iterates close to the linearizations. Two extreme cases are of interest:

Recovering the BCL subproblem. Set $\sigma_k = 0$. Then (ELC_k) and (ELC'_k) reduce to the equivalent bound-constrained minimization problem

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

Subproblem (BC_k) is used by BCL methods (e.g., Hestenes [28], Powell [38], Bertsekas [1], Conn et al. [10]) and in particular by LANCELOT [11].

Recovering the LCL subproblem. The ℓ_1 penalty function is *exact*. If the linearization is feasible and σ_k is larger than a certain threshold, v and w are likely to be zero and the minimizers of the elastic problem (ELC_k) will coincide with the minimizers of the inelastic problem (LC_k). Exact penalty functions have been studied by [27, 1, 17] among others. See Conn et al. [13] for a more recent discussion.

We are particularly interested in this feature when the iterates generated by the sLCL algorithm are approaching a solution (x_*, y_*, z_*). Recovering the canonical LCL subproblem ensures that sLCL inherits the fast local convergence properties of the canonical LCL algorithm. In section 6.2 we give conditions under which the elastic variables will be forced to zero.

3.2. Early termination of the LC subproblems. Poor values of x_k, y_k, or ρ_k may imply subproblems whose accurate solutions are far from a solution of (GNP). We therefore terminate subproblems early by relaxing (3.1c)–(3.1e) by an amount ω_k. However, we enforce nonnegativity on x, v, w (also implied by (3.1c)–(3.1e)):

$$x, v, w \geq 0 \tag{3.3a}$$
$$\bar{c}_k(x) + v - w = 0 \tag{3.3b}$$
$$\nabla \mathcal{L}_k(x) - J_k^T \Delta y = z \tag{3.3c}$$
$$\|\min(x, z)\|_\infty \leq \omega_k \tag{3.3d}$$
$$\|\min(v, \sigma_k - \Delta y)\|_\infty \leq \omega_k \tag{3.3e}$$
$$\|\min(w, \sigma_k + \Delta y)\|_\infty \leq \omega_k. \tag{3.3f}$$

Note that (3.3e) and (3.3f) imply

$$\|\Delta y\|_\infty \leq \sigma_k + \omega_k. \tag{3.4}$$

As discussed in connection with (1.5), in practice (3.3a) is typically relaxed by a fixed tolerance δ by instead requiring that $x, v, w \geq -\delta e$.

3.3. Relation to Sℓ_1QP and SNOPT. Subproblems (ELC_k) and (ELC'_k) are reminiscent of the QP subproblems arising in the Sℓ_1QP method of Fletcher [17] and the SQP method of SNOPT [24]. The main difference is that the QP objectives are *quadratic approximations to the Lagrangian* (not the Lagrangian itself, and not the augmented Lagrangian).

In Sℓ_1QP, trust-region bounds are used to ensure boundedness of $\|x_k^* - x_k\|_\infty$ and to permit the subproblem solution x_k^* to reduce the exact penalty function $\nu f(x) + \|c(x)\|_1$ for some (small enough) scalar ν. Also, second-order corrections are needed to maintain rapid convergence of the major iterations (avoiding the Maratos effect [31]).

In SNOPT, the QP subproblems are used to define *search directions* $(\Delta x, \Delta y)$ for the augmented Lagrangian $\mathcal{L}(x, y, \rho_k)$ regarded as a function of $x = x_k + \alpha \Delta x$ and $y = y_k + \alpha \Delta y$ for some steplength $\alpha \in (0, 1]$. The elastic slacks v and w are initially absent but become a permanent part of the original problem (and the QP subproblems) if the linearized constraints are infeasible at some x_k, or if $\|y_k\|$ becomes large. The net effect is that for problems whose constraints are infeasible, Sℓ_1QP and SNOPT minimize the 1-norm of $c(x)$, whereas sLCL minimizes the 2-norm (see section 5.6).

In all three methods (Sℓ_1QP, SNOPT, and sLCL), the penalty term $\sigma_k e^T(v + w)$ may be nonzero at a particular subproblem solution, indicating that some of the

linearized constraints are not satisfied. Thus, the active linearized constraints may be just a subset of the full set (cf. Fletcher [17, p. 32]). The global convergence properties of sLCL do not require independent constraint gradients or bounded multipliers for every subproblem. (These are required only at limit points of the sequence generated by the method.)

4. The stabilized LCL algorithm. Algorithm 2 outlines the sLCL algorithm. Its structure closely parallels the BCL algorithm described in Conn et al. [10]. 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 by using information from the current subproblem solution. Otherwise, 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. To prevent the linearized constraints from interfering with the penalty parameter's ability to reduce the primal infeasibility, the algorithm may reduce the elastic penalty parameter σ_k. (A small value of σ_k encourages deviation from the current constraint linearizations, which may be poor approximations to the true constraints.)

The parameter $\bar{\sigma}$ ensures that $\|\Delta y_k^*\|$ remains uniformly bounded. In particular, $\rho_k > 0$ and $\tau_\sigma > 1$ in Steps 7 and 11 ensure that σ_k is uniformly bounded, and Step 1 ensures that ω_k is uniformly bounded; therefore $\|\Delta y_k^*\|$ is uniformly bounded by virtue of (3.4). The division by $1 + \rho_k$ is a safeguard if the algorithm continues to oscillate between successful and unsuccessful iterations; in that event, $\rho_k \to \infty$ and $\sigma_k \to 0$, and the algorithm becomes increasingly similar to a quadratic penalty function method.

The two salient features of this algorithm are that it is globally convergent and asymptotically equivalent to the canonical LCL method. In section 5 we demonstrate the global convergence properties of the algorithm by proving results analogous to Lemma 4.3 and Theorem 4.4 in [10]. In section 6 we demonstrate that the algorithm eventually reduces to the canonical LCL method and hence inherits that method's asymptotic convergence properties.

5. Global convergence. At all points x for which $\widehat{J}(x)$ has full row rank we define the *least-squares multiplier estimate* $\widehat{y}(x)$ as the solution of the following least-squares problem:

$$\widehat{y}(x) \equiv \arg\min_y \|\widehat{g}(x) - \widehat{J}(x)^T y\|^2. \tag{5.1}$$

Note that the definitions of $\widehat{g}$, $\widehat{J}$, and hence $\widehat{y}$ require a priori knowledge of the bounds active at x_*. We emphasize that $\widehat{y}$ is used only as an analytical device and its computation is never required. Assumption 1.4 guarantees the uniqueness of $\widehat{y}$ at every limit point of the sequence $\{x_k^*\}$.

For the global analysis we assume that $\omega_* = \eta_* = 0$.

5.1. Preliminaries. We need the following lemma to bound the errors in the least-squares multiplier estimates relative to $\|x_k^* - x_*\|$, the error in x_k^*. The lemma follows from Lemmas 2.1 and 4.4 of Conn et al. [9]. It simply demonstrates that $\widehat{y}(x)$ is Lipschitz continuous in a neighborhood of x_*.

LEMMA 5.1. *Suppose Assumption* 1.4 *holds, and let* $(x_*, \mathcal{K})$ *be a limit point of* $\{x_k^*\}$. *Then there exists a positive constant* α *such that* $\|\widehat{y}(x_k^*) - \widehat{y}(x_*)\| \le \alpha \|x_k^* - x_*\|$ *for all* $k \in \mathcal{K}$ *sufficiently large.*

Algorithm 2: Stabilized LCL.

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

[**Initialize parameters**]
Set $\overline{\sigma} \gg 1$. Set constants $\tau_\rho, \tau_\sigma > 1$. Set initial penalty parameters $\rho_0 > 0$ and $1 \le \sigma_0 \le \overline{\sigma}$. Set positive convergence tolerances $\omega_*, \eta_* \ll 1$ and initial infeasibility tolerance $\eta_0 > \eta_*$. Set constants $\alpha, \beta > 0$ with $\alpha < 1$;
$k \leftarrow 0$;
converged $\leftarrow$ false;
repeat
1 Choose $\omega_k > 0$ such that $\lim_{k\to\infty} \omega_k \le \omega_*$;
2 [**Solve the linearly constrained subproblem** (ELC$_k$)]
Find a point $(x_k^*, v_k^*, w_k^*, \Delta y_k^*, z_k^*)$ that satisfies (3.3). If there is more than one such point, choose one such that $(x_k^*, \Delta y_k^*, z_k^*)$ is closest in norm to $(x_k, 0, z_k)$;
3 **if** $\|c(x_k^*)\| \le \max(\eta_*, \eta_k)$ **then**
4 [**Update solution estimates**]
$x_{k+1} \leftarrow x_k^*$;
5 $y_{k+1} \leftarrow y_k + \Delta y_k^* - \rho_k c(x_k^*)$; [$\equiv \widetilde{y}(x_k^*, y_k + \Delta y_k^*, \rho_k)$]
6 $z_{k+1} \leftarrow z_k^*$;
[**Keep penalty parameter and update elastic weight**]
$\rho_{k+1} \leftarrow \rho_k$; [keep ρ_k]
7 $\sigma_{k+1} \leftarrow \frac{1}{1+\rho_k} \min(1 + \|\Delta y_k^*\|_\infty, \overline{\sigma})$; [reset σ_k]
8 [**Test convergence**]
if $(x_{k+1}, y_{k+1}, z_{k+1})$ satisfies (1.6) **then** converged $\leftarrow$ true;
9 $\eta_{k+1} \leftarrow \eta_k/(1 + \rho_{k+1}^\beta)$; [decrease η_k]
else
10 [**Keep solution estimates**]
$x_{k+1} \leftarrow x_k$; $y_{k+1} \leftarrow y_k$; $z_{k+1} \leftarrow z_k$;
11 [**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]
12 $\eta_{k+1} \leftarrow \eta_0/(1 + \rho_{k+1}^\alpha)$; [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_*;

To prove the global convergence properties of Algorithm 2, we first describe the properties of any limit point that the algorithm 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 [9].

LEMMA 5.2. *Suppose Assumptions* 1.3 *and* 1.4 *hold. 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 in* $\mathcal{B}$ *and* $\{y_k\}$ *be any sequence of* m*-vectors. Let* $\{(x_k^*, \Delta y_k^*, z_k^*)\}$ *be a sequence of vectors satisfying* (3.3a), (3.3c), *and* (3.3d). *Let* $(x_*, \mathcal{K})$ *be a limit point of the sequence* $\{x_k^*\}$. *Set* $\widetilde{y}_k = \widetilde{y}(x_k^*, y_k + \Delta y_k^*, \rho_k)$ *and* $y_* = \widehat{y}(x_*)$. *The following properties then hold:*

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

$$\|\widetilde{y}_k - y_*\| \leq \beta_1 \equiv \alpha_1\omega_k + M\|x_k^* - x_k\|\,\|\Delta y_k^*\| + \alpha_2\|x_k^* - x_*\|, \tag{5.2}$$
$$\rho_k\|c(x_k^*)\| \leq \beta_2 \equiv \beta_1 + \|\Delta y_k^*\| + \|y_k - y_*\|, \tag{5.3}$$

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

2. *If* $\|\Delta y_k^*\| \to 0$ *as* $k \in \mathcal{K} \to \infty$, *or* $\|\Delta y_k^*\|$ *is bounded above and* $\|x_k^* - x_k\| \to 0$, *then* $\widetilde{y}_k \to y_*$ *and* $z_k^* \to z_* \stackrel{\text{def}}{=} \nabla_x\mathcal{L}(x_*, y_*, 0)$.
3. *If* $c_* = 0$ *also, then* (x_*, y_*, z_*) *is a first-order KKT point for* (GNP).

Proof. From Assumption 1.4, $\widehat{J}(x_k^*)$ has full row rank for all $k \in \mathcal{K}$ large enough. The least-squares multiplier estimate $\widehat{y}(x_k^*)$ (5.1) therefore exists. For full-rank least-squares problems in general, it is straightforward to show that the error in an approximate solution is bounded by a finite multiple of the associated residual. Hence with $x = x_k^*$ in problem (5.1) and with $y = \widetilde{y}_k$ as an approximate solution, we may write

$$\|\widehat{y}(x_k^*) - \widetilde{y}_k\| \leq \frac{\alpha_1}{\sqrt{n}}\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T\widetilde{y}_k\| \tag{5.4}$$

for some positive scalar α_1.

We now show that $\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T\widetilde{y}_k\|$ is bounded. By hypothesis, $(x_k^*, \Delta y_k^*, z_k^*)$ satisfies (3.3c). Together with (1.2),

$$\begin{aligned} z_k^* &= \nabla\mathcal{L}_k(x_k^*) - J_k^T\Delta y_k^* \\ &= g(x_k^*) - J(x_k^*)^T(y_k - \rho_k c(x_k^*)) - J_k^T\Delta y_k^* \\ &= g(x_k^*) - J(x_k^*)^T(y_k + \Delta y_k^* - \rho_k c(x_k^*)) + (J(x_k^*) - J_k)^T\Delta y_k^* \\ &= g(x_k^*) - J(x_k^*)^T\widetilde{y}_k + (J(x_k^*) - J_k)^T\Delta y_k^*, \end{aligned} \tag{5.5}$$

where $\widetilde{y}_k \stackrel{\text{def}}{=} \widetilde{y}(x_k^*, y_k + \Delta y_k^*, \rho_k) = y_k + \Delta y_k^* - \rho_k c(x_k^*)$. For $k \in \mathcal{K}$ large enough, x_k^* is sufficiently close to x_* so that $[x_k^*]_j > 0$ if $[x_*]_j > 0$. Therefore, (3.3d) and $\omega_k \to 0$ imply that $\min([x_k^*]_j, [z_k^*]_j) = [z_k^*]_j$, so that

$$\|[z_k^*]_{\mathcal{I}}\| \leq \|\min(x_k^*, z_k^*)\|, \tag{5.6}$$

where $\mathcal{I}$ is the index set of inactive bounds at x_* as defined in section 1.4. Because x_k^* and z_k^* both satisfy (3.3d), (5.6) implies that

$$\|[z_k^*]_{\mathcal{I}}\| \leq \sqrt{n}\;\omega_k. \tag{5.7}$$

Combining (5.5) and (5.7), we obtain

$$\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T\widetilde{y}_k + (\widehat{J}(x_k^*) - \widehat{J}_k)^T\Delta y_k^*\| \leq \sqrt{n}\;\omega_k$$

(see (1.7)). Also, because c is twice-continuously differentiable, J is Lipschitz continuous over $\mathcal{B}$ and there exists a positive constant M such that $\|\widehat{J}(x_k^*) - \widehat{J}_k\| \leq M\frac{\sqrt{n}}{\alpha_1}\|x_k^* - x_k\|$. Hence, from the triangle and Cauchy-Schwartz inequalities, we have

$$\begin{aligned} \|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T\widetilde{y}_k\| &\leq \|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T\widetilde{y}_k + (\widehat{J}(x_k^*) - \widehat{J}_k)^T\Delta y_k^*\| \\ &\qquad + \|\widehat{J}(x_k^*) - \widehat{J}_k\|\,\|\Delta y_k^*\| \\ &\leq \sqrt{n}\;\omega_k + M\frac{\sqrt{n}}{\alpha_1}\|x_k^* - x_k\|\,\|\Delta y_k^*\|, \end{aligned} \tag{5.8}$$

and so we have bounded the left-hand side as required.

We now derive (5.2). From the triangle inequality,

$$\|\widetilde{y}_k - y_*\| \le \|\widehat{y}(x_k^*) - \widetilde{y}_k\| + \|\widehat{y}(x_k^*) - y_*\|. \tag{5.9}$$

Using inequality (5.8) in (5.4), we deduce that

$$\|\widehat{y}(x_k^*) - \widetilde{y}_k\| \le \alpha_1\omega_k + M\|x_k^* - x_k\|\|\Delta y_k^*\|, \tag{5.10}$$

and Lemma 5.1 implies that there exists a constant α_2 such that

$$\|\widehat{y}(x_k^*) - y_*\| \le \alpha_2\|x_k^* - x_*\|, \tag{5.11}$$

for all $k \in \mathcal{K}$ large enough (recall that $y_* \equiv \widehat{y}(x_*)$). Substituting (5.10) and (5.11) into (5.9), we obtain $\|\widetilde{y}_k - y_*\| \le \beta_1$ as stated in (5.2).

We now prove (5.3). From the definition of $\widetilde{y}_k$, rearranging terms yields

$$\rho_k c(x_k^*) = y_k + \Delta y_k^* - \widetilde{y}_k. \tag{5.12}$$

Taking norms of both sides of (5.12) and using (5.2) yields

$$\begin{aligned}
\rho_k\|c(x_k^*)\| = \|y_k + \Delta y_k^* - \widetilde{y}_k\| &= \|y_k - y_* + y_* - \widetilde{y}_k + \Delta y_k^*\| \\
&\le \|\widetilde{y}_k - y_*\| + \|y_k - y_*\| + \|\Delta y_k^*\| \\
&\le \beta_1 + \|y_k - y_*\| + \|\Delta y_k^*\| \\
&\equiv \beta_2,
\end{aligned}$$

and so Part 1 of Lemma 5.2 is proved.

Now suppose $\|\Delta y_k^*\| \to 0$ 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. From (5.2) and the fact that $x_k^* \to x_*$ and $\omega_k \to 0$, we conclude that $\widetilde{y}_k \to y_*$ as $k \in \mathcal{K}$ goes to infinity. We also conclude from the continuity of J on the compact set $\mathcal{B}$ that $\|J(x_k^*) - J_k\|$ is bounded, so that

$$\lim_{k\in\mathcal{K}} \|(J(x_k^*) - J_k)^T\Delta y_k^*\| = 0. \tag{5.13}$$

On the other hand, suppose $\|\Delta y_k^*\|$ is uniformly bounded and $\lim_{k\in\mathcal{K}} \|x_k^* - x_k\| = 0$. We then conclude from (5.2) that $\widetilde{y}_k \to y_*$ as $k \in \mathcal{K}$ goes to infinity and again (5.13) holds. Because $\lim_{k\in\mathcal{K}}(x_k^*, \widetilde{y}_k) = (x_*, y_*)$, we have

$$\lim_{k\in\mathcal{K}} g(x_k^*) - J(x_k^*)^T\widetilde{y}_k = g_* - J_*^T y_*,$$

and so (5.5) and (5.13) together imply that $\lim_{k\in\mathcal{K}} z_k^* = z_* \equiv \nabla_x\mathcal{L}(x_*, y_*, 0)$. Thus we have proved Part 2 of Lemma 5.2.

Now note that each (x_k^*, z_k^*) satisfies (3.3d), so that $\lim_{k\in\mathcal{K}}(x_k^*, z_k^*) = (x_*, z_*)$ and $\omega_k \to 0$ together imply $\min(x_*, z_*) = 0$. Hence if $c_* = 0$ also, (x_*, y_*, z_*) satisfies (1.4) and so is a first-order KKT point for (GNP). Part 3 is thus proved. □

The conclusions of Lemma 5.2 pertain to any sequence $\{(x_k^*, \Delta y_k^*, z_k^*)\}$ satisfying the approximate first-order conditions (3.3). Algorithm 2 generates such a sequence and also generates auxiliary sequences of scalars $\{\omega_k\}$ and $\{\rho_k\}$ in such a way as to guarantee that the hypotheses of Lemma 5.2 hold. We demonstrate in Theorem 5.4 below that the condition of Part 3 of Lemma 5.2 holds. Therefore, every limit point of $\{(x_k^*, \widetilde{y}_k, z_k^*)\}$ is a first-order KKT point for (GNP).

5.2. Convergence of $\|y_k\|/\rho_k$. Before stating and proving the global convergence properties of the sLCL algorithm, we need to show that if $\rho_k \to \infty$ then the quotient $\|y_k\|/\rho_k$ converges to 0. This property is required (and used by Conn et al. [9, 10]) in lieu of assuming that $\|y_k\|$ remains bounded.

LEMMA 5.3. *Suppose $\rho_k \to \infty$ as k increases when Algorithm 2 is executed. Then $\|y_k\|/\rho_k \to 0$.*

Proof. Define the sequence $\{k_1, k_2, k_3, \ldots\}$ as the set of iterations that execute Step 11 of Algorithm 2, and let $\{l_1, l_2, l_3, \ldots\}$ be the associated sequence of iterations for which Step 5 was *last* executed. With these definitions,

$$y_{k_i} = \widetilde{y}_{l_i} \equiv \widetilde{y}(x^*_{l_i}, y_{l_i} + \Delta y^*_{l_i}, \rho_{l_i}) \qquad \text{and} \qquad \rho_{k_i} = \tau_\rho^{k_i - l_i}\rho_{l_i}. \tag{5.14}$$

Because the parameter ρ_k increases if and only if Step 11 is executed, it is sufficient to show that $\|y_{k_i}\|/\rho_{k_i} \to 0$ as $i \to \infty$.

Suppose there exists a maximum $\bar{l}$ over $\{l_1, l_2, l_3, \ldots\}$ (set $\bar{l} = 0$ if the set is empty). Then Step 4 is executed only finitely many (or zero) times and $y_k \equiv y_{\bar{l}}$ for all k large enough. Therefore, $\|y_k\|/\rho_k \to 0$, as required. Assume for the remainder that $\{l_1, l_2, l_3, \ldots\}$ is not empty and $l_i \to \infty$ as $i \to \infty$, implying that Step 4 is executed infinitely many times.

The multiplier update used in Step 4, together with the triangle inequality, (5.14), (3.4), and the fact that $\tau_\rho^{(k_i - l_i)} \geq 1$, imply that for each k_i,

$$\begin{aligned} \frac{\|y_{k_i}\|}{\rho_{k_i}} = \frac{\|\widetilde{y}_{l_i}\|}{\tau_\rho^{(k_i-l_i)}\rho_{l_i}} &\leq \frac{\|\widetilde{y}_{l_i}\|}{\rho_{l_i}} = \frac{\|y_{l_i} + \Delta y^*_{l_i} - \rho_{l_i} c(x^*_{l_i})\|}{\rho_{l_i}} \\ &\leq \underbrace{\frac{\|\Delta y^*_{l_i}\|}{\rho_{l_i}}}_{(a)} + \underbrace{\frac{\|y_{l_i} - \rho_{l_i} c(x^*_{l_i})\|}{\rho_{l_i}}}_{(b)}. \end{aligned} \tag{5.15}$$

As discussed in section 4, $\|\Delta y^*_k\|$ is uniformly bounded, so that (a) goes to zero as $\rho_{l_i} \to \infty$. The construction of the forcing sequence η_k (Steps 9 and 12) satisfies the requirements of [10, Lemma 4.2], where it is proven that term (b) goes to zero as $\rho_{l_i} \to \infty$. Therefore, (5.15) implies that $\|y_{k_i}\|/\rho_{k_i} \to 0$. ∎

5.3. Convergence of LC subproblem solutions. With Lemmas 5.2 and 5.3 in hand, we are now able to prove global convergence of the sLCL method under the assumption that *only a single limit point exists.* At the end of this section we discuss a modification of Algorithm 2 that guarantees global convergence even when multiple limit points exist.

THEOREM 5.4 (Global convergence of subproblem solutions). *Suppose Assumptions 1.3 and 1.4 hold. Let $\{(x^*_k, \Delta y^*_k, z^*_k)\}$ be the sequence of vectors generated by Algorithm 2 with tolerances $\omega_* = 0$ and $\eta_* = 0$. Let x_* be the single limit point of $\{x^*_k\}$. Then all parts of Lemma 5.2 hold. In other words, (x_*, y_*, z_*) as defined in Lemma 5.2 is a first-order KKT point.*

Proof. Algorithm 2 generates positive scalars ρ_k and, by Step 1, generates positive scalars $\omega_k \to 0$. Step 2 of the algorithm generates a sequence $\{(x^*_k, \Delta y^*_k, z^*_k)\}$ that satisfies (3.3) for each k. Therefore, the hypotheses of Lemma 5.2 hold, and Part 1 of the lemma follows immediately.

Note that each x^*_k satisfies (3.3a), and so all $x^*_k \geq 0$. Thus, $x_* \geq 0$. Moreover, σ_k is uniformly bounded. We then need to consider the four possible cases:

1. ρ_k is uniformly bounded, and $\sigma_k \to 0$;

2. ρ_k is uniformly bounded, and $\sigma_k \nrightarrow 0$;
3. $\rho_k \to \infty$ and $\sigma_k \to 0$;
4. $\rho_k \to \infty$ and $\sigma_k \nrightarrow 0$.

We dismiss Cases 1 and 4 because they cannot be generated by the algorithm. (As k gets large, $\sigma_k \to 0$ only if Step 11 is executed infinitely many times, contradicting the finiteness of ρ_k. Conversely, if $\rho_k \to \infty$, then Steps 7 and 11 ensure that $\sigma_k \to 0$.)

Case 2 implies that Step 9 of Algorithm 2 is executed for all k large enough. Thus, $x_{k+1} = x_k^*$ for all large k, and hence $x_k^* \to x_*$ implies $x_{k+1} \to x_*$. The "single limit point" assumption implies $x_k \to x_*$. Therefore, $\|x_k^* - x_k\| \to 0$. Because $\|\Delta y_k^*\|$ is bounded (see section 4), Part 2 of Lemma 5.2 holds. In addition, $\eta_k \to 0$ because $\rho_k > 0$ and Step 9 is executed for all k large enough. Therefore, the condition $\|c(x_k^*)\| \le \eta_k$ for all k large enough implies that $c(x_k^*) \to 0$. By continuity of c, $c_* = 0$. Thus, Part 3 of Lemma 5.2 holds.

Now consider Case 3. Because $\sigma_k \to 0$ and $\omega_k \to 0$, (3.4) implies that $\|\Delta y_k^*\| \to 0$ as k increases. Then Part 2 of the lemma holds. To show that $c(x_k^*) \to 0$, divide both sides of (5.3) by ρ_k to obtain

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

Term (a) clearly goes to zero as ρ_k increases. Because Δy_k^* satisfies (3.4), and because x_k^* and x_k belong to the compact set $\mathcal{B}$, (b) and (c) go to zero as ρ_k increases. By Lemma 5.3, $\|y_k\|/\rho_k \to 0$, and so (d) goes to 0. We conclude that $\|c(x_k^*)\| \to 0$ as k increases, as required. □

Note that the "single limit point" assumption in Theorem 5.4 is needed only in Case 2 to ensure that $\|x_k^* - x_k\| \to 0$, as required by Part 2 of Lemma 5.2. (If multiple limit points exist, we can only ensure that $\|x_k^* - x_{k+1}\| \to 0$.) To allow for multiple limit points, Step 7 of Algorithm 2 could be replaced by the following update:

7a **if** $\|\Delta y_k^*\|_\infty \le \delta(\frac{1}{2})^j$ **then**
  $\sigma_{k+1} \leftarrow \frac{1}{1+\rho_k}\min(1 + \|\Delta y_k^*\|_\infty, \overline{\sigma})$; [reset σ_k]
else
7b $\sigma_{k+1} \leftarrow \sigma_k/\tau_\sigma$; [decrease σ_k]

(The counter j is the number of consecutive successful major iterations, and δ is a positive parameter.) We now consider Case 2 under the alternative update. If ρ_k is uniformly bounded, then all iterations are successful for k large enough, and $j \to \infty$. Then if Δy_k^* does not satisfy Step 7a infinitely often, $\sigma_k \to 0$, and (3.4) implies that $\|\Delta y_k^*\| \to 0$. Then Part 2 of Lemma 5.2 holds, as required.

Any forcing sequence converging to zero could be used in Step 7a. However, requiring only a gentle decrease in $\|\Delta y_k^*\|_\infty$ at each iteration should interfere less with the fast local convergence of the sLCL method, because $\|\Delta y_k^*\|$ may be expected to decrease at a linear rate, as noted in Theorem 6.2, equation (6.6).

5.4. Termination in the limit. Note that the convergence test takes place only if Step 3 of Algorithm 2 tests true; i.e., if $\|c(x_k^*)\| \le \eta_k$ (because $\eta_* = 0$). To guarantee that the algorithm will eventually terminate as the iterates x_k, y_k, and z_k converge, we need to guarantee that Step 8 executes 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 to evaluate true (except in rare occasions when $c(x_k^*) = 0$). The forcing sequence defined by Steps 9 and 12 of Algorithm 2 is suggested by Conn et al. [9, 10]. The following corollaries show that this forcing sequence has the desired property and summarize the global convergence properties of Algorithm 2.

COROLLARY 5.5 (Global convergence with a single limit point). *Let* $\{(x_k, y_k, z_k)\}$ *be the sequence of vectors generated by Algorithm* 2. *Let* x_* *be the single limit point of* $\{x_k^*\}$. *Suppose Assumptions* 1.3 *and* 1.4 *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. Let $\{(x_k^*, \Delta y_k^*, z_k^*)\}$ be the sequence of vectors generated by Step 2 of Algorithm 2 and set $\widetilde{y}_k = \widetilde{y}(x_k^*, y_k + \Delta y_k^*, \rho_k)$. By Lemma 5.2 and Theorem 5.4,

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

Moreover, (x_*, y_*, z_*) is a first-order KKT point for (GNP). Suppose Step 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^*, $\widetilde{y}_k$, and z_k^*.

We now show by contradiction that Step 4 does occur infinitely often. Suppose instead that it does not. Then there exists a k_1 large enough so that Steps 10 and 11 are executed for all $k > k_1$. Consider only iterations $k > k_1$. Then $y_k \equiv \bar{y}$ and $\rho_k \to \infty$. As in (5.12), the definition of $\widetilde{y}_k$ gives

$$\begin{aligned} \rho_k \|c(x_k^*)\| &= \|\bar{y} + \Delta y_k^* - \widetilde{y}_k\| \\ &\le \|\Delta y_k^*\| + \|\bar{y}\| + \|\widetilde{y}_k\|. \end{aligned} \tag{5.16}$$

Each Δy_k^* satisfies $\|\Delta y_k^*\|_\infty \le \sigma_k + \omega_k$ (3.4) with $\sigma_k + \omega_k$ uniformly bounded. Moreover, $\lim_{k\to\infty} \widetilde{y}_k = y_*$ and y_* is bounded. (Assumption 1.4 ensures that the least-squares solution of (5.1) exists and is unique.) Then from (5.16) there exists some constant $L > 0$, independent of k, such that

$$\rho_k \|c(x_k^*)\| \le L \tag{5.17}$$

for all k. But the test at Step 3 fails at every iteration, so that

$$\eta_k < \|c(x_k^*)\|. \tag{5.18}$$

Combining (5.17) and (5.18), we find that

$$\rho_k \eta_k < \rho_k \|c(x_k^*)\| \le L. \tag{5.19}$$

From Step 12, $\eta_{k+1} = \eta_0 / \rho_{k+1}^\alpha$, so

$$\rho_k \eta_k = \rho_k \frac{\eta_0}{\rho_k^\alpha} = \eta_0 \rho_k^{1-\alpha}. \tag{5.20}$$

Substituting (5.20) into (5.19), 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 4 must occur infinitely often. □

5.5. Finite termination. The following result simply asserts that Algorithm 2 will eventually exit when ω_* and η_* are positive, as they are in practice.

COROLLARY 5.6 (Finite Termination). *Suppose the convergence tolerances ω_* and η_* are strictly positive. Then, under the conditions of Corollary* 5.5*, Algorithm* 2 *terminates after a finite number of iterations.*

Proof. Let $\{(x_k^*, \Delta y_k^*, z_k^*)\}$ and x_* be as in Theorem 5.4. Set $\widetilde{y}_k = \widetilde{y}(x_k^*, y_k + \Delta y_k^*, \rho_k)$. (This is y_{k+1} is Step 5.) By Theorem 5.4,

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

and (x_*, y_*, z_*) is a first-order KKT point for (GNP). By continuity, $\lim_{k\to\infty} \|c(x_k^*)\| = c_* = 0$, and hence $\|c(x_k^*)\| < \eta_* \leq \max(\eta_k, \eta_*)$ for all k large enough. Consequently, Step 4 is executed infinitely often and

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

Therefore, (x_k, y_k, z_k) satisfies conditions (1.6) for some k large enough. □

5.6. Infeasible problems. Not all optimization problems are well defined. 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 a system $c(x) = 0$, $x \geq 0$, is equivalent to verifying that the *global* minimizer of

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|c(x)\|^2 \\ \text{subject to} & x \geq 0 \end{array} \tag{5.21}$$

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

We analyze the properties of the sLCL algorithm when it is applied to an infeasible problem with convergence tolerances $\omega_* = \eta_* = 0$. We show that Algorithm 2 converges to a point that satisfies the first-order optimality conditions of the minimum-norm problem (5.21).

THEOREM 5.7 (Convergence for infeasible problems). *Suppose Assumption* 1.3 *holds. Let $(x_*, \mathcal{K})$ be a limit point of the sequence $\{x_k^*\}$ generated by Algorithm* 2*. Suppose* (GNP) *is infeasible. Then*

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

and (x_*, z_*) is a first-order KKT point for (5.21).

Proof. The pair (x_*, z_*) satisfies the first-order KKT conditions of (5.21) if

$$J_*^T c_* = z_* \qquad \text{and} \qquad \min(x_*, z_*) = 0. \tag{5.22}$$

Because (GNP) is infeasible, there exists a constant $\delta > 0$ such that $\delta < \|c(x)\|$ for all $x \geq 0$. Moreover, Steps 9 and 12 of Algorithm 2 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 11 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 $\bar{y}$,

$$x_k \equiv \bar{x} \quad \text{and} \quad y_k \equiv \bar{y}. \tag{5.23}$$

Note that Algorithm 2 generates x_k^* satisfying (3.3). Therefore, $x_k^* \geq 0$ for all k, and so $\lim_{k\in\mathcal{K}} x_k^* = x_*$ implies

$$x_* \geq 0. \tag{5.24}$$

As in (5.5), and using (3.3d) and (5.23), we have

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

or, after rearranging terms,

$$\underbrace{g(x_k^*) - J(x_k^*)^T\bar{y}}_{(a)} - \underbrace{J(\bar{x})^T\Delta y_k^*}_{(b)} + \rho_k J(x_k^*)^T c(x_k^*) \geq -\omega_k e. \tag{5.25}$$

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 $\bar{x}$ is constant, and from (3.4) we have $\|\Delta y_k^*\|_\infty \leq \sigma_k + \omega_k$. Then, because $\omega_k \to 0$ and $\rho_k \to \infty$, (5.25) implies that $J(x_k^*)^T c(x_k^*) \geq 0$ for all k large enough. Otherwise, (5.25) would eventually be violated as ρ_k grew large. Then,

$$z_* \stackrel{\text{def}}{=} \lim_{k\in\mathcal{K}} J(x_k^*)^T c(x_k^*) = J_*^T c_* \geq 0. \tag{5.26}$$

Because all x_k^* lie in a compact set, there exists some constant $L > 0$ such that

$$\|x_k^* - \bar{x}\| \leq \frac{L\alpha_1}{\sqrt{n}M}, \tag{5.27}$$

where M and α_1 are as defined in Lemma 5.2. Expression (5.8) can be derived again under the assumptions of this theorem, and does not require Assumption 1.4. Substituting (5.27) into (5.8) and using (3.4) and $\widetilde{y}_k \equiv y_k + \Delta y_k^* - \rho_k c(x_k^*)$, we have

$$\|\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T(\bar{y} + \Delta y_k^*) + \rho_k \widehat{J}(x_k^*)^T c(x_k^*)\| \leq \sqrt{n}\{\omega_k + \widehat{L}(\sigma_k + \omega_k)\}, \tag{5.28}$$

where $\widehat{L} > 0$ accounts for the ∞-norm in (3.4). Dividing (5.28) through by ρ_k, we obtain

$$\left\| \frac{1}{\rho_k}\big(\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T(\bar{y} + \Delta y_k^*)\big) + \widehat{J}(x_k^*)^T c(x_k^*) \right\| \leq \frac{\sqrt{n}\{\omega_k + \widehat{L}(\sigma_k + \omega_k)\}}{\rho_k}. \tag{5.29}$$

The quantity $\widehat{g}(x_k^*) - \widehat{J}(x_k^*)^T(y_k + \Delta y_k^*)$ is bounded for the same reasons that (a) and (b) above are bounded. Taking limits on both sides of (5.29), we see that $\rho_k \to \infty$ and $\omega_k, \sigma_k \to 0$ imply $\widehat{J}(x_k^*)^T c(x_k^*) \to 0$. By continuity of J and c, $\widehat{J}_*^T c_* = 0$. Equivalently, we may write

$$[J_*^T c_*]_j = 0 \quad \text{if} \quad [x_*]_j > 0, \quad j = 1, \ldots, n. \tag{5.30}$$

Together, (5.24), (5.26) and (5.30) imply that (x_*, z_*) satisfies conditions (5.22), as required. □

Theorem 5.7 describes a useful feature of Algorithm 2. When applied to an infeasible problem, the algorithm converges to a solution of (5.21), 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 2 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. Any test we devise is necessarily heuristic, however; it is impossible to know for certain whether a larger value of ρ_k would force $\|c(x_k^*)\|$ to be less than η_*. We discuss this point further in section 7.6.

5.7. Second-order optimality. The sLCL method imposes few requirements on the manner in which the LC subproblems are solved. Our implementation (see section 7) 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 5.2 and Theorem 5.4 assert that the sLCL iterates 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 2 generates iterates converging to points satisfying the second-order sufficiency conditions for (GNP).

DEFINITION 5.8 (Strict Complementarity). *The point* (x_*, y_*, z_*) *satisfies strict complementarity if* $\max(x_*, z_*) > 0$.

DEFINITION 5.9 (Second-Order Sufficiency). *The point* (x_*, y_*, z_*) *satisfies the second-order sufficiency conditions for* (GNP) *if it satisfies the first-order conditions* (1.4) *and strict complementarity, and if for all* $\rho \geq 0$,

$$p^T \nabla^2_{xx} \mathcal{L}(x_*, y_*, \rho) p > 0 \tag{5.31}$$

for all $p \neq 0$ *satisfying*

$$J(x_*)p = 0 \text{ and } [p]_j = 0 \text{ for all } j \text{ such that } [x_*]_j = 0 \text{ (and } [z_*]_j > 0\text{).} \tag{5.32}$$

Again, second-order sufficiency is normally defined in terms of the Lagrangian, not the augmented Lagrangian used in Definition 5.9, but our definition is not more restrictive: if (5.31) holds for $\rho = 0$ then it certainly holds for all $\rho \geq 0$.

The following assumption strengthens the first-order conditions (3.1).

ASSUMPTION 5.10. *Let* $\{(x_k^*, \Delta y_k^*, z_k^*)\}$ *be a sequence of local solutions to subproblems* (ELC$_k$)*, and let* $(x_*, \mathcal{K})$ *be a limit point of* $\{x_k^*\}$*. For all* $k \in \mathcal{K}$ *large enough, the following conditions hold at each solution for some* $\delta > 0$ *independent of* k*:*

1. *(Strict Complementarity)*

$$\max(x_k^*, z_k^*) > \delta e;$$

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

$$p^T \nabla^2_{xx} \mathcal{L}(x_k^*, y_k + \Delta y_k^*, \rho) p > \delta \|p\|^2 \tag{5.33}$$

for all $p \neq 0$ *satisfying*

$$J(x_k^*)p = 0 \text{ and } [p]_j = 0 \text{ for all } j \text{ such that } [x_k^*]_j = 0. \tag{5.34}$$

Condition (5.33) implies that the reduced Hessian of $\mathcal{L}$ is uniformly positive definite at all x_k^*. Note that if (5.33) holds for $\rho = 0$, then it holds for all $\rho \geq 0$.

The following result extends the global analysis of section 5.3 to the case in which the iterates generated by Algorithm 2 satisfy Assumption 5.10. Conn et al. [10] show a similar result for their BCL method.

THEOREM 5.11 (Convergence to a local minimizer). *Let* $\{(x_k^*, \Delta y_k^*, z_k^*)\}$ *be a sequence satisfying Assumption* 5.10. *Let* $(x_*, \mathcal{K})$ *be a limit point of* $\{x_k^*\}$, *set* $\widetilde{y}_k = \widetilde{y}(x_k^*, y_k + \Delta y_k^*, \rho_k)$, *and let*

$$(x_*, y_*, z_*) = \lim_{k \in \mathcal{K}} (x_k^*, \widetilde{y}_k, z_k^*) \tag{5.35}$$

be a first-order KKT point of (GNP). *Then* (x_*, y_*, z_*) *is an isolated local minimizer.*

Proof. By hypothesis, x_k^* and z_k^* satisfy Part 1 of Assumption 5.10 for all $k \in \mathcal{K}$. Therefore, their limit points satisfy $\max(x_*, z_*) \geq \delta e > 0$, and so x_* and z_* satisfy strict complementarity (Definition 5.8). Now let p be any nonzero vector satisfying (5.34) for all $k \in \mathcal{K}$ large enough. Then

$$p^T \nabla^2_{xx} \mathcal{L}(x_k^*, y_k + \Delta y_k^*, \rho_k) p = p^T \big(H(x_k^*) - \sum_{i=1}^m [\widetilde{y}_k]_i H_i(x_k^*) \big) p \tag{5.36}$$

for all $k \in \mathcal{K}$ large enough. Part 2 of Assumption 5.10 and (5.36) imply that

$$p^T \big(H(x_k^*) - \sum_{i=1}^m [\widetilde{y}_k]_i H_i(x_k^*) \big) p > \delta \|p\|^2, \tag{5.37}$$

where δ is some positive constant. If we take the limit of (5.37), the continuity of H and H_i (Assumption 1.2) and (5.35) imply that

$$p^T \nabla^2_{xx} \mathcal{L}(x_*, y_*, \rho) p = p^T \big(H(x_*) - \sum_{i=1}^m [y_*]_i H_i(x_*) \big) p \geq \delta \|p\|^2 > 0$$

for all $\rho \geq 0$ and for all $p \neq 0$ satisfying (5.32). Therefore, (x_*, y_*, z_*) satisfies the second-order sufficiency conditions for (GNP), as required. □

Note that section 5.3 (with two options for Step 7) guarantees that the limit (5.35) is a first-order KKT point whether or not there are multiple limit points.

6. Local convergence. In this section we show that the sLCL algorithm preserves the local convergence characteristics of Robinson's original LCL algorithm. Moreover, it can retain fast local convergence under inexact solutions to the subproblems.

Bertsekas [1] and Conn et al. [9, 10] show how to construct a forcing sequence $\{\eta_k\}$ to guarantee that $\|c(x_k^*)\| \leq \eta_k$ will eventually always be true so that the iterates x_k, y_k, and z_k are updated (see Step 4 of Algorithm 2) 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 (5.3). We know from BCL convergence theory that the convergence rate is superlinear if $\rho_k \to \infty$ and linear otherwise (cf. [1] and [9, 10]). Because η_k is reduced at a *sub*linear rate, $\|c(x_k^*)\|$ 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 Algorithm 2 does not increase ρ_k without bound.

Bertsekas [1] and Powell [38] suggest constructing the sequence η_k as

$$\eta_{k+1} = \zeta \|c(x_k^*)\|, \tag{6.1}$$

for some $\zeta < 1$. Within Algorithm 2, this would lead to the following update rule:

$$\rho_{k+1} = \begin{cases} \rho_k & \text{if } \|c(x_k^*)\| \leq \zeta \|c(x_k)\| \\ \tau_\rho \rho_k & \text{if } \|c(x_k^*)\| > \zeta \|c(x_k)\|. \end{cases} \tag{6.2}$$

As ρ_k gets larger, the convergence rate becomes arbitrarily close to superlinear, so that the first case of (6.2) is always satisfied, and ρ_k becomes constant for all k large enough. We prefer not to use rule (6.1) because it may be too strict. Any intermediate (and nonoptimal) 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. [9, 10] does not suffer from this defect and has been proven by them to keep ρ_k bounded. We have used this update in Algorithm 2 (see Steps 9 and 12).

For this analysis and the remainder of this section, 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 consider only iterations $k > \bar{k}$.

The local convergence analysis requires the following second-order condition.

ASSUMPTION 6.1. *The point (x_*, y_*, z_*) satisfies the second-order sufficiency conditions for* (GNP).

We begin by discussing the local convergence rates of the Algorithm 2 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 sLCL becomes equivalent to canonical LCL (and to MINOS) as it approaches the solution.

6.1. Convergence rate. Robinson's [39] local convergence analysis applies to the canonical LCL algorithm under the special case in which $\rho_k \equiv 0$ and each subproblem is solved to full accuracy (i.e., $\omega_k \equiv 0$). He proved that one can expect fast convergence from a good enough starting point. In particular, under Assumptions 1.2, 6.1, and 1.4, we can expect an R-quadratic rate of convergence. (See Ortega and Rheinboldt [37] for an in-depth discussion of root-convergence rates.) For a sufficiently good starting point, Robinson [40] proves that the subproblems (LC_k) are always feasible. He also shows that near a solution, the solutions to the linearly constrained subproblems, if parameterized appropriately, form a continuous path converging to (x_*, y_*, z_*).

In a later paper, Bräuninger [6] 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.

Robinson's local convergence analysis also applies to the canonical LCL algorithm when $\rho_k \equiv \bar{\rho} > 0$. One can see this by considering the optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) + \frac{1}{2}\bar{\rho}\|c(x)\|^2 \\ \text{subject to} & c(x) = 0, \quad x \geq 0 \end{array} \tag{6.3}$$

and applying Robinson's method. The solutions of (6.3) are identical to the solutions of (GNP). The Robinson LCL subproblem objective for problem (6.3) is

$$R_k(x) \equiv f(x) + \tfrac{1}{2}\bar{\rho}\|c(x)\|^2 - y_k^T c(x).$$

The canonical LCL subproblem objective is $\mathcal{L}_k(x) \equiv \mathcal{L}(x, y_k, \rho_k)$, and so $\mathcal{L}_k(x) \equiv R_k(x)$ for all k because $\rho_k \equiv \bar{\rho}$. We then observe that the canonical LCL subproblem corresponding to (GNP), with a penalty parameter $\rho_k \equiv \bar{\rho}$, is *equivalent* to the

Robinson LCL subproblem corresponding to problem (6.3), with $\rho_k \equiv 0$. The convergence characteristics of the canonical LCL algorithm are therefore the same as those demonstrated by Robinson [39]. (However, while the asymptotic convergence rate remains R-quadratic, we expect a different asymptotic error constant.)

Under the assumption that the elastic variables are always equal to 0 and that $\bar{\rho}$ is finite, the steps executed by Algorithms 1 and 2 are identical, and the subproblems (ELC_k) and (LC_k) are also identical. The only difference is the multiplier update formulas:

$$\text{Canonical LCL update} \qquad y_{k+1} = y_k + \Delta y_k^* \tag{6.4a}$$

$$\text{Stabilized LCL update} \qquad y_{k+1} = y_k + \Delta y_k^* - \bar{\rho} c(x_k^*), \tag{6.4b}$$

which differ by the vector $\bar{\rho} c(x_k^*)$. We may think of this vector as a perturbation of the LCL multiplier update (6.4a). Moreover, Robinson [39] shows that 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 sLCL iterates. Robinson's local convergence analysis then applies to the sLCL method.

We summarize the convergence results in Theorem 6.2. Note that the function

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

captures the first-order optimality conditions of (GNP), in the sense that the vector $F(x_*, y_*, z_*) = 0$ 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 6.2 (Robinson [39]; Bräuninger [6]). *Suppose Assumptions* 1.4 *and* 6.1 *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* $k \geq 0$,

$$\|r_k - r_*\| \leq L(\tfrac{1}{2})^{2^k} \tag{6.5}$$

for some positive constant L. *Also, the difference between consecutive iterates converges at an R-linear rate:*

$$\|r_{k+1} - r_k\| \leq \delta(\tfrac{1}{2})^k. \tag{6.6}$$

Robinson does not state (6.6) as part of a theorem, but it is found in the proof of (6.5).

6.2. Asymptotic equivalence to canonical LCL. 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 sLCL 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. 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.

As we show in the next theorem, the elastic variables v_k^* and w_k^* will be bounded by the parameter ω_k that controls the termination of the subproblems. In practice, we might choose to set $\omega_k \equiv \omega_*$ after some number of iterations, in effect asking for accurate subproblem solutions for all remaining iterates. Hence, x_k^* will eventually always satisfy the linearized constraints to within the specified tolerance ω_*.

THEOREM 6.3 (Convergence to inelastic subproblem solutions). *Let $\{(x_k, y_k, z_k)\}$ and $\{(v_k^*, w_k^*)\}$ be the sequences generated by Algorithm 2. Let x_* be the single limit point of $\{x_k^*\}$. Suppose Assumption 1.4 holds and the sequence $\{\rho_k\}$ remains bounded. Then for all k large enough, $\|v_k^*\|_\infty \le \omega_k$ and $\|w_k^*\|_\infty \le \omega_k$.*

Proof. The assumed boundedness of ρ_k implies that $\rho_k \equiv \bar\rho$ for all k large enough. Consider only such k for the remainder of the proof. Set $\underline\sigma = 1/(1+\bar\rho)$.

The hypotheses of the theorem are the same as those for Corollary 5.5. Therefore, Corollary 5.5 applies and $x_k^* \to x_*$ with $c(x_*) = 0$ and $y_k \to y_*$. Because c is continuous, the sequence $\{y_k\}$ is Cauchy, and $\omega_k \to 0$, there exists some $\bar k$ large enough so that

$$\bar\rho\|c(x_k^*)\|_\infty < \tfrac{1}{2}\underline\sigma \tag{6.7a}$$

$$\|y_{k+1} - y_k\|_\infty < \tfrac{1}{2}\underline\sigma - \omega_k \tag{6.7b}$$

for all $k > \bar k$. Step 5 is executed for all k large enough, so that $y_{k+1} = y_k + \Delta y_k^* - \bar\rho c(x_k^*)$. From the triangle inequality and (6.7), we see that

$$\begin{aligned}
\|\Delta y_k^*\|_\infty &\le \|\Delta y_k^* - \bar\rho c(x_k^*)\|_\infty + \bar\rho\|c(x_k^*)\|_\infty \\
&= \|y_k + \Delta y_k^* - \bar\rho c(x_k^*) - y_k\|_\infty + \bar\rho\|c(x_k^*)\|_\infty \\
&= \|y_{k+1} - y_k\|_\infty + \bar\rho\|c(x_k^*)\|_\infty \\
&< \underline\sigma - \omega_k
\end{aligned} \tag{6.8}$$

for all $k > \bar k$. However, Step 7 of Algorithm 2 guarantees that $\underline\sigma \le \sigma_k$ for all k, and so from (6.8),

$$\sigma_k e - \Delta y_k^* > \sigma_k e - (\underline\sigma - \omega_k)e \ge \omega_k e, \tag{6.9}$$

for all $k > \bar k$. Because v_k^* and Δy_k^* both satisfy (3.3e), (6.9) implies that $\|v_k^*\|_\infty \le \omega_k$ for all k large enough. By a similar argument, $\sigma_k e + \Delta y_k^* > \omega_k e$, implying that $\|w_k^*\|_\infty \le \omega_k$ for all k large enough. ∎

7. Implementation. The practical implementation of an algorithm invariably requires many features that are not made explicit by its theory. In this section we discuss some important details of our sLCL implementation. The algorithm has been implemented in MATLAB, version 6 [32] and is called LCLOPT. It uses the Fortran codes MINOS [34, 35] or SNOPT [23] to solve the linearly constrained subproblems. We now turn our attention to an optimization problem with more general constraints and leave (GNP) behind.

7.1. Problem formulation. LCLOPT solves problems of the form

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

This matches the problem formulation used by SNOPT and is 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

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

7.2. The main algorithm. 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. The main tasks of the outer level are to form the subproblems, update solution estimates, update parameters, and test for convergence or errors.

7.3. Solving the LC subproblems. For the linearly constrained subproblems, MINOS uses a reduced-gradient method, coupled with a quasi-Newton approximation of the reduced Hessian of 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 is 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 [24].

Both MINOS and SNOPT are available as libraries of Fortran 77 routines. We implemented MEX interfaces [33] 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 $\mathcal{L}_k$. In turn, the routine for $\mathcal{L}_k$ makes calls to routines (available as MATLAB or MEX routines) to evaluate the original nonlinear functions f and c.

7.4. 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 sLCL iteration, LCLOPT uses SNOPT to solve one of the following *proximal-point* (PP) problems:

$$\boxed{\begin{array}{ll} \text{(PP1) or (PP2)} & \underset{x}{\text{minimize}} \quad \|x-\widetilde{x}\|_1 \quad \text{or} \quad \tfrac{1}{2}\|x-\widetilde{x}\|_2^2 \\ & \text{subject to} \;\; l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u, \end{array}}$$

where $\widetilde{x}$ is a vector provided by the LCLOPT user. The solution is used as initial point x_0 for the sLCL algorithm. The PP objective functions help 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 the PP problem proves infeasible, (NPi) is declared infeasible and LCLOPT exits immediately with an error message.

The computational results presented in section 8 were derived by using (PP2) to compute x_0. As suggested by Gill et al. [24], 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.

7.5. Early termination of the LC subproblems. The global convergence results for the sLCL algorithm (cf. Lemma 5.2 and Theorem 5.4) assume that the optimality tolerances ω_k for the subproblems converge 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 6.2.

To construct $\{\omega_k\}$, we replace Step 1 of Algorithm 2 by

$$\begin{aligned} \omega &\leftarrow \min(\omega_k, \|F(x_k, y_k, z_k)\|_\infty^2) \\ \omega_{k+1} &\leftarrow \max(\tfrac{1}{2}\omega, \omega_*), \end{aligned} \tag{7.1}$$

where ω_0 can be set by a user to any value between $\frac{1}{2}$ and ω_*. The update (7.1) guarantees that $\omega_k \to \omega_*$, as required. A gentler reduction is discussed in section 8.5.

Following the prescription outlined in section 3.2, 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.

7.6. Detecting infeasibility and unboundedness. As discussed in section 5.6, Algorithm 2 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 $\bar{\rho}$. In particular, at Step 11, Algorithm 2 exits and (NPi) is declared infeasible if

$$\max(\|[l_c - c_k]^+\|_\infty, \|[c_k - u_c]^+\|_\infty) > \eta_* \qquad \text{and} \qquad \rho_k > \bar{\rho},$$

where l_c and u_c are the lower and upper bounds for the nonlinear constraints and $[\,\cdot\,]^+$ is the positive part of a vector. For the computational results in section 8 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 $f(x)$ 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.

7.7. Summary of the stabilized LCL method. The following is a summary of the sLCL algorithm implemented in LCLOPT. We assume that $\widetilde{x}$ is given and the starting tolerances ω_0, η_0 and the parameters ρ_0, 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$_k$) with optimality tolerance ω_k to obtain $(x_k^*, \Delta y_k^*, z_k^*)$. Set $y_k^* = y_k + \Delta y_k^*$.
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: $(x_{k+1}, y_{k+1}, z_{k+1}) \leftarrow (x_k^*, y_k^* - \rho_k c(x_k^*), z_k^*)$. Keep the penalty parameter ρ_k fixed and reset the elastic weight σ_k.
7. Test convergence: If $(x_{k+1}, y_{k+1}, z_{k+1})$ satisfies the optimality conditions for (NPi), declare optimality, 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 next nonlinear feasibility threshold η_{k+1} and LC subproblem optimality tolerance ω_{k+1}, so that $\{(\omega_k, \eta_k)\} \to (\omega_*, \eta_*)$.
10. Set $k \leftarrow k + 1$. Return to Step 2.

8. Numerical results. This section summarizes the results of applying two versions of LCLOPT to a subset of nonlinearly constrained test problems (specified in the later subsections) from the COPS 2.0 [15], Hock-Schittkowski [29], and CUTE [5] test suites. The first version uses AMPL/MINOS 5.5 [19], version 19981015, to solve the sequence of linearly constrained subproblems; the second version uses SNOPT version 6.1-1(5).

We used the AMPL versions of all problems, as formulated by Vanderbei [44]. A MEX interface to the AMPL libraries makes functions and gradients available in MATLAB (see Gay [22] 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. (The CUTE versions of the problems could also have been used from MATLAB.)

Figure 8.1 shows *performance profiles*, as described by Dolan and Moré [16], for the two versions of LCLOPT (the dotted and dashed lines) and MINOS (the solid line). The statistic profiled in the top chart is the total number of function and gradient evaluations. In the bottom chart it is the total minor iterations. (Because the

nonlinear objective, constraint, and gradient evaluations always occur together, each evaluation of (f, c, g, J) is counted once.) All 135 problems selected from the COPS, Hock-Schittkowski, and CUTE test suites are included in each profile. For each solver and each τ on the horizontal axis, a profile shows the percentage of problems for which the statistic in question (following a successful solve) is within a factor τ of the best.

We see that LCLOPT with MINOS as subproblem solver solved the largest proportion of problems and may therefore be regarded as the most reliable method. Compared with MINOS, LCLOPT tends to require more minor iterations (a measure of total computational work). We comment further in section 8.5.

8.1. Default parameters. Figure 8.2 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 the subproblem optimality tolerances ω_k (see section 7.5).

Each test problem supplies a default starting point. It is used as $\widetilde{x}$ in the proximal-point problem (see section 7.4). The initial multiplier vector 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 options `Start = 'Hot'`.

The parameters used by Algorithm 2 are set as follows. The upper bound on the elastic penalty parameters is $\overline{\sigma} = 10^4$. The initial elastic 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 $\tau_\rho = \tau_\sigma = 10$. As suggested in [10], 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 $\eta_0 = 1$.

Default options are used for the MINOS benchmarks, except for the special values `Major Iterations 500` and `Superbasics Limit 2000`.

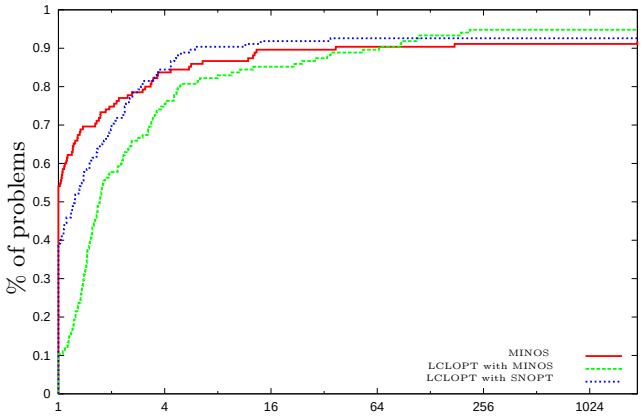
8.2. The COPS test problems. The COPS 2.0 collection [15] comprises 17 problems. Five problems are excluded for the following reasons:

- *bearing*, *minsurf*, and *torsion* are unconstrained or bound constrained.
- *glider* and *marine* cause system errors when called from the AMPL MEX interface.

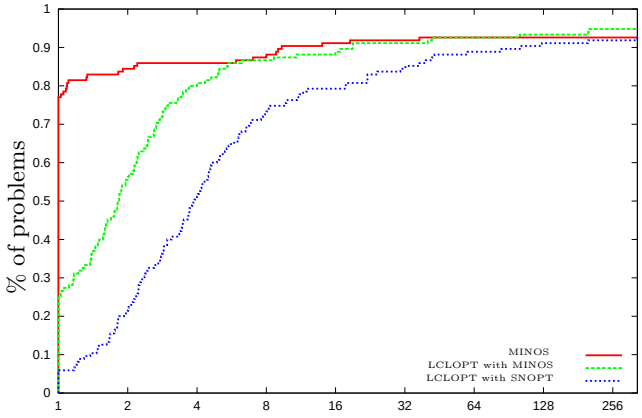
The dimensions of the COPS test problems can be adjusted. In all cases, the solvers were applied to the largest version of the problem that would not cause the system to page memory to disk. Tables 8.1 and 8.2 specify the chosen dimensions.

As shown in Table 8.3, 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.

We note that different local optima appear to have been found for problems *camshape, methanol, polygon*, and *rocket*. Also, many minor iterations were required on *catmix, elec*, and *robot* with SNOPT as its subproblem solver. Especially during early major iterations, SNOPT was unable to solve the subproblems to the required



(a) Function and gradient evaluations



(b) Minor iterations

FIG. 8.1. *Performance profiles for the number of function and gradient evaluations and the total minor iterations. For each solver, the vertical axes represent the percentage of problems for which the relevant statistic is within a factor τ of the best (among all three solvers). The horizontal axes are based on a log scale. The profiles include results for all 135 test problems.*

```
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

FIG. 8.2. *The fixed optional parameters for every subproblem solve.*

TABLE 8.1
The heads used in Tables 8.2 and 8.5

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 8.2
Dimensions of 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

optimality tolerance within the 5000 iteration limit. Rather than terminate with an error message, LCLOPT restarts SNOPT several times on the same subproblem. In practice, a different strategy would be adopted, but our goal was to test the robustness of the outer iterations (the sLCL method), not the robustness of the subproblem solvers.

TABLE 8.3
Summary of results for the 12 selected COPS test problems

	LCLOPT		
	(MINOS)	(SNOPT)	MINOS
Optimal	12	11	10
False Infeasibility		1	1
Terminated by iteration limit			1
Major iterations	118	179	380
Minor iterations	53950	147518	61388
Function evaluations	53081	11014	63701

8.3. The Hock-Schittkowski test problems. The HS test suite contains 86 nonlinearly constrained problems [29]. These are generally small and dense problems. We exclude five problems from this set for the following reasons:

- *hs67, hs85*, and *hs87* are not smooth.
- *hs68* and *hs69* require external functions.

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.

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

Recall that LCLOPT and MINOS use only first derivatives and hence may not converge to local minimizers 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. Similar differences exist for problems *hs47* and *hs77*.

TABLE 8.4
Summary of results for the 81 selected Hock-Schittkowski test problems

	LCLOPT		
	(MINOS)	(SNOPT)	MINOS
Optimal	80	80	80
False infeasibility	1	1	1
Major iterations	654	648	1160
Minor iterations	7415	25290	10111
Function evaluations	12269	14712	27127

8.4. A selection of CUTE test problems. With the `select` utility [5], we extracted from the CUTE test suite dated September 7, 2000, problems with the following characteristics (where `*` 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, but we excluded 66 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 either by 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*.

(To avoid such an exclusion rate, future experiments will work directly with the CUTEr interface [26].) Of the remaining 42 problems, 17 can be adjusted in size. The solvers were again applied to the largest versions that would not cause memory paging. Table 8.5 gives the dimensions.

TABLE 8.5
Dimensions of the variable-size CUTE test problems

Problem	m	m_c	n	n_c	n_f
bdvalue	1000	1000	1000	1000	0
bratu2d	4900	4900	4900	4900	0
bratu3d	512	512	512	512	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
hadamard	257	128	65	64	65
manne	731	364	1094	364	729
reading1	5001	5000	10001	10000	10000
reading3	103	101	202	202	202
sreadin3	5001	5000	10000	9998	9998
ssnlbeam	21	10	31	11	22
svanberg	1001	1000	1000	1000	1000
ubh5	14001	2000	19997	6003	0

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 34 problems to first-order optimality. Table 8.6 summarizes these results. In terms of major iterations, the sLCL method was largely independent of the subproblem solver if the latter did not fail. The high *minor* iterations count for LCLOPT with SNOPT is due to an inordinate number of iterations (about 83,500) to solve *cresc50*, *svanberg*, and *ubh5*, and to another 102,500 minor iterations for failed subproblem solves terminated by the iteration limit.

TABLE 8.6
Summary of results for the 42 selected CUTE test problems

	LCLOPT		
	(MINOS)	(SNOPT)	MINOS
Optimal	36	34	34
False infeasibility	4	3	3
Terminated by iteration limit	1	3	1
Terminated by superbasics limit		1	
Unbounded/badly scaled			3
Final point cannot be improved	1	1	1
Major iterations	400	368	1149
Minor iterations	70476	278162	29021
Function evaluations	59216	57732	53069

8.5. Importance of early termination. Tables 8.3, 8.4 and 8.6 show that MINOS solved many of the test problems using a reasonable number of minor iterations but rather many *major* iterations (i.e., LC subproblem solves). This is because MINOS terminates progress on each of its subproblems after only 40 minor iterations. In contrast, LCLOPT attempts to constrain the subproblem iterations by means of an initially loose optimality tolerance (we set $\omega_0 = \sqrt{\omega_*}$ for the numerical results). 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 (and 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 while satisfying the requirement that the subproblem optimality tolerances ω_k converge to zero (cf. Lemma 5.2). For example, LCLOPT might specify a small subproblem iteration limit initially, and only gradually increase the 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. section 6.1). Bräuninger [7] 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 6.2). The first-order KKT conditions (3.1) for the LCL subproblem can be expressed as

$$\begin{pmatrix} \nabla^2 \mathcal{L}_k(x_k) & J_k^T \\ J_k & \end{pmatrix} \begin{pmatrix} p \\ -y \end{pmatrix} + O(\|p\|^2) = \begin{pmatrix} -g_k + J_k^T y_k \\ -c_k \end{pmatrix}, \tag{8.1}$$

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 [39, 40] shows that the correct active set is identified by the subproblems near a solution.) The nonlinear equations (8.1) are closely related to the linear equations that would be derived from applying Newton's method to (3.1) (again, ignoring bound constraints). In that case, the theory from inexact Newton methods (Dembo et al. [14]) 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 (8.1) and the Newton equations hints at the possibility of recovering the quadratic convergence rate of the LCL and sLCL methods by reducing ω_k at the rate $O(\|F(x_k, y_k, z_k)\|)$. See also Conn et al. [12]. We note, however, that stronger assumptions may be needed on the smoothness of the nonlinear functions. This issue deserves more study.

8.6. 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 $\bar{\rho}$. In contrast, LCL methods can converge when $\rho_k \equiv 0$ if they are started near a solution (see section 6.2).

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 5.2, and may be effective in practice. A form of this strategy was used for the runs shown in section 8.

9. Conclusions. The stabilized LCL method developed in this paper is a generalization of the augmented Lagrangian methods discussed in section 4 and it shares the strengths of its predecessors: it is globally convergent (the BCL advantage) and it has quadratic local convergence (the LCL advantage). The ℓ_1 penalty function brings the two together. Because the method operates in a reduced space (like all LCL methods), it is less sensitive than BCL methods to the choice of each penalty parameter ρ_k.

9.1. A second-derivative LC solver. We prove in section 5.7 that the sLCL method will converge to second-order stationary points if the 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.

The sLCL 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 section 8.5). Over the same number of iterations, a subproblem solver using second derivatives may make more progress toward a solution than a first-derivative solver.

Any future sLCL implementation would ideally be flexible enough to allow for a variety of solvers to be used for the LC subproblems. The choice of 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 et al. [4, 3]). These may be used by SQP and interior methods for nonlinearly constrained (NC) problems. 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 sLCL algorithm.

9.2. Looking ahead (and behind). A Fortran 90 implementation of the sLCL algorithm, to be named KNOSSOS, is currently under development [21]. As in Robinson [39] and the MINOS implementation [35], a key concept is *departure from linearity* (meaning the difference between the constraint functions and their linearization at the current solution estimate x_k). For problem (GNP), we define d_k and a *modified augmented Lagrangian* $\mathcal{M}_k$ as follows:

$$\begin{aligned} d_k(x,v,w) &= c(x) - \bar{c}_k(x) - v + w, \\ \mathcal{M}_k(x,v,w) &= f(x) - y_k^T d_k(x,v,w) + \tfrac{1}{2}\rho_k \|d_k(x,v,w)\|^2. \end{aligned}$$

We then use $\mathcal{M}_k$ in place of $\mathcal{L}_k$ in the elastic subproblem (ELC$_k$) of section 3:

(ELC$''_k$)	$\underset{x,v,w}{\text{minimize}}$	$\mathcal{M}_k(x,v,w) + \sigma_k e^T(v+w)$
	subject to	$\bar{c}_k(x) + v - w = 0$
		$x, v, w \geq 0.$

Note that $d_k(x,v,w)$ and $c(x)$ have the same value at any point (x,v,w) that satisfies the (elastic) linearized constraints $\bar{c}_k(x)+v-w=0$. Hence $\mathcal{M}_k$ and $\mathcal{L}_k$ have the same value at such a point. The primal solutions for (ELC$_k''$) and (ELC$_k$) are therefore the same. For points (x,v,w) satisfying the elastic linearized constraints, let $\widetilde{y}_k(x,v,w) \equiv y_k - \rho_k d_k(x,v,w)$. Then a solution $(x_k^*, v_k^*, w_k^*, \pi_k^*, z_k^*)$ of (ELC$_k''$) satisfies the first-order KKT conditions

$$\begin{aligned}
\bar{c}_k(x) + v - w &= 0 \\
g(x) - (J(x) - J_k)^T \widetilde{y}_k(x,v,w) - J_k^T \pi &= z \\
\min(x, z) &= 0 \\
\min(v,\, \widetilde{y}_k(x,v,w) + \sigma_k e - \pi) &= 0 \\
\min(w, -\widetilde{y}_k(x,v,w) + \sigma_k e + \pi) &= 0.
\end{aligned}$$

Now defining

$$\Delta y_k^* \equiv \pi_k^* - \widetilde{y}_k(x_k^*, v_k^*, w_k^*),$$

we find that $(x_k^*, v_k^*, w_k^*, \Delta y_k^*, z_k^*)$ satisfies (3.1), the first-order KKT conditions for (ELC$_k$). The multiplier update used in Algorithm 2 (see Step 5) is therefore

$$y_{k+1} = y_k + \Delta y_k^* - \rho_k c(x_k^*) = \widetilde{y}(x_k^*, v_k^*, w_k^*) + \Delta y_k^* = \pi_k^*.$$

In other words, the new estimate y_{k+1} is the dual solution of (ELC$_k''$). This is exactly the multiplier update first suggested by Robinson and used in MINOS.

The use of $\mathcal{M}_k$ in (ELC$_k''$) follows another important aspect of the MINOS implementation for problems in which only some of the variables enter the constraints nonlinearly. When $v = w = 0$, the Lagrangian term $y_k^T d_k$ and the penalty term $\|d_k\|^2$ are both nonlinear in the same variables, whereas $\|c(x)\|^2$ would be nonlinear in all components of x.

In retrospect, we see that when the linearized constraints $\bar{c}_k(x) = 0$ are infeasible, the MINOS strategy of relaxing those constraints in gentle stages corresponds to introducing v and w (without the help of σ_k). However, MINOS continues to define $\mathcal{M}_k$ in terms of $d_k = c(x) - \bar{c}_k(x)$ rather than $c(x) - \bar{c}_k(x) - v + w$, and therefore is no longer using the true augmented Lagrangian. This explains the increased failure rate observed with MINOS when the subproblems continue to be infeasible. The sLCL approach must regard v and w as nonlinear variables within the LC subproblems, but this is a small price to pay for improved reliability, and the actual nonlinearity of v and w fades as they become zero near a solution.

Acknowledgements. The authors are indebted to Margaret Wright, Nick Gould and Annick Sartenaer for their exceptionally careful readings of this paper. Many valuable suggestions helped clarify the description. In particular, Nick Gould helped revise Theorems 5.4 and 5.7, and his cautions about the ℓ_1 penalty function led to Theorem 6.3. Annick Sartenaer's remarkable list of comments also led to a correction of Theorem 5.4, as well as refinement of the proof of Lemma 5.3 and many other detailed changes.

REFERENCES

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

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

[3] C. Bischof, A. Carle, P. Hovland, P. Khademi, and A. Mauer, *ADIFOR 2.0 Users' Guide*, Tech. Report 192, Mathematics and Computer Science Division, Argonne, IL, June 1998.

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

[5] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[6] J. Bräuninger, *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, Würzburg, 1977, pp. 33–41.

[7] ———, *A globally convergent version of Robinson's algorithm for general nonlinear programming problems without using derivatives*, J. Opt. Theory Applic., 35 (1981), pp. 195–216.

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

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

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

[11] ———, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer-Verlag, Berlin, 1991.

[12] ———, *On the number of inner iterations per outer iteration of a globally convergent algorithm for optimization with general nonlinear equality constraints and simple bounds*, in Numerical Analysis 1991, D. F. Griffiths and G. A. Watson, eds., Longman Scientific & Technical, Harlow, Essex, UK, 1992, pp. 49–68.

[13] ———, *Trust-Region Methods*, MPS-SIAM Series on Optimization, SIAM Publications, Philadelphia, 2000.

[14] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[15] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with COPS*, Tech. Report ANL/MCS-246, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2000. Revised January 2, 2001.

[16] ———, *Benchmarking optimization software with performance profiles*, Math. Prog., 91 (2002), pp. 201–213.

[17] R. Fletcher, *An ℓ_1 penalty method for nonlinear constraints*, in Numerical Optimization, P. T. Boggs, R. H. Byrd, and R. B. Schnabel, eds., Boulder, CO, June 1984, SIAM, pp. 26–40.

[18] ———, *Practical Methods of Optimization*, John Wiley and Sons, New York, second ed., 1987.

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

[20] M. P. Friedlander, *A globally convergent linearly constrained Lagrangian method for nonlineary constrained optimization*, PhD thesis, Stanford University, Stanford, CA, August 2002.

[21] M. P. Friedlander and M. A. Saunders, *An LCL implementation for nonlinear optimization,* presented at 18th International Symposium on Mathematical Programming, Copenhagen, Denmark, August 2003. http://www.stanford.edu/group/SOL/talks.html.

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

[23] P. E. Gill, W. Murray, and M. A. Saunders, *User's guide for SNOPT 5.3: a Fortran package for large-scale nonlinear programming*, Numerical Analysis Report 97-5, Department of Mathematics, University of California, San Diego, La Jolla, CA, 1997.

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

[25] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[26] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr and SifDec: A constrained and unconstrained testing environment, revisited*, ACM Transactions on Mathematical Software, 29 (2003), pp. 373–394.
[27] S.-P. Han and O. L. Mangasarian, *Exact penalty functions in nonlinear programming*, Math. Prog., 17 (1979), pp. 251–269.
[28] M. R. Hestenes, *Multiplier and gradient methods*, J. Opt. Theory Applic., 4 (1969), pp. 303–320.
[29] W. Hock 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.
[30] O. L. Mangasarian, *Nonlinear Programming*, McGraw-Hill, New York, 1969.
[31] N. Maratos, *Exact penalty function algorithms for finite dimensional and optimization problems*, PhD thesis, Imperial College of Science and Technology, London, UK, 1978.
[32] MathWorks, *MATLAB User's Guide*, The MathWorks, Inc., Natick, Massachussetts, 1992.
[33] ———, *MATLAB: External Interfaces*, Natick, Massachussetts, 1995.
[34] B. A. Murtagh and M. A. Saunders, *Large-scale linearly constrained optimization*, Math. Prog., 14 (1978), pp. 41–72.
[35] B. A. Murtagh and M. A. Saunders, *A projected Langrangian algorithm and its implementation for sparse nonlinear constraints*, Mathematical Programming Study, 16 (1982), pp. 84–117.
[36] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[37] J. M. Ortega and W. C. Rheinboldt, *Iterative Solutions of Nonlinear Equations in Several Variables*, Academic Press, London, 1970.
[38] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, in Optimization, R. Fletcher, ed., Academic Press, London and New York, 1969, ch. 19.
[39] S. M. Robinson, *A quadratically-convergent algorithm for general nonlinear programming problems*, Math. Prog., 3 (1972), pp. 145–156.
[40] ———, *Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms*, Math. Prog., 7 (1974), pp. 1–16.
[41] J. B. Rosen, *Two-phase algorithm for nonlinear constraint problems*, in Nonlinear Programming 3, O. Mangasarian, R. Meyer, and S. Robinson, eds., New York, 1978, Academic Press, pp. 97–124.
[42] J. B. Rosen and J. Kreuser, *A gradient projection algorithm for non-linear constraints*, in Numerical Methods for Nonlinear Optimization, F. A. Lootsma, ed., Academic Press, London, 1972, pp. 297–300.
[43] G. Van Der Hoek, *Asymptotic properties of reduction methods applying linearly equality constrained reduced problems*, Mathematical Programming Study, 16 (1982), pp. 162–189.
[44] R. Vanderbei, *Benchmarks for nonlinear optimization.* http://www.princeton.edu/~rvdb/bench.html, December 2002.