JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 79, No. 3, DECEMBER 1993

Analysis and Implementation of a Dual Algorithm for Constrained Optimization[1,2]

W. W. HAGER[3]

Communicated by E. Polak

Abstract. This paper analyzes a constrained optimization algorithm that combines an unconstrained minimization scheme like the conjugate gradient method, an augmented Lagrangian, and multiplier updates to obtain global quadratic convergence. Some of the issues that we focus on are the treatment of rigid constraints that must be satisfied during the iterations and techniques for balancing the error associated with constraint violation with the error associated with optimality. A preconditioner is constructed with the property that the rigid constraints are satisfied while ill-conditioning due to penalty terms is alleviated. Various numerical linear algebra techniques required for the efficient implementation of the algorithm are presented, and convergence behavior is illustrated in a series of numerical experiments.

Key Words. Constrained optimization, multiplier methods, preconditioning, global convergence, quadratic convergence.

1. Introduction

We consider optimization problems of the following form:

$$\min f(x), \quad \text{s.t. } h(x) = 0, \qquad x \in \Omega, \tag{1}$$

where x is a vector in R^n, f is a real-valued function, h maps R^n to R^m, and $\Omega \subset R^n$. The constraint set Ω contains the explicit constraints; these are the constraint that must be satisfied accurately. The constraint $h(x) = 0$, on the

[1]This research was supported by the National Science Foundation Grant DMS-89-03226 and by the U.S. Army Research Office Contract DAA03-89-M-0314.

[2]We thank the referees for their many perceptive comments which led to substantial improvements in the presentation of this paper.

[3]Professor, Department of Mathematics, University of Florida, Gainesville, Florida.

0022-3239/93/1200-0427$07.00/0

other hand, will be satisfied approximately in our numerical algorithms, and as the iterations progress, the constraint violation will be reduced. In our numerical experiments, we include all nonlinear constraints in h, while the linear equalities and inequalities are incorporated in the constraint $x \in \Omega$. More precisely, in our computer code, we assume that Ω is given by

$$\Omega = \{x: Ax = b, l \leq x \leq u\}, \tag{2}$$

where A is a matrix and b is a vector of compatible dimensions, l is a vector of lower bounds, and u is a vector of upper bounds. Of course, any optimization problem constrained by systems of equalities and inequalities can be expressed in the form (1) with Ω given by (2).

Our goal is to find a feasible point that satisfies the Kuhn–Tucker condition associated with (1). Let L denote the Lagrangian defined by

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

where the superscript T denotes transpose. If Ω is given by a system of equalities and inequalities,

$$\Omega = \{x \in R^n: g(x) = 0 \text{ and } G(x) \leq 0\},$$

where g maps R^n to R^l and G maps R^n to R^k, then a point $x \in \Omega$ with $h(x) = 0$ satisfies the Kuhn–Tucker condition associated with (1) if there exists a vector $\lambda \in R^m$ such that

$$-\nabla_x L(\lambda, x) \in N_x(\Omega),$$

where ∇_x stands for the gradient with respect to x and $N_x(\Omega)$ is the normal cone defined by

$$N_x(\Omega) = \{\mu^T \nabla g(x) + v^T \nabla G(x): v \geq 0, v_i = 0 \text{ if } g_i(x) < 0, \mu \text{ arbitrary}\}.$$

Given $x \in \Omega$ and $\lambda \in R^m$, let E denote the error expression defined by

$$E(\lambda, x) = K(\lambda, x) + C(x),$$

where $C(x) = |h(x)|$ measures the constraint violation in some norm $|\cdot|$ and K measures the error in the Kuhn–Tucker condition,

$$\begin{aligned} K(\lambda, x) &= \text{distance}\{-\nabla_x L(\lambda, x), N_x(\Omega)\} \\ &= \text{minimum}\{|\nabla_x L(\lambda, x) + y|: y \in N_x(\Omega)\}. \end{aligned}$$

For convenience, we use the Euclidean norm throughout this paper, although any norm can be employed.

Observe that, if $K(\lambda, y) = 0$ for some $y \in \Omega$ and $\lambda \in R^m$, then $E(\lambda, y) = 0$ for the problem

$$\min f(x), \quad \text{s.t. } h(x) = z, \qquad x \in \Omega, \tag{3}$$

where $z = h(y)$. Loosely speaking, y is an optimal solution of the wrong problem, assuming that $h(y)$ does not vanish. Hence, the constraint error vanishes at any point in the feasible set, while the Kuhn–Tucker error vanishes at any point that optimizes (3) for some z.

The algorithm developed in this paper is based on the augmented Lagrangian techniques of Ref. 1. Recall that an augmented Lagrangian is obtained from the ordinary Lagrangian by adding a penalty term. If $p \geq 0$ denotes the penalty parameter, then we utilize the quadratic-penalty augmented Lagrangian defined by

$$L_p(\lambda, x) = f(x) + \lambda^T h(x) + p|h(x)|^2.$$

Multiplier methods seem to originate from work by Arrow and Solow (Ref. 2), Hestenes (Ref. 3), and Powell (Ref. 4). Additional results are developed in the sequence of papers (Refs. 5–7) by Rockafellar. See Ref. 8 by Bertsekas for a comprehensive study of multiplier methods. Results for problems formulated in a Hilbert space appear in Refs. 9 and 10. Some interesting applications of augmented Lagrangian techniques to boundary-value problems are developed by Fortin, Glowinski, and their collaborators in Ref. 11.

With the notation given above, the algorithm that we focus on has the following general form: If x_k denotes the current approximation to a solution of (1) and λ_k denotes the current estimate for a Lagrange multiplier associated with the constraint $h(x) = 0$, then the next approximations x_{k+1} and λ_{k+1} are obtained by successively executing the following two steps when the convergence is at least linear:

Step 1. Constraint Step. Apply an iterative method to the equation $h(x) = 0$, stopping the iteration when the constraint error is less than the Kuhn–Tucker error.

Step 2. Kuhn–Tucker Step. Apply an iterative method to the equation $K(\lambda, x) = 0$, stopping the iteration when the Kuhn–Tucker error is sufficiently small relative to the constraint error.

When the Constraint Step followed by the Kuhn–Tucker Step do not decrease the total error at least linearly, execute the following Global Step, then branch back to the Constraint Step.

Step 3. Global Step. Increase the penalty and apply a preconditioned iterative technique to the problem

$$\text{minimize}\{L_p(\lambda_k, x): x \in \Omega\},$$

stopping when the Kuhn–Tucker error is smaller than the constraint error.

In this paper, we analyze a Newton–Armijo iterative implementation of the Constraint Step, and we devise a new iterative scheme for the Kuhn–Tucker Step. Assuming an independence condition for the gradients of the active constraints, we show that the 3-step algorithm possesses a global convergence property. Of course, independence of constraint gradients implies that the number of equality constraints is at most n. If in addition, a second-order sufficient optimality condition holds, then the Constraint Step followed by the Kuhn–Tucker Step possesses a quadratic convergence property. A new preconditioner is developed for the Global Step that eliminates instabilities associated with the penalty term in the augmented Lagrangian while enforcing the explicit constraints. The paper concludes with a series of numerical experiments in which our algorithm is compared to a reduced gradient, quasi-Newton algorithm, and to a sequential quadratic programming algorithm.

As discussed in Ref. 1, our approach to the constrained optimization problem is related to an algorithm of Rosen and Kreuser (Ref. 12). In the scheme of Rosen and Kreuser, each iteration involves the minimization of the Lagrangian subject to linearized constraints. By the analysis of Robinson in Ref. 13, this scheme is locally quadratically convergent. The software package MINOS (Ref. 14) provides an implementation of this scheme in which the Lagrangian is replaced by an augmented Lagrangian. Note that the Rosen–Kreuser method as well as the usual SQP method reduce both the constraint error and the Kuhn–Tucker error simultaneously, while in our approach, each error is treated in separate steps that are loosely coupled together. This allows us to apply equation solving techniques to the constraints and unconstrained optimization techniques to the Kuhn–Tucker error.

The analysis of Coleman and Conn in Refs. 15 and 16 demonstrates a connection between SQP methods and a related 2-step process. In each iteration of the SQP method, a quadratic approximation to the Lagrangian is minimized subject to linearized constraints. Coleman and Conn show that solving this quadratic programming problem is nearly the same as applying a single Newton step to the constraints, then solving the quadratic program subject to a null space constraint. They show that this 2-step process possesses a superlinear convergence property similar to that of SQP

methods. Again, Coleman and Conn's scheme, like the Rosen–Kreuser and SQP schemes, treats both the constraint error and the Kuhn–Tucker error simultaneously, while our approach addresses each error separately.

2. Constraint Error

We consider the following Newton–Armijo (see Ref. 17) process to solve the equation $h(x) = 0$. Let σ and τ be fixed positive constants less than 1, and let $w_0 = x_k$ be the starting guess. Then, the new iterate w_{i+1} is obtained from the current iterate w_i by the rule

$$w_{i+1} = w_i + s_i d_i, \tag{4}$$

where

$$d_i = \arg\min\{|d|: \nabla h(w_i)d = -h(w_i),\ w_i + d \in \Omega\}, \tag{5}$$

and $s_i = \sigma^j$, where j is the smallest nonnegative integer with the property that, for $s = \sigma^j$, we have

$$|h(w_i + sd_i)| \leq (1 - \tau s)|h(w_i)|. \tag{6}$$

There is an extensive literature concerning the convergence properties of Newton's method for systems of equations and inequalities. Some of the relevant literature includes Refs. 18–22. One difference between the iteration (4)–(5) and the iterations in these earlier papers is that, in the earlier work, each of the constraints is linearized, while in (4)–(5) only part of the constraints is linearized. The scheme (4)–(5) is most closely related to the Armijo scheme of Ref. 21, modified to take into account the constraint set Ω and the fact that inequality constraints are embedded in Ω.

In analyzing this algorithm, we assume for convenience that Ω is given by

$$\Omega = \{x \in R^n: g(x) = 0\}, \tag{7}$$

where g maps R^n to R^l. Since an inequality $G(x) \leq 0$ can be converted to an equality by Valentine's device, inequalities can be embedded in the equation $g(x) = 0$ (see Ref. 23); that is, introduce an extra variable z and work with the equation $G(x) + z^2 = 0$. Moreover, if the gradients of the active constraints in the original problem are linearly independent, then the gradients of the transformed equality constraints are linearly independent. The conversion of inequalities to equalities is done to facilitate the analysis. This conversion is not needed for the implementation of our algorithm. The global convergence properties of the iteration (4)–(5) are developed in Section 6, while the local convergence properties are described by

Theorem 2.1. Suppose that $x = x^*$ satisfies the equations $h(x) = 0$ and $g(x) = 0$, g and h are twice continuously differentiable near x^*, and the gradients $\nabla h_j(x^*)$ and $\nabla g_k(x^*)$ are linearly independent. Then, there exists a neighborhood Δ of x^* and a constant c such that, for any $w_0 \in \Delta \cap \Omega$, the iteration (4)–(5), where Ω is given by (7), converges to a point w with the following properties: $h(w) = g(w) = 0$ and $s = 1$ satisfies (6) for each i. Moreover, for every i, we have

$$|w_i - w| \leq c|h(w_0)|2^{-2^i}, \quad |w_i - w_0| \leq c|h(w_0)|, \quad |h(w_{i+1})| \leq c|h(w_i)|^2. \quad (8)$$

Thus, the root convergence order (see Ref. 24) of the iteration (4)–(5) is at least 2.

A proof of this result appears in Appendix A (Section 11).

3. Convergence Analysis with Rigid Constraints

In this section, we state a result that provides the basis for our algorithm to iteratively solve the equation $K(\lambda, x) = 0$. This result extends the convergence analysis of Ref. 1 to handle the rigid constraint $x \in \Omega$. An analysis of rigid constraints for the classical method of multipliers appears in Ref. 8, pp. 141–144, by Bertsekas, while an analysis of a sequential quadratic programming scheme with rigid constraints appears in Ref. 25.

As in the analysis of Bertsekas, we take Ω of the form (7). Let us assume that (1) has a local minimizer x^*, that f, g, h are twice continuously differentiable in a neighborhood of x^*, and that the constraint gradients $\nabla g_j(x^*)$ and $\nabla h_k(x^*)$ are linearly independent. In addition, if $\lambda = \lambda^*$ and $\mu = \mu^*$ are the unique solutions to the equation

$$\nabla f(x^*) + \lambda^T \nabla h(x^*) + \mu^T \nabla g(x^*) = 0,$$

we assume that the Hessian $\nabla_x^2[f(x) + h(x)^T\lambda^* + g(x)^T\mu^*]_{x=x^*}$ is positive definite in the null space of $\nabla h(x^*)$ intersect the null space of $\nabla g(x^*)$. This assumption, which will be referred to as the second-order sufficiency condition throughout this paper, implies that x^* is a strict local minimizer of (1); see Ref. 26.

Theorem 3.1. If the second-order sufficiency condition holds, Ω is given by (7), and the constraint gradients are linearly independent at a local minimizer x^* of (1), then there exists a neighborhood Δ of (x^*, λ^*) for which the problem

$$\text{minimize}\{L_p(\lambda_k, x): \nabla h(y_k)(x - y_k) = 0, g(x) = 0\} \tag{9}$$

has a local minimizer $x = x_{k+1}$, whenever (y_k, λ_k) lies in Δ. Moreover, there exists a constant c such that

$$|x_{k+1} - x^*| \leq c|\lambda_k - \lambda^*|^2 + c|y_k - x^*|^2 + c|h(y_k)|,$$

for every (y_k, λ_k) in Δ with $y_k \in \Omega$.

A proof of Theorem 3.1 appears in Appendix B (Section 12). Approximations λ_{k+1} and μ_{k+1} to the Lagrange multipliers associated with the constraints $h(x) = 0$ and $g(x) = 0$ that satisfy the same error bound as x_{k+1} are given by the least-squares solutions λ and μ to the linear system

$$\nabla f(x) + \lambda^T \nabla h(x) + \mu^T \nabla g(x) = 0 \tag{10}$$

corresponding to $x = x_{k+1}$.

We now combine Theorems 2.1 and 3.1 to show that quadratic convergence is achieved when the Newton–Armijo iteration is used to generate the y_k of Theorem 3.1.

Corollary 3.1. Under the hypotheses of Theorem 3.1, there exists a neighborhood Δ of (x^*, λ^*) for which the problem

$$\text{minimize}\{L_p(\lambda_k, x): \nabla h(y_k)(x - y_k) = 0, g(x) = 0\}$$

has a local minimizer $x = x_{k+1}$, whenever (x_k, λ_k) lies in Δ, $x_k \in \Omega$, and $y_k = w_I$ for any $I \geq 1$ where the iterates w_i are generated by (4)–(5), starting from $w_0 = x_k$. Moreover, for some constant c, we have

$$|x_{k+1} - x^*| \leq c|\lambda_k - \lambda^*|^2 + c|x_k - x^*|^2,$$

where c is independent of (x_k, λ_k) in $\Delta \cap \Omega \times R^m$.

Proof. Since $I \geq 1$, the last inequality in (8) gives

$$|h(y_k)| = |h(w_I)| = O(|h(x_k)|^2) = O(|x_k - x^*|^2),$$

and by the middle inequality in (8),

$$|y_k - x_k| = O(|h(x_k)|) = O(|x_k - x^*|).$$

By the triangle inequality,

$$|y_k - x^*| \leq |y_k - x_k| + |x_k - x^*| = O(|x_k - x^*|).$$

These inequalities combined with Theorem 3.1 complete the proof. □

4. Kuhn–Tucker Error

In Section 2, we presented an implementation of the Constraint Step for which each successive iteration reduced the constraint violation. Moreover, near a feasible point where the constraint gradients are linearly independent, we have $C(w_{i+1}) = O(C(w_i)^2)$. In this section, we present an iterative implementation of the Kuhn–Tucker Step which is similar to the Newton process in that the Kuhn–Tucker error is locally squared, $K(w_{i+1}, \Lambda_{i+1}) = O(K(w_i, \Lambda_i)^2)$. But unlike the Newton technique, the Kuhn–Tucker error may not decrease monotonically. Nonetheless, a globally convergent update can be formulated; see Section 6.

Let L_p^i be defined by

$$L_p^i(\lambda, x) = f(x) + \lambda^T h(x) + p|h(x) - h(w_i)|^2,$$

and consider the following iteration:

$$w_{i+1} = \arg\min\{L_p^i(\Lambda_i, x): \nabla h(w_i)(x - w_i) = 0, x \in \Omega\}, \tag{11}$$

where

$$\Lambda_i = \arg\min\{K(\Lambda, w_i): \Lambda \in R^m\}.$$

To be consistent with the presentation in the previous sections, it is assumed that Ω has the form (7), so that K can be expressed as

$$K(\lambda, x) = \text{minimum}\{|\nabla_x L(\lambda, x) + \mu^T \nabla g(x)|: \mu \in R^l\}.$$

Theorem 4.1. Under the hypotheses of Theorem 3.1, there exists neighborhoods Δ_1 and Δ_2 of x^* and a constant c such that, for every $w_0 \in \Delta_1 \cap \Omega$, the iteration

$$\Lambda_i = \arg\min\{K(\Lambda, w_i): \Lambda \in R^m\}, \tag{12a}$$

$$w_{i+1} = \arg\min\{L_p^i(\Lambda_i, x): \nabla h(w_i)(x - w_i) = 0, x \in \Delta_2 \cap \Omega\} \tag{12b}$$

converges to a point (Λ, w), and the following properties hold:

(a) $K(\Lambda_{i+1}, w_{i+1}) \leq cK(\Lambda_i, w_i)^2$, for each $i \geq 0$;
(b) $K(\Lambda, w) = 0$;
(c) $|w_i - w| \leq c2^{-2^i}$, for each $i \geq 0$;
(d) $|w_i - w_1| \leq c|w_0 - x^*|^2$, for each $i \geq 1$;
(e) $|w_i - x^*| \leq c|w_0 - x^*|^2 + c|h(w_0)|$, for each $i \geq 1$.

Proof. In order to analyze the iteration (11), we need to consider a perturbed optimization problem,

$$\text{minimize} \quad f(x), \tag{13a}$$

$$\text{subject to} \quad h(x) = z, \qquad g(x) = 0, \tag{13b}$$

where z is a fixed vector in R^m. To show that the Kuhn–Tucker error is squared in each iteration, we employ the lemmas of Appendix C (Section 13). By Lemma 13.1, there exists a local minimizer of (13) and an associated multiplier that depend Lipschitz continuously on z near 0. Defining

$$E_z(\lambda, x) = K(\lambda, x) + |h(x) - z|,$$

Lemma 13.2 implies that, for z near 0, for λ near λ^*, and for x near x^* with $x \in \Omega$, the distance between (λ, x) and a solution-multiplier pair associated with (13) is bounded from below and from above by constant multiples of $E_z(\lambda, x)$.

Given w_i near x^*, let w_i^* denote the local minimizer of (13) near x^* associated with $z = h(w_i)$, and let $\Lambda = \Lambda_i^*$ be the corresponding multiplier for which $K(\Lambda_i^*, w_i^*) = 0$. By Lemma 13.1, w_i^* is near x^* when w_i is near x^*. By Theorem 3.1 and the fact that $x = w_i$ satisfies the constraint $h(x) = h(w_i)$, we have

$$|w_{i+1} - w_i^*| = O(|w_i - w_i^*|^2) + O(|\Lambda_i - \Lambda_i^*|^2). \tag{14}$$

Since Λ_i is the least-squares solution to (10) when $x = w_i$, it follows that

$$|\Lambda_i - \Lambda_i^*| = O(|w_i - w_i^*|).$$

Hence, (14) implies that

$$|\Lambda_{i+1} - \Lambda_i^*| + |w_{i+1} - w_i^*| = O(|w_i - w_i^*|^2). \tag{15}$$

Note that

$$E_z(\Lambda_i, w_i) = K(\Lambda_i, w_i), \qquad \text{when } z = h(w_i).$$

Hence, by Lemma 13.2,

$$|w_i - w_i^*| = O(K(\Lambda_i, w_i)).$$

Also, by Lemma 13.2 with $z = h(w_i)$, we have

$$|\Lambda_{i+1} - \Lambda_i^*| + |w_{i+1} - w_i^*| \geq c_1 E_z(\Lambda_{i+1}, w_{i+1}) \geq c_1 K(\Lambda_{i+1}, w_{i+1}).$$

Combining these relations, we have

$$K(\Lambda_{i+1}, w_{i+1}) = O(K(\Lambda_i, w_i)^2),$$

which establishes (a).

Next, let us show that the w_i converge locally quadratically. Since $h(w_i^*) = h(w_i)$, (15) yields

$$\begin{aligned} |h(w_{i+1}) - h(w_i)| &= |h(w_{i+1}) - h(w_i^*)| = O(|w_{i+1} - w_i^*|) \\ &= O(|w_i - w_i^*|^2). \end{aligned} \tag{16}$$

By Lemma 13.1 and (16),

$$|w_{i+1}^* - w_i^*| = O(|h(w_{i+1}) - h(w_i)|) = O(|w_i - w_i^*|^2). \tag{17}$$

Since

$$|w_i - w_i^*|^2 \leq 2|w_i - w_{i-1}^*|^2 + 2|w_i^* - w_{i-1}^*|^2,$$

(15) and (17) imply that $a_{i+1} = O(a_i^2)$, where

$$a_{i+1} = |w_{i+1} - w_i^*| + |w_{i+1}^* - w_i^*|.$$

In particular, if $a_{i+1} \leq \gamma a_i^2$, then

$$\gamma a_{i+1} \leq [\gamma a_1]^{2^i}, \qquad a_{i+1} \leq a_1 [\gamma a_1]^{2^i - 1}.$$

Combining Lemma 13.1, (15), and (17), it follows that a_1 approaches zero as w_0 approaches x^*. Hence, for w_0 sufficiently close to x^*, there exists a constant c such that $a_i \leq c a_1 2^{-2^i}$. By the triangle inequality, we have

$$\begin{aligned} |w_{i+1} - w_i| &\leq |w_{i+1} - w_i^*| + |w_i^* + w_{i-1}^*| + |w_{i-1}^* - w_i| \\ &\leq a_{i+1} + a_i \leq 2ca_1 2^{-2^i}, \end{aligned} \tag{18}$$

for each $i \geq 1$.

If w denotes the limit of the sequence w_i, then relation (18) yields (c). Since $K(\Lambda_i^*, w_i^*) = 0$, and since both w_i and w_i^* are approaching the same limit w, we conclude that the Λ_i approach a limit Λ and (b) holds. Moreover, (18) implies that $|w_i - w_1| = O(a_1)$. By (15) and (17), $a_1 = O(|w_0 - w_0^*|^2)$. By Lemma 13.1,

$$|w_0^* - x^*| = O(|h(w_0) - h(x^*)|) = O(|w_0 - x^*|), \tag{19a}$$

$$|w_0 - w_0^*| \leq |w_0 - x^*| + |w_0^* - x^*| = O(|w_0 - x^*|). \tag{19b}$$

Hence, $a_1 = O(|w_0 - x^*|^2)$, which establishes (d). By the triangle inequality, (15), and (19), we have

$$|w_1 - x^*| \leq |w_1 - w_0^*| + |w_0^* - x^*| = O(|w_0 - x^*|^2) + O(|w_0^* - x^*|).$$

This inequality combined with (19) and (d) yield (e). □

5. Total Error

Combining Theorem 2.1 and Theorem 4.1, we obtain quadratic convergence of the total error.

Corollary 5.1. Under the hypotheses of Theorem 3.1, there exist neighborhoods Δ_1 and Δ_2 of x^* and a constant c such that

$$|x_{k+1} - x^*| \leq c|x_k - x^*|^2,$$

for every k where the iterate x_{k+1} is generated from x_k, starting from any point $x_0 \in \Delta_1 \cap \Omega$, in the following way. Setting $w_0 = x_k$, we repeat the iteration (4)–(5) any number of times, stopping with w_I, $I \geq 1$; setting $w_0 = w_I$, we repeat the iteration (12) any number of times, stopping with w_J, $J \geq 1$; then, we set $x_{k+1} = w_J$.

Proof. By Theorem 2.1, we have $|h(w_I)| \leq c|h(x_k)|^2$. Combining this with part (e) of Theorem 4.1 yields

$$|w_J - x^*| \leq c|w_I - x^*|^2 + c|h(w_I)| \leq c|w_I - x^*|^2 + O(|h(x_k)|^2). \tag{20}$$

By the middle inequality in (8),

$$|w_I - x_k| \leq c|h(x_k)| = O(|x_k - x^*|).$$

This inequality, the triangle inequality, and (20) complete the proof. □

Recall that our objective is to reduce the total error (constraint violation plus Kuhn–Tucker error) beneath some given error tolerance. If the constraint error is much smaller than the Kuhn–Tucker error, then it is inefficient to spend a lot of effort reducing the constraint error. Similarly, if the Kuhn–Tucker error is much smaller than the constraint error, then it is inefficient to spend a lot of effort solving the optimization problem (11). In Section 2, we presented an iteration that locally reduced the constraint error in successive iterations, and in Section 4 we presented an iteration that locally reduced the Kuhn–Tucker error. These iterations, which are quadratically convergent, can be performed any number of times without interfering with the quadratic convergence property associated with the 2-step process. Hence, we can continue the Armijo–Newton iteration until the constraint error is beneath the Kuhn–Tucker error, and we can continue iteration (11) until the Kuhn–Tucker error is beneath the constraint error. In this way, a balanced reduction in the error is achieved.

6. Global Convergence

We begin with a global convergence result for the iteration (4)–(5).

Lemma 6.1. Suppose that there exists a solution to (5) for each i, there exists a constant M such that $|d_i| \leq M$ for every i, and there exists $\rho > 0$ such that ∇h is Lipschitz continuous with modulus κ in the ball with center w_i and radius ρ for each i. Then for each i where $h(w_i)$ does not vanish, the iterate w_{i+1} defined by (4) exists, and we have

$$|h(w_{i+1})| \leq (1 - \gamma_i \tau)|h(w_i)|,$$

where

$$\gamma_i = \text{minimum}\{1, \sigma\rho/M, 2|h(w_i)|\sigma(1-\tau)/\kappa M^2\}.$$

Hence, either an element of the sequence $h(w_0), h(w_1), \ldots$ vanishes after a finite number of steps, or the entire sequence tends to zero.

Proof. Applying the fundamental theorem of calculus in any neighborhood of a point w where the derivative of h satisfies the Lipschitz condition, we have

$$\begin{aligned} h(w + sd) &= h(w) + \int_0^s \nabla h(w + td)d\, dt \\ &= h(w) + s\nabla h(w)d + \int_0^s (\nabla h(w + td) - \nabla h(w))d\, dt. \end{aligned}$$

Assuming that

$$|d| \leq M \quad \text{and} \quad \nabla h(w)d = -h(w)$$

[see (5)], we conclude that

$$|h(w + sd)| \leq (1 - s)|h(w)| + \kappa s^2 M^2/2. \tag{21}$$

Since Lemma 6.1 holds trivially when $h(w_i)$ vanishes, let us assume that $h(w_i) \neq 0$. In this case, (21) implies that (6) holds for j sufficiently large. From the definition of s_i, it follows that either $s_i = 1$ or

$$|h(w_i + \sigma^{-1} s_i d_i)| \geq (1 - \tau\sigma^{-1} s_i)|h(w_i)|. \tag{22}$$

If $s_i|d_i|/\sigma > \rho$, then

$$s_i > \rho\sigma/|d_i| \geq \rho\sigma/M.$$

Conversely, if $s_i|d_i|/\sigma \leq \rho$, then by (21) with $w = w_i$, $d = d_i$, and $s = s_i/\sigma$, and by (22), we have

$$s_i \geq \sigma(1 - \tau)|h(w_i)|/\kappa M^2.$$

Combining these inequalities yields $s_i \geq \gamma_i$. Inserting the lower bound $s = \gamma_i$ in (6) completes the proof. □

If Ω is compact and w_0 lies in the convex hull of Ω, then the direction vectors d_i are uniformly bounded, and the assumption of Lemma 6.1 that $|d_i| \leq M$ for some M is satisfied. Note though that, except for special values of $h(w_i)$, the linear system $\nabla h(w_i)d = -h(w_i)$ only has a solution when the rows of $\nabla h(w_i)$ are linearly independent.

Unlike the Newton–Armijo iteration, the Kuhn–Tucker iteration (11) may not be globally convergent. One way to achieve a globally convergent iteration is to update the multiplier approximation Λ_i only when the Kuhn–Tucker error decreases by some fixed factor $\sigma < 1$. More precisely, if $\lambda = \lambda_i$ minimizes $K(\lambda, w_i)$ over λ, then the globally convergent algorithm takes the following form: Setting $K_0 = K(\Lambda_0, w_0)$ and $\Lambda_0 = \lambda_0$, we perform the iteration

$$w_{i+1} = \arg\min\{L_p^i(\Lambda_i, x): \nabla h(w_i)(x - w_i) = 0, x \in \Omega\}, \tag{23a}$$

$$\Lambda_{i+1} = \lambda_{i+1} \text{ and } K_{i+1} = K(\lambda_{i+1}, w_{i+1}), \quad \text{if } K(\lambda_{i+1}, w_{i+1}) \leq \sigma K_i, \tag{23b}$$

$$\Lambda_{i+1} = \Lambda_i \text{ and } K_{i+1} = K_i, \quad \text{otherwise.} \tag{23c}$$

In Theorem 4.1, we saw that

$$K(\lambda_{i+1}, w_{i+1}) = O(K(\lambda_i, w_i)^2)$$

in a neighborhood of a local minimizer of (1). Consequently, the condition

$$K(\lambda_{i+1}, w_{i+1}) \leq \sigma K_i$$

is always satisfied in a neighborhood of a local minimizer, and the Kuhn–Tucker error decays quadratically in accordance with Theorem 4.1. Globally, we have the following convergence result.

Lemma 6.2. Suppose that Ω is given by (7), Ω is compact, and f, g, h are continuously differentiable in Ω. Then, a subsequence of the w_i approaches a limit w^*, and if the constraint gradients are linearly independent at w^*, then there exists λ^* such that $K(\lambda^*, w^*) = 0$.

Proof. If the condition $K(\lambda_{i+1}, w_{i+1}) \leq \sigma K_i$ is satisfied an infinite number of times, then the Kuhn–Tucker error tends to zero for a subsequence of the iterations, and the corollary holds trivially. On the other hand, if the condition $K(\lambda_{i+1}, w_{i+1}) > \sigma K_i$ is satisfied for each i sufficiently large, then $\Lambda_{i+1} = \Lambda_i$ for i sufficiently large. We let Λ denote the limit of the Λ_i, and we let w^* denote any convergent subsequence of the w_i with the property that the constraint gradients are linearly independent at w^*. Letting L_p^* be defined by

$$L_p^*(\Lambda, x) = f(x) + \Lambda^T h(x) + p|h(x) - h(w^*)|^2,$$

we consider the optimization problem

$$\text{minimize} \quad L_p^*(\Lambda, x), \tag{24a}$$

$$\text{subject to} \quad \nabla h(w^*)(x - w^*) = 0, \qquad g(x) = 0. \tag{24b}$$

If $x = w^*$ is a local minimizer for this problem, then by the Kuhn–Tucker conditions, there exists λ^* such that $K(\lambda^*, w^*) = 0$. Conversely, suppose that $x = w^*$ is not a local minimizer for (24). We show that this leads to a contradiction.

By the structure of L_p^i, we have

$$L_p^i(\Lambda, w_i) \geq L_p^i(\Lambda, w_{i+1}) \geq L_p^{i+1}(\Lambda, w_{i+1}). \tag{25}$$

Since w^* is not a local minimizer for (24), there exists y near w^* such that

$$L_p^*(\Lambda, y) < L_p^*(\Lambda, w^*),$$

where

$$\nabla h(w^*)(y - w^*) = 0 \quad \text{and} \quad g(y) = 0.$$

Let ϵ be defined by

$$\epsilon = L_p^*(\Lambda, w^*) - L_p^*(\Lambda, y).$$

As a subsequence of the w_i approaches w^*, it follows from the independence of constraint gradients and the implicit function theorem that there exists an associated sequence y_i approaching y with

$$L_p^*(\Lambda, w^*) - L_p^*(\Lambda, y_i) \geq \epsilon/2, \qquad /h(w_i)(y_i - w_i) = 0, \qquad g(y_i) = 0,$$

for i sufficiently large. Hence, for i sufficiently large,

$$L_p^i(\Lambda, w_i) - L_p^i(\Lambda, w_{i+1}) \geq L_p^i(\Lambda, w_i) - L_p^i(\Lambda, y_i) \geq \epsilon/4.$$

This inequality along with (25) violate the fact that L_p^i is bounded from below uniformly. □

As discussed in Section 1, when the Constraint Step followed by the Kuhn–Tucker Step do not decrease the total error, we minimize $L_p(\lambda_k, x)$ over $x \in \Omega$. As discussed in Section 5, computing time is potentially lowered when the error is reduced in a balanced fashion. If x_k minimizes $L_p(\lambda_k, x)$ over $x \in \Omega$, and the Kuhn–Tucker condition holds, then

$$K(\lambda_k + 2ph(x_k), x_k) = 0.$$

Hence, when the optimization problem in the Global Step is solved exactly, the Kuhn–Tucker error is reduced to zero, while the constraint error is

typically positive. To achieve a balanced reduction in the error, we only need to obtain an approximation x_k to a minimizer of $L_p(\lambda_k, x)$ for which

$$K(\lambda_k + 2ph(x_k), x_k) \leq C(x_k).$$

We now establish a global convergence property for this approximation.

Theorem 6.1. Suppose that Ω is given by (7), Ω is compact, and f, g, h are continuously differentiable in Ω. If p_k is a sequence of scalars tending to infinity, λ_k is a uniformly bounded sequence in R^m, and x_k is a sequence in Ω with the property that $K(\lambda_k + 2ph(x_k), x_k) \leq C(x_k)$ for every k, then every subsequence of the x_k that approaches a limit x^* where the constraint gradients are linearly independent has the property that $E(\lambda, x^*)$ vanishes for some $\lambda \in R^m$.

Proof. To simplify notation, let x_k also denote a subsequence of the iterates that approaches a limit x^* where the constraint gradients are linearly independent. By the construction of the x_k, there exists a vector μ_k such that

$$|\nabla_x f(x_k) + v_k^T \nabla h(x_k) + \mu_k^T \nabla g(x_k)| \leq C(x_k),$$

where

$$v_k = \lambda_k + 2p_k h(x_k).$$

Hence, we have

$$|v_k^T \nabla h(x_k) + \mu_k^T \nabla g(x_k)| \leq C(x_k) + |\nabla f(x_k)|.$$

Since Ω is compact and the constraint gradients at x^* are linearly independent, v_k and μ_k are uniformly bounded for k sufficiently large. From the definition of v_k, it follows that

$$h(x_k) = (v_k - \lambda_k)/2p_k.$$

Since λ_k is uniformly bounded, we conclude that $h(x_k)$ tends to zero as the penalty p_k tends to infinity. Let μ_k minimize $K(\lambda, x_k)$ over $\lambda \in R^m$. Since the constraint gradients are linearly independent at x^*, this minimizer is unique for k sufficiently large. The following inequalities complete the proof:

$$K(\mu_k, x_k) \leq K(\lambda_k + 2ph(x_k), x_k) \leq C(x_k) = |h(x_k)|. \qquad \square$$

7. Preconditioning with Rigid Constraints

In the case $\Omega = R^n$, we saw in Ref. 1 that the following preconditioner eliminates the ill conditioning associated with the penalty parameter in the Global Step:

$$H = [S + p\nabla h(x_k)^T \nabla h(x_k)]^{-1}, \tag{26}$$

where S is any symmetric, positive-definite matrix. For illustration, with the Fletcher–Reeves formulation of the conjugate gradient method, each preconditioned iteration has the following structure:

$$x_{i+1} = x_i + \alpha_i d_i, \qquad d_{i+1} = -Hg_i + \beta_i d_i, \tag{27}$$

where $g_i = \nabla_x L_p(\lambda_k, x_i)$, α_i is the stepsize (an efficient stepsize procedure is developed in Ref. 27), and β_i is given by

$$\beta_i = g_{i+1}^T H g_{i+1} / g_i^T H g_i.$$

We now develop an appropriate preconditioner for the minimization of the augmented Lagrangian $L_p(\lambda_k, x)$ subject to the constraint $x \in \Omega$. This preconditioner is obtained by converting the constrained problem to an unconstrained problem for which we know an appropriate preconditioner. Given $x_k \in \Omega$, let O denote an open neighborhood of x_k, and suppose that there exists a map J between a neighborhood of the origin and $O \cap \Omega$ with the property that minimizing $L_p(\lambda_k, J(y))$ over y near the origin is in some sense equivalent to minimizing $L_p(\lambda_k, x)$ over $x \in O \cap \Omega$. If J is differentiable, then by (26) an appropriate preconditioner for the unconstrained problem is

$$H_y = [S + p\nabla J(0) C^T C \nabla J(0)^T]^{-1},$$

where S is any symmetric, positive-definite matrix and $C = \nabla h(x_k)$.

Since linear equalities and inequalities are so important in applications, let us construct a J in the special case where Ω is given by (2). First, some terminology: Given x in Ω, the constraint $x_i \leq u_i$ is called active if $x_i = u_i$. Similarly, the constraint $x_i \leq u_i$ is called inactive if $x_i < u_i$. The same terminology applies to the lower bounds $l_i \leq x_i$. Every constraint associated with the linear system $Ax = b$ is considered active.

Given $x_k \in \Omega$, let N denote a matrix whose columns are an orthonormal basis for the space of vectors orthogonal to the gradients of the active constraints associated with x_k, and let us define $J(y) = x_k + Ny$. For y in a neighborhood of the origin, $x = J(y)$ lies in Ω. At least locally, minimizing a function over $x \in \Omega$, while requiring that the active constraints at x are the same as the active constraints at x_k, is equivalent to an unconstrained minimization over y near the origin. From the discussion above, an appropriate preconditioner for the y-minimization problem is

$$H_y = [S + pN^T C^T C N]^{-1}. \tag{28}$$

By the Sherman–Morrison–Woodbury modification formula (see Ref. 28), (28) can be expressed as

$$H_y = S^{-1} - S^{-1}N^TC^T[p^{-1}I + CNS^{-1}N^TC^T]^{-1}CNS^{-1}.$$

The preconditioner H_y acts on the gradient with respect to y to produce a search direction in the space of vectors y. We now transform from y to x to see how the preconditioner acts on the gradient with respect to x. Since the appropriate preconditioner for the x-minimization problem is $H_x = NH_yN^T$, we have

$$H_x = N(S^{-1} - S^{-1}N^TC^T[p^{-1}I + CNS^{-1}N^TC^T]^{-1}CNS^{-1})N^T.$$

Let P denote the matrix that projects a vector into the space orthogonal to the gradients to the active constraints. Since $P = NN^T$, we see that when $S = I$, the preconditioner reduces to

$$H_x = P - PC^T[p^{-1}I + CPC^T]^{-1}CP, \qquad C = \nabla h(x_k). \tag{29}$$

In summary, the preconditioner (29) has two properties: It maps a vector into the space orthogonal to the gradients of the active constraints, and when it is incorporated in minimization schemes like the conjugate gradient method (27), the ill-conditioning associated with the penalty parameter p is reduced.

8. Numerical Linear Algebra

In this section, we examine the linear algebra associated with the implementation of our algorithm. Throughout this discussion, it is assumed that Ω is given by (2) and the preconditioner (29) is utilized. We focus on the normal equation approach, although there is an alternative approach based on a QR factorization. In the normal equation approach to the linear algebra, the unifying element in the analysis is the symmetric matrix $\bar{B}\bar{B}^T$, where B is the linear constraint matrix A augmented by the gradients of the nonlinear constraints, and $\bar{B}$ is obtained from B by deleting columns associated with the active inequality constraints. Each time that an inequality constraint changes between active and inactive, there is a rank-one change in $\bar{B}\bar{B}^T$. The problem of updating a factorization after a rank-one change has been studied extensively; we refer the reader to Refs. 29–32 by Bartels, Gill, Golub, Murray, and Saunders. Throughout this section, we assume that the rows of $\bar{B}$ are linearly independent.

Observe that the set of indices (i, j) associated with nonzero elements of BB^T contains the set of indices associated with nonzero elements of $\bar{B}\bar{B}^T$. Hence, a storage structure that is suitable for the nonzero elements of BB^T can be used to store the nonzero elements of $\bar{B}\bar{B}^T$, regardless of the choice for the active constraints. Also note that, if B is sparse, then in many cases so is BB^T. Hence, sparse matrix technology (see Ref. 33 or 34) can be employed in both the storage and the factorization of $\bar{B}\bar{B}^T$.

(a) Projections. When solving the quadratic program (5) using an active set strategy, when solving the optimization problem in (23) using the conjugate gradient method, and when utilizing the preconditioner (29), we repeatedly project vectors into the space orthogonal to active constraint gradients. Let B denote the matrix obtained by augmenting the matrix A by $\nabla h(y_k)$,

$$B = \begin{bmatrix} A \\ \nabla h(y_k) \end{bmatrix}.$$

Given a vector q, its projection p into the space perpendicular to the active constraint gradients can be expressed as

$$p = q - B^T\mu - U^T v, \tag{30}$$

where the rows of U are the gradients of the active inequality constraints, and the vectors μ and v are chosen to minimize the Euclidean norm of p. If the ith inequality is active (that is, either $x_i = l_i$ or $x_i = u_i$), then v_i can be chosen so that $p_i = 0$ without affecting any other component of p. Hence, the components of p corresponding to active inequalities are zero. Let $\bar{p}, \bar{q}, \bar{B}$ denote the vectors and matrix obtained from p, q, B by deleting components and columns corresponding to active inequality constraints. The μ that minimizes the norm of p in (30) is the same as the μ that minimizes the norm of $\bar{p}$, defined by

$$\bar{p} = \bar{q} - \bar{B}^T\mu. \tag{31}$$

Since $\bar{p}$ is orthogonal to the rows of $\bar{B}$,

$$\bar{B}\bar{p} = \bar{B}[\bar{q} - \bar{B}^T\mu] = 0.$$

Consequently, μ satisfies the classical normal equation

$$\bar{B}\bar{B}\ \mu = \bar{B}\bar{q}.$$

If the rows of $\bar{B}$ are linearly independent, then the normal equation has a unique solution μ that can be substituted into (31) to obtain $\bar{p}$. The components of p not contained in $\bar{p}$ are identically zero.

(b) Multiplier Estimates. In the implementation of the Kuhn–Tucker Step, we estimate the Lagrange multipliers associated with a given x_k by computing the least-squares solution to an overdetermined linear system (see Ref. 35). More precisely, we compute the least-squares solution to a system of the form

$$q = B^T\mu + U^T v, \qquad q = -\nabla f(x_k)^T. \tag{32}$$

Computing the least-squares solution to (32) is equivalent to minimizing the norm of p in (30). Hence, the multiplier estimates μ and v are a byproduct of the projection discussed in (a).

(c) Implementation of (5). We solve the quadratic programming problem (5) using an active set strategy; see Ref. 36 for an interesting exposition of active set strategies. Active set strategies can be built around the matrix $\bar{B}\bar{B}^T$ and its updates, and the observations in (a) and (b) are applicable. Due to the special form of the cost function in (5), the implementation of an active set strategy can be simplified, and convergence is often very fast since the cost function is perfectly conditioned. Near an optimum, the active constraints associated with the solution to (5) are usually the same as the active constraints associated with the current iterate x_k. Hence, an active set strategy typically converges in one iteration near an optimum. If $\bar{d}$ is the inactive vector obtained from d by deleting the components associated with active constraints, then the inactive vector associated with the solution to (5) is given by the classic formula

$$\bar{d} = \bar{B}^T(\bar{B}\bar{B}^T)^{-1}f, \qquad f = \begin{bmatrix} 0 \\ -h(x_k) \end{bmatrix},$$

which involves the familiar product $\bar{B}\bar{B}^T$.

(d) Preconditioner. Let us consider the preconditioner H_x in (29). Each iteration of the preconditioned conjugate gradient method involves multiplying the gradient vector by this matrix. In the process of multiplying a vector by H_x, we must compute a matrix-vector product involving the inverse of the matrix $R = p^{-1}I + CPC^T$ that appears in the middle of (29). The computation of the projection P was discussed already. In order to multiply a vector by R^{-1} efficiently, we can store R in Cholesky factored form. Given a vector z, the product $y = R^{-1}z$ is obtained by solving the Cholesky factored system $Ry = z$ for the unknown y.

When a constraint becomes either active or inactive, R changes and its Cholesky factorization must be updated. Since the change in R is intimately related to the change in the projection P, let us examine how the projection changes when a constraint becomes either active or inactive.

Theorem 8.1. Given a matrix D, let P be the matrix that projects a vector into the null space of D. Given a vector c that does not lie in the row space of D, let Q be the matrix that projects a vector into the null space of D intersect the orthogonal complement of c. Then, we have

$$Q = P - (1/c^TPc)(Pc)(Pc)^T. \tag{33}$$

Proof. Assuming that the rows of D are linearly independent (if not, remove the redundant rows of D), P has the following classical representation:

$$P = I - D^T(DD^T)^{-1}D. \tag{34}$$

Similarly, if E is the matrix obtained by appending c^T after the last row of D,

$$E = \begin{bmatrix} D \\ c^T \end{bmatrix},$$

then Q can be expressed as

$$Q = I - E^T(EE^T)^{-1}E.$$

By Eq. (5a) in Ref. 28, we have

$$(EE^T)^{-1} = \begin{bmatrix} M & -(DD^T)^{-1}Dc/\alpha \\ -(Dc)^T(DD^T)^{-1}/\alpha & 1/\alpha \end{bmatrix},$$

where $\alpha = c^TPc$ and

$$M = (DD^T)^{-1} + \alpha^{-1}(DD^T)^{-1}Dcc^TD^T(DD^T)^{-1}.$$

Hence, Q can be written as

$$Q = I - D^TMD + \alpha^{-1}[D^T(DD^T)^{-1}Dcc^T + cc^TD^T(DD^T)^{-1}D - cc^T]. \tag{35}$$

By the definition of M, we have

$$D^TMD = D^T(DD^T)^{-1}D + \alpha^{-1}D^T(DD^T)^{-1}Dcc^TD^T(DD^T)^{-1}D. \tag{36}$$

Combining (34), (35), and (36), the proof is complete. □

Theorem 8.1 reveals how the projection matrix P changes when a new constraint of the form $c^Tx = d$ becomes active. On the other hand, when the constraint $c^Tx = d$ becomes inactive, it follows from Theorem 8.1 that the new projection matrix Q satisfies the relation

$$Q = P + (1/c^TQc)(Qc)(Qc)^T. \tag{37}$$

On the surface, this formula seems less useful than (33), since the new projection Q appears on both sides of the equation. Note though that we have already developed an algorithm to compute a projection and to update the related Cholesky factorization. Consequently, we do not use (33) and (37) to evaluate the new projection after a change in the active

constraints; instead, we use these relations to evaluate the rank-one change in the projection. That is, the rank-one change in R is $-(CPc)(CPc)^T/c^TPc$ when a new constraint becomes active, while it is $(CQc)(CQc)^T/c^TQc$ when a constraint becomes inactive. Knowing the rank-one change in R, we can apply one of the algorithms of Bartels, Gill, Golub, Murray, and Saunders to update the Cholesky factorization of R.

9. Subroutine MIN

We now state a specific implementation of the Constraint Step, the Kuhn–Tucker Step, and the Global Step. This implementation is used in the numerical experiments of Section 10. In presenting the algorithm, the subscript k is used for the current big iteration of the algorithm, while the subscript i is used for subiterations within the big loop. The index I or J corresponds to the final subscript in the subiteration. The various constants that appear in the algorithm are somewhat arbitrary; the constants given below are the ones used in the numerical experiments of Section 10. The function $m(x)$ appearing below is the multiplier λ that minimizes $K(\lambda, x)$ over λ.

Step 1. Constraint Step. Set $E_k = E(\lambda_k, x_k)$. Starting with the initial guess $w_0 = x_k$, perform the iteration (4)–(5) taking $\tau = \sigma = 1/2$. If either $i > 40$ or the feasible set in (5) is empty, then branch to the Global Step, after making the update $x_k \leftarrow w_I$. Otherwise, continue the iteration until $C(w_I) \leq K(m(w_I), w_I)$.

Step 2. Kuhn–Tucker Step. Initialize $w_0 = w_I$ and $\Lambda_0 = m(w_I)$ and perform the iteration (23) taking $\sigma = 0.95$. Branch to the Global Step, after making the updates $x_k \leftarrow w_J$ and $\lambda_k \leftarrow \Lambda_J$, in any of the following situations:

(a) $K(\Lambda_J, w_J) \leq 4C(w_J)$ and $E(\Lambda_J, w_J) > 0.95E_k$,
(b) $K(\Lambda_J, w_J) > 4C(w_J) \geq 2E_k$,
(c) $K_J = K_{J-1} = K_{J-2}$.

Otherwise, continue the iteration (23) until

$$K(\Lambda_J, w_J) \leq 4C(w_J) \quad \text{and} \quad E(\Lambda_J, w_J) \leq 0.95E_k;$$

then, increment k, perform the updates $x_k \leftarrow w_J$ and $\lambda_k \leftarrow \Lambda_J$, and branch to the Constraint Step.

Step 3. Global Step. Increase the penalty by the factor 5. Starting with the initial guess x_k, use an iterative method to minimize $L_p(\lambda_k, x)$ over $x \in \Omega$, taking into account the preconditioner (29). Continue the iteration until a point w is generated for which

$$\text{either } K(\lambda_k + 2ph(w), w) \leq C(w) \text{ or } E(m(w), w) \leq 0.6E_k.$$

Increment k, then perform the updates $x_k \leftarrow w$ and $\lambda_k \leftarrow m(w)$, and branch to the Constraint Step.

In our numerical experiments, the optimization problems in the Kuhn–Tucker Step and the Global Step were solved using the conjugate gradient scheme CG developed in Ref. 27. In our implementation of the conjugate gradient scheme, the gradients were always projected into the space orthogonal to the active constraint gradients associated with linear constraints. Rosen's criterion of Ref. 37 was used to decide when to free an active constraint. Whenever an inactive inequality constraint became active, or a constraint was deactivated, the projection matrix was updated and the conjugate gradient iteration was reinitialized in the direction of the negative projected gradient.

A proof that the line search strategy employed in our conjugate gradient algorithm yields n-step local quadratic convergence appears in the thesis by Hirst (Ref. 38). Subroutine CG can be obtained by email from the netlib facility: mail netlib@ornl.gov, send cg from napack. See Chapter 8 of Ref. 39 for a description of the netlib facility developed by Grosse and Dongarra. Note that the Fletcher–Reeves update formula used in CG should be replaced by the Polak–Ribière update (see Ref. 40) in large-scale problems to take advantage of the superior convergence properties of the Polak–Ribière conjugate gradient algorithm in large-scale optimization.

In the Kuhn–Tucker Step, we performed $n - m - l$ conjugate gradient iterations, where m is the number of components of h and l is the current number of active constraints. In the Global Step, we performed $n - l$ preconditioned conjugate gradient iterations before testing whether the current point w satisfied any of the termination criteria. Obviously, in a large-scale optimization problem, $n - m - l$ can be large, and a different criterion is needed for deciding when to terminate the conjugate gradient iteration. In this paper, we do not address the issue of an appropriate termination criterion in the large-scale case.

Reference 41 gives a different way of implementing the Global Step. Loosely speaking, in Ref. 41 the penalty is increased until the constraint violation associated with an approximate minimizer of $L_p(\lambda_k, x)$ over $x \in \Omega$ is smaller than some given tolerance. Then, the tolerance is decreased and

the process is repeated. In our approach, we try to achieve a balanced reduction in the constraint and Kuhn–Tucker errors. We minimize $L_p(\lambda_k, x)$ over $x \in \Omega$ until a point is found with Kuhn–Tucker error smaller than constraint error, then the penalty is increased. Under the independence assumption of Theorem 6.1, this process leads to a reduction in the total error. The idea of computing successively more accurate minimizers to a penalized Lagrangian, then increasing the penalty after some criterion is satisfied, is not new; for example, see Ref. 42 by Polak, where this idea is analyzed in a general framework. What is new is our criterion for deciding when to increase the penalty.

Observe that we branch to the Global Step whenever the quadratic program (5) associated with the Constraint Step is infeasible. An alternative to this way of handling infeasibility is to compute the minimum norm solution to (5) associated with the minimum norm residual; see Ref. 43 by Burke and Han for results relating to this strategy.

In the Kuhn–Tucker Step, there are three different situations where we branch to the Global Step. The motivation behind these conditions is that, after the Constraint Step is complete, the constraint error is often much smaller than the Kuhn–Tucker error. As the Kuhn–Tucker iteration progresses, the constraint error increases and the Kuhn–Tucker error decreases, typically. Whenever it appears likely that further reductions in the Kuhn–Tucker error coupled with increases in the constraint error will not lower the total error, we branch to the Global Step. In case (a), the Kuhn–Tucker error is the same order of magnitude as the constraint error, but the total error is not decreasing at least linearly (by the factor 0.95). In case (b), we abort the subiteration even though the Kuhn–Tucker error is still relatively large since the constraint violation has increased to at least half the total error. In case (c), we abort the subiteration, since the Kuhn–Tucker error is not decreasing very quickly—although the convergence is locally quadratic, the convergence can be slow when the iterates are far from a solution.

Under suitable assumptions, Corollary 5.1 implies that the Constraint Step followed by the Kuhn–Tucker Step are locally quadratically convergent. On the other hand, if for any reason the algorithm branches to the Global Step an infinite number of times, the global convergence result of Theorem 6.1 is applicable.

10. Numerical Experiments

In this section, we investigate how the algorithm of Section 9 performs using some standard test problems. The code that we used in these numerical experiments is by no means the ultimate in efficiency; for

example, we employed a Fletcher–Reeves conjugate gradient update rather than the Polak–Ribière update (see Section 9), the procedure used to update a Cholesky factorization was somewhat inefficient although very stable, and reductions in the penalty parameter were not incorporated in our algorithm. Nonetheless, the numerical results that follow are very encouraging. First, let us consider the test problems that appear in Ref. 44 by Colville. These problems also appear in Ref. 45 by Himmelblau and in Ref. 46 by Hock and Schittkowski. We did not attempt Colville 5, since the cost function in this problem is discontinuous; our computer code assumes that the derivative of the cost is available.

The performance of the algorithm of Section 9 on the Colville test problems appears in Table 1. We present the number of times the cost function is evaluated (NF), the number of times the derivative of the cost is evaluated (NDF), the number of times the constraint function h is evaluated (NH), and the number of times the constraint Jacobian ∇h is evaluated (NDH). To be consistent with Ref. 46, we used the feasible starting guess for Colville 2 and the infeasible starting guess for Colville 3. The iterations were terminated when the total error E was less than or equal to 10^{-6}. The CPU time is given in seconds for a Sun 3/50 computer with Fortran 77 compiler. The optimizing feature of the F77 compiler was not utilized. The relative error appearing in Table 1 is the norm of the difference between the exact solution and the computed solution divided by the norm of the exact solution. For exact solutions, we used those published in Ref. 46. Observe that, in these test problems, the stopping criterion $E \leq 10^{-6}$ generates between 7 and 10 significant digits in the computed solution. In implementing a test problem, nonbound inequality constraints were converted to bound constraints, like those in (2), by introducing slack variables. Hence, when a problem involves nonbound inequality constraints, the effective number of unknowns and equations increases when the constraints are converted to standard form. The data in Table 1 correspond to an initial penalty of 10 in each problem.

Table 1. Results for Colville test problems and subroutine MIN.

Problem	NF	NDF	NH	NDH	CPU time	Relative error
Colville 1	18	10			0.37	5×10^{-8}
Colville 2	311	144	318	144	5.28	1×10^{-7}
Colville 3	9	7	11	7	0.11	2×10^{-10}
Colville 4	58	23			0.12	3×10^{-10}
Colville 6	54	29	56	29	0.46	8×10^{-8}
Colville 7	13	7			0.44	2×10^{-9}
Colville 8	251	110	255	110	5.00	6×10^{-7}

Table 2. Results for Colville test problems using software packages MINOS and NPSOL.

	MINOS			NPSOL		
Problem	NF	CPU time	Relative error	NF	CPU time	Relative error
Colville 1	10	2.44	$<1 \times 10^{-8}$	8	1.81	$<1 \times 10^{-8}$
Colville 2	189	7.62	7×10^{-8}	26	9.39	7×10^{-8}
Colville 3	11	1.62	6×10^{-12}	6	0.45	1×10^{-10}
Colville 4	70	1.65	7×10^{-12}	35	0.92	3×10^{-9}
Colville 6	67	2.19	5×10^{-8}	13	1.41	5×10^{-8}
Colville 7	18	2.80	2×10^{-7}	14	2.61	3×10^{-8}
Colville 8	177	7.30	6×10^{-7}	24	5.97	6×10^{-7}

For comparison, we solved these same test problems using the high-quality software packages MINOS (Ref. 14), Version 5.3, and NPSOL (Ref. 47), November 1990 version. As mentioned in the introduction, MINOS implements the Rosen–Kreuser algorithm using a reduced-gradient algorithm (see Ref. 48) combined with a quasi-Newton method (see Ref. 49) to solve the linearly constrained optimization problems that arise in each iteration. The implementation is described in Refs. 50 and 51. The software package NPSOL implements an SQP-quasi-Newton method. In running these programs, we turned off all printing. In NPSOL, this was accomplished by setting the print levels to 0, while in MINOS we inserted comments in parts of the program dealing with printing. Since both of these programs request function values and gradients simultaneously, Table 2 only reports the number of function evaluations; the number of gradient evaluations is identical and the constraints are evaluated whenever the cost function is evaluated. We did not touch the stopping criterion in either program; the observed relative errors in the computed solutions are given in Table 2. Although the relative errors in Tables 1 and 2 vary slightly, small changes in the stopping criteria in any of the programs will not significantly alter the computational effort on the test problems. The algorithms implemented by either MIN, MINOS, or NPSOL converge rapidly in a neighborhood of an optimum, and the incremental effort needed to reduce the relative error from 10^{-4} to 10^{-8}, for example, is usually a fraction of the effort needed to reduce the initial error to 10^{-4}.

In comparing the data of Tables 1 and 2, we see that subroutines MIN and MINOS tend to require comparable numbers of function and gradient evaluations, while NPSOL tends to require fewer function and gradient evaluations. Of course, the reason for the smaller number of evaluations with NPSOL is that the SQP-quasi-Newton approach essentially

Table 3. Normalized computing times.

Problem	MIN	MINOS	NPSOL
Colville 1	0.15	1.00	0.74
Colville 2	0.56	0.81	1.00
Colville 3	0.07	1.00	0.28
Colville 4	0.07	1.00	0.56
Colville 6	0.21	1.00	0.64
Colville 7	0.16	1.00	0.82
Colville 8	0.68	1.00	0.82

stores more gradient information. On the other hand, the iteration overhead in the SQP approach can be relatively high, since each iteration requires the solution of a quadratic program. Hence, the small number of function and gradient evaluations must be compared to both the iteration overhead and storage requirements to determine which approach is better in any particular application. If the time to evaluate functions and gradients is large, and if the problem dimension is not too big, then the SQP approach is superior. But when the problem size increases, or the relative cost of evaluating functions and gradients decreases, then the conjugate gradient approach employed in subroutine MIN is superior. To compare the efficiency of the three packages on the set of test problems, we divided the computing time by the maximum computing time for each test problem to obtain the normalized times given in Table 3. Observe that subroutine MIN is faster than the other routines for these test problems. On average, MINOS is 1.03 times faster than the slowest routine, NPSOL is 1.41 times faster than the slowest routine, and MIN is 3.67 times faster than the slowest routine.

In the next group of experiments, we investigate the relative importance of various components of subroutine MIN. To study the sensitivity in performance relative to the starting penalty, we resolved the test problems involving nonlinear constraints using three different starting penalties. In Table 4, we see that much of the ill-conditioning due to the penalty has been eliminated by the preconditioner. Nonetheless, as a general rule, small initial penalties yield better performance; the algorithm will increase the penalty if necessary.

To evaluate the significance of the preconditioner (29), we applied the algorithm of the Global Step both with and without a preconditioner. We found that, in almost any nontrivial problem, the preconditioned iterations were much faster than the original iterations. A picture best illustrates the convergence properties. Let us consider the test problem Colville 2. Applying the algorithm of Section 9 to this test problem, the code soon detected convergence slower than linear, and it branched to the Global Step. The

Table 4. Results for Colville test problems using various starting penalties.

Problem	Penalty	NF	NDF	NH	NDH
Colville 2	10	311	144	318	144
	50	753	343	774	343
	250	1242	544	1278	544
Colville 3	10	9	7	11	7
	50	9	7	11	7
	250	9	7	11	7
Colville 6	10	54	29	56	29
	50	74	40	74	40
	250	99	52	102	52
Colville 8	10	251	110	255	110
	50	313	131	315	131
	250	666	300	674	300

optimization problem in the Global Step was solved using the penalty $p = 1250$, and with the choice $S = I$ in the preconditioner. We made two different computer runs. In one run, the gradients were preconditioned using (29). In the other run, there was no preconditioning. The number of conjugate gradient iterations between restarts was equal to the number of free variables (or equivalently, the dimension of the space orthogonal to the active constraint gradients). After each restart, the preconditioner was reevaluated. The convergence is depicted in Fig. 1. The horizontal axis in Fig. 1 is the iteration number. The vertical axis is the logarithm of the value of L_p minus the optimal value $32.348\cdots$ associated with the test problem. The lower curve corresponds to preconditioned iterations while the upper curve corresponds to no preconditioning. After 58 preconditioned iterations, the error E was sufficiently small that the algorithm branched back to the Constraint Step. The dotted line in Fig. 1 corresponds to the value of the cost when the algorithm stopped executing the preconditioned version of the Global Step. With preconditioning, convergence is fairly linear initially; the convergence rate seems to increase near iteration 58. In contrast, without preconditioning, the cost function first decreases quickly as the penalty terms are minimized, but when the penalty terms are comparable to f, the convergence speed is essentially negligible; thousands of iterations will be needed to reach the dotted line. When Hock and Schittkowski reported convergence results for various methods applied to test problem 117 (Colville 2), both of the multiplier method codes failed to

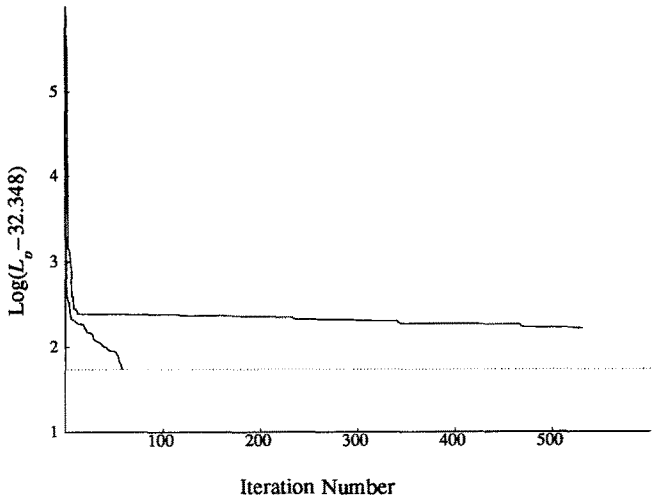

Fig. 1. Convergence of the cost with preconditioning (lower curve) and without preconditioning.

solve the problem; apparently, the convergence was so slow that the codes stopped far from the true solution.

11. Appendix A: Proof of Theorem 2.1

Let $F: R^{m+l} \times R^n \to R^{m+l}$ be defined by

$$F(y, x) = \begin{bmatrix} \nabla h(x)M(x)y + h(x) \\ g(x + M(x)y) \end{bmatrix},$$

where M is the matrix whose columns are the constraint gradients,

$$M(x) = [\nabla h(x)^T \mid \nabla g(x)^T].$$

Evaluating the Jacobian of F with respect to y at $x = x^*$ and $y = 0$ gives

$$\nabla_y F(0, x^*) = M(x^*)^T M(x^*).$$

Since the columns of M are linearly independent, $M(x^*)^T M(x^*)$ is nonsingular. By the implicit function theorem, there exist neighborhoods Δ_1 of 0 and Δ_2 of x^* such that the equation $F(y, x) = 0$ has a unique solution $y = y(x)$ in Δ_1 for every $x \in \Delta_2$, and by Ref. 9 or 52, Δ_2 can be chosen so that

$$|y(x)| \leq \sigma |F(0, x) - F(0, x^*)| \leq \sigma(|h(x)| + |g(x)|),$$

for some constant σ that is independent of $x \in \Delta_2$. Observe that, if $g(x) = 0$, then this bound for $y(x)$ simplifies to

$$|y(x)| \leq \sigma |h(x)|. \tag{38}$$

Let B be a ball with center x^* and with radius so small that $B \subset \Delta_2$, and both h and g are twice continuously differentiable inside B. Defining the parameters

$$\rho = \underset{x \in B}{\text{maximum}}\, \sigma |M(x)|,$$

$$\delta = \left[\sum_{i=1}^{m} \underset{y_j \in B}{\text{maximum}} |\nabla^2 h_j(y_j)|^2 \right]^{1/2},$$

let $w_0 \in B \cap \Omega$ be any point with the property that both

$$\gamma |h(w_0)| \leq \text{minimum}\{1/2, 1 - \tau\}, \qquad \text{where } \gamma = \delta \rho^2/2, \tag{39}$$

and the ball with center w_0 and with radius $R = 2\rho |h(w_0)|$ lies in B.

Suppose that w_{i+1} and w_i lie in B, where w_{i+1} satisfies (4) and (5) with $s_i = 1$. When $h(w_{i+1})$ is expanded in a Taylor series about w_i, the first two terms cancel since

$$\nabla h(w_i) d_i = \nabla h(w_i)(w_{i+1} - w_i) = -h(w_i).$$

Using the intermediate value form for the remainder term, we have

$$h(w_{i+1}) = (1/2) \sum_{j=1}^{m} [d_i^T \nabla^2 h_j(\xi_j) d_i] e_j, \tag{40}$$

where ξ_j lies between w_{i+1} and w_i for each j and e_j is the unit vector with every component equal to zero, except for the jth component which is one. Since $w_i \in B$, (38) yields

$$|y(w_i)| \leq \sigma |h(w_i)|.$$

Since $d = M(w_i) y(w_i)$ satisfies the constraints of (5), the minimum norm solution in (5) is bounded in terms of $M(w_i) y(w_i)$,

$$|d_i| \leq |M(w_i) y(w_i)| \leq \rho |h(w_i)|. \tag{41}$$

Hence, (40) gives

$$|h(w_{i+1})| \leq \gamma |h(w_i)|^2. \tag{42}$$

Consequently, if $w_0, w_1, \ldots, w_{i+1}$ lie in B and (6) holds with $s = 1$ in each iteration, then we have

$$\gamma |h(w_{i+1})| \leq |\gamma h(w_i)|^2 \leq \cdots \leq |\gamma h(w_0)|^{2^{i+1}}. \tag{43}$$

We now show by induction that the entire sequence $w_0, w_1, \ldots$ is contained in B, that

$$|d_i| \leq R2^{-2^i} \tag{44}$$

for every $i \geq 0$, and that (6) holds with $s = 1$ in each iteration. By assumption, w_0 lies in B. Suppose that $w_0, w_1, \ldots, w_i$ lie in B, that (44) holds at iteration $i-1$, and that

$$s_0 = s_1 = \cdots = s_{i-1} = 1.$$

Combining (41), (43), and (39), we have

$$|d_i| \leq \rho|h(w_i)| \leq (\rho/\gamma)(\gamma|h(w_0)|)^{2^i} \leq \rho|h(w_0)|2^{-2^{i+1}} = R2^{-2^i}.$$

Thus, (44) holds at iteration i. Note that $w_i + d_i$ lies in B, since

$$|w_i + d_i - w_0| \leq \sum_{j=0}^{i} |d_j| \leq R \sum_{j=0}^{i} 2^{-2^j} < R. \tag{45}$$

In iteration i, (6) holds with $s = 1$, since

$$|h(w_i + d_i)| \leq \gamma|h(w_i)|^2 \leq (\gamma|h(w_0)|)^{2^i}|h(w_i)| \leq (1-\tau)|h(w_i)|.$$

This completes the induction. By (44), the w_i form a Cauchy sequence that converges to some limit w satisfying the first inequality in (8). The second inequality appears in (45), while the third inequality appears in (42). □

12. Appendix B: Proof of Theorem 3.1

By the Kuhn–Tucker condition characterizing a local minimizer $x = x_{k+1}$ associated with (9), there exist vectors v and μ_{k+1} such that

$$\nabla_x L_p(\lambda_k, x_{k+1}) + v^T \nabla h(y_k) + \mu_{k+1}^T \nabla g(x_{k+1}) = 0. \tag{46}$$

Using the vector v appearing in (46), we construct a new approximation λ_{k+1} to λ^* by the rule

$$\lambda_{k+1} = \lambda_k + v + 2ph(x_{k+1}). \tag{47}$$

Solving for v in (47) and substituting in (46) yields

$$\nabla f(x_{k+1}) + \lambda_{k+1}^T \nabla h(y_k) + [\lambda_k + 2ph(x_{k+1})]^T [\nabla h(x_{k+1}) - \nabla h(y_k)] + \mu_{k+1}^T \nabla g(x_{k+1}) = 0. \tag{48}$$

Since x_{k+1} satisfies the constraints of (9), we have

$$\nabla h(y_k)(x_{k+1} - y_k) = 0, \qquad g(x_{k+1}) = 0. \tag{49}$$

Defining the function $F: R^{n+m+l} \times R^{n+m} \to R^{n+m+l}$ by

$$F(x, \lambda, \mu, y, \eta) = \begin{bmatrix} \nabla f(x) + \lambda^T \nabla h(y) + [\eta + 2ph(x)]^T [\nabla h(x) - \nabla h(y)] + \mu^T \nabla g(x) \\ \nabla h(y)(x - y) \\ g(x) \end{bmatrix},$$

(48)–(49) can be written as

$$F(x_{k+1}, \lambda_{k+1}, \mu_{k+1}, y_k, \lambda_k) = 0.$$

The Jacobian of F with respect to its first three arguments evaluated at the star-variables is

$$\nabla_{x, \lambda, \mu} F(x^*, \lambda^*, \mu^*, x^*, \lambda^*) = \begin{bmatrix} \nabla_x^2 M_p(\lambda^*, \mu^*, x^*) & \nabla h(x^*)^T & \nabla g(x^*)^T \\ \nabla h(x^*) & 0 & 0 \\ \nabla g(x^*) & 0 & 0 \end{bmatrix}, \tag{50}$$

where

$$M_p(\lambda, \mu, x) = L_p(\lambda, x) + \mu^T g(x).$$

Since the constraint gradients are linearly independent and $\nabla_x^2 M_p(\lambda^*, \mu^*, x^*)$ is positive definite in the null space of $\nabla h(x^*)$ intersect the null space of $\nabla g(x^*)$, we have the following well-known result (see Ref. 8, Lemma 1.27): The Jacobian (50) is nonsingular for every $p \geq 0$. By the implicit function theorem, there exist neighborhoods Δ_1 and Δ_2 of (x^*, λ^*) such that the equation $F(x, \lambda, \mu, y, \eta) = 0$ has a unique solution

$$(x, \lambda, \mu) = (x(y, \eta), \lambda(y, \eta), \mu(y, \eta))$$

in Δ_1 for every (y, η) in Δ_2. Moreover, by Ref. 9, Corollary 6.2 or Ref. 52, Δ_2 can be chosen so that

$$\begin{aligned} &|x(y, \eta) - x^*| + |\lambda(y, \eta) - \lambda^*| + |\mu(y, \eta) - \mu|^* \\ &\leq c|F(x^*, \lambda^*, \mu^*, y, \eta) - F(x^*, \lambda^*, \mu^*, x^*, \lambda^*)|, \end{aligned} \tag{51}$$

for some constant c which is independent of $(y, \eta) \in \Delta_2$. By Ref. 9, Lemma 6.5, and by the second-order sufficiency condition, $x(y_k, \lambda_k)$ is the unique local minimizer of (9) near x^* when (y_k, λ_k) is near (x^*, λ^*). From the definition of F, we have

$$F(x^*, \lambda^*, \mu^*, y, \eta) - F(x^*, \lambda^*, \mu^*, x^*, \lambda^*) = \begin{bmatrix} [\nabla h(x^*) - \nabla h(y)]^T (\eta - \lambda^*) \\ \nabla h(y)(x^* - y) \\ 0 \end{bmatrix}, \tag{52}$$

provided $g(y) = 0$. In Ref. 1, Theorem 3.1, we establish the bounds

$$2|[\nabla h(x^*) - \nabla h(y)]^T(\eta - \lambda^*)| \leq |\eta - \lambda^*|^2 + \delta^2|y - x^*|^2,$$

$$|\nabla h(y)(x^* - y)| \leq (\delta/2)|y - x^*|^2 + |h(y)|,$$

where

$$\delta = \left[\sum_{i=1}^{m} \underset{z_i \in [y, x^*]}{\text{maximum}} |\nabla^2 h_i(z_i)|^2\right]^{1/2}.$$

Here, $[y, x^*]$ denotes the line segment between y and x^*. Combining these bounds with (51) and (52), the proof is complete. □

13. Appendix C: Some Sensitivity Results

In this appendix, we state two results that are needed for the analysis of Kuhn–Tucker error in Section 4. Let us consider the optimization problem

$$\text{minimize} \quad f(x), \tag{53a}$$

$$\text{subject to} \quad h(x) = z, \tag{53b}$$

where z is a fixed vector in R^m. Suppose that (53) has a local minimizer x^* corresponding to $z = z^*$, that f and h are twice continuously differentiable in a neighborhood of x^*, and that the constraint gradients $\nabla h_i(x^*)$ are linearly independent. Letting $\lambda = \lambda^*$ denote the unique solution to the equation

$$\nabla f(x) + \lambda^T \nabla h(x) = 0, \tag{54}$$

corresponding to $x = x^*$, we assume that the second-order sufficiency condition holds. That is, the Hessian

$$\nabla_x^2[\, f(x) + h(x)^T \lambda^*]_{x = x^*}$$

is positive definite in the null space of $\nabla h(x^*)$. For a proof of the following result, see Ref. 52.

Lemma 13.1 If the second-order sufficiency condition holds and the constraint gradients are linearly independent at a local minimizer x^* of (53) associated with $z = z^*$, then there exists a neighborhood Δ of z^* such that, for every $z \in \Delta$, (53) has a local minimizer $x(z)$ and an associated multiplier $\lambda(z)$ such that $x(z^*) = x^*$, and $\lambda = \lambda(z)$, $x = x(z)$ satisfy (54). Moreover, there exists a constant c, such that

$$|x(z_1) - x(z_2)| + |\lambda(z_1) - \lambda(z_2)| \leq c|z_1 - z_2|,$$

for every z_1 and z_2 in Δ.

The proof of the following result is analogous to that of Proposition 5.2 in Ref. 1.

Lemma 13.2 Suppose that the assumptions of Lemma 13.1 hold. Let Δ denote the neighborhood of z^* given in Lemma 13.1, and define

$$E_z(\lambda, x) = |\nabla f(x) + \lambda^T \nabla h(x)| + |h(x) - z|.$$

Then, there exists a neighborhood Δ_1 of (x^*, λ^*), a neighborhood Δ_2 of z^* with $\Delta_2 \subset \Delta$, and positive constants c_1 and c_2 such that

$$c_1 E_z(\lambda, x) \leq |x - x(z)| + |\lambda - \lambda(z)| \leq c_2 E_z(\lambda, x),$$

for every $z \in \Delta_2$ and $(x, \lambda) \in \Delta_1$.

References

1. HAGER, W. W., *Dual Techniques for Constrained Optimization*, Journal of Optimization Theory and Applications, Vol. 55, pp. 37–71, 1987.
2. ARROW, K. J., and SOLOW, R. M., *Gradient Methods for Constrained Maxima, with Weakened Assumptions*, Studies in Linear and Nonlinear Programming, Edited by K. Arrow, L. Hurwicz, and H. Uzawa. Stanford University Press, Stanford, California, 1958.
3. HESTENES, M. R., *Multiplier and Gradient Methods*, Journal of Optimization Theory and Applications, Vol. 4, pp. 303–320, 1969.
4. POWELL, M. J. D., *A Method for Nonlinear Constraints in Minimization Problems*, Optimization, Edited by R. Fletcher, Academic Press, New York, New York, pp. 283–298, 1972.
5. ROCKAFELLAR, R. T., *The Multiplier Method of Hestenes and Powell Applied to Convex Programming*, Journal of Optimization Theory and Applications, Vol. 12, pp. 555–562, 1973.
6. ROCKAFELLAR, R. T., *A Dual Approach to Solving Nonlinear Programming Problems by Unconstrained Optimization*, Mathematical Programming, Vol. 5, pp. 354–373, 1973.
7. ROCKAFELLAR, R. T., *Augmented Lagrange Multiplier Functions and Duality in Nonconvex Programming*, SIAM Journal on Control, Vol. 12, pp. 268–285, 1974.
8. BERTSEKAS, D. P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, New York, 1982.
9. HAGER, W. W., *Approximations to the Multiplier Method*, SIAM Journal on Numerical Analysis, Vol. 22, pp. 16–46, 1985.

10. POLYAK, B. T., and TRET'YAKOV, N. V., *The Method of Penalty Estimates for Conditional Extremum Problems*, Akademija Nauk SSSR, Žurnal Vyčislitel'noi Matematiki i Matematičeskoi Fiziki, Vol. 13, pp. 34–46, 1973.
11. FORTIN, R., and GLOWINSKI, R., *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*, North-Holland, Amsterdam, Holland, 1983.
12. ROSEN, J. B., and KREUSER, J. L., *A Gradient Projection Algorithm for Nonlinear Constraints*, Numerical Methods for Nonlinear Optimization, Edited by F. A. Lootsma, Academic Press, London, England, pp. 297–300, 1972.
13. ROBINSON, S. M., *A Quadratically-Convergent Algorithm for General Nonlinear Programming Problems*, Mathematical Programming, Vol. 3, pp. 145–156, 1972.
14. MURTAGH, B. A., and SAUNDERS, M. A., *MINOS 5.0 User's Guide*, Report 83-20R, Department of Operations Research, Stanford University, Stanford, California, 1987.
15. COLEMAN, T. F., and CONN, A. R., *Nonlinear Programming via an Exact Penalty Function: Asymptotic Analysis*, Mathematical Programming, Vol. 24, pp. 123–136, 1982.
16. COLEMAN, T. F., and CONN, A. R., *Nonlinear Programming via an Exact Penalty Function: Global Analysis*, Mathematical Programming, Vol. 24, pp. 137–161, 1982.
17. ARMIJO, L., *Minimization of Functions Having Lipschitz Continuous First-Partial Derivatives*, Pacific Journal of Mathematics, Vol. 16, pp. 1–3, 1966.
18. BURKE, J., and HAN, S. P., *A Gauss–Newton Approach to Solving Generalized Inequalities*, Mathematics of Operations Research, Vol. 11, pp. 632–643, 1986.
19. DANIEL, J. W., *Newton's Method for Nonlinear Inequalities*, Numerische Mathematik, Vol. 21, pp. 381–387, 1973.
20. GARDIA-PALOMARES, U., and RESTUCCIA, A., *Application of the Armijo Stepsize Rule to the Solution of a Nonlinear System of Equalities and Inequalities*, Journal of Optimization Theory and Applications, Vol. 41, pp. 405–415, 1983.
21. PSHENICHNYI, B. N., *Newton's Method for the Solution of Systems of Equalities and Inequalities*, Mathematical Notes of the Academy of Sciences of the USSR, Vol. 8, pp. 827–830, 1970.
22. ROBINSON, S. M., *Extension of Newton's Method to Nonlinear Functions with Values in a Cone*, Numerische Mathematik, Vol. 19, pp. 341–347, 1972.
23. VALENTINE, F. A., *The Problem of Lagrange with Differential Inequalities as Added Side Conditions*, Contributions to the Calculus of Variations, University of Chicago Press, Chicago, Illinois, pp. 407–448, 1937.
24. ORTEGA, J. M., and RHEINBOLDT, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, New York, 1970.
25. BURKE, J. V., *A Sequential Quadratic Programming Method for Potentially Infeasible Mathematical Programs*, Journal of Mathematical Analysis and Applications, Vol. 139, pp. 319–351, 1989.
26. LUENBERGER, D. G., *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, Massachusetts, 1973.

27. HAGER, W. W., *A Derivative-Based Bracketing Scheme for Univariate Minimization and the Conjugate Gradient Method*, Computers and Mathematics with Applications, Vol. 18, pp. 779–795, 1989.
28. HAGER, W. W., *Updating the Inverse of a Matrix*, SIAM Review, Vol. 31, pp. 221–239, 1989.
29. BARTELS, R. H., GOLUB, G. H., and SAUNDERS, M. A., *Numerical Techniques in Mathematical Programming*, Nonlinear Programming, Edited by R. B. Rosen, O. L. Mangasarian, and K. Ritter, Academic Press, New York, New York, pp. 123–176, 1970.
30. GILL, P. E., GOLUB, G. H., MURRAY, W., and SAUNDERS, M. A., *Methods for Modifying Matrix Factorizations*, Mathematics of Computation, Vol. 28, pp. 505–535, 1974.
31. GILL, P. E., and MURRAY, W., *Modification of Matrix Factorizations after a Rank-One Change*, The State of the Art in Numerical Analysis, Edited by D. Jacobs, Academic Press, New York, New York, pp. 55–83, 1977.
32. GILL, P. E., MURRAY, W., and SAUNDERS, M. A., *Methods for Computing and Modifying the LDV Factors of a Matrix*, Mathematics of Computation, Vol. 29, pp. 1051–1077, 1975.
33. DUFF, I. S., ERISMAN, A. M., and REID, J. K., *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, England, 1986.
34. GEORGE, A., and LIU, J. W. H., *Computer Solution of Large Sparse Positive-Definite Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.
35. TAPIA, R. A., *Diagonalized Multiplier Methods and Quasi-Newton Methods for Constrained Optimization*, Journal of Optimization Theory and Applications, Vol. 22, pp. 135–194, 1977.
36. DJANG, A., *Algorithmic Equivalence in Quadratic Programming*, PhD Dissertation, Department of Operations Research, Stanford University, Stanford, California, 1979.
37. DU, D. Z., and ZHANG, X. S., *Global Convergence of Rosen's Gradient Projection Method*, Mathematical Programming, Vol. 44, pp. 357–366, 1989.
38. HIRST, H. P., *N-Step Quadratic Convergence in the Conjugate Gradient Method*, PhD Dissertation, Pennsylvania State University, University Park, Pennsylvania, 1989.
39. HAGER, W. W., *Applied Numerical Linear Algebra*, Prentice-Hall, Englewood Cliffs, New Jersey, 1988.
40. POLAK, E., and RIBIÈRE, G., *Note sur la Convergence de Methodes de Directions Conjugées*, Revue Française d'Informatique et Recherche Opérationnelle, Vol. 16, pp. 35–43, 1969.
41. CONN, A. R., GOULD, N. I. M., and TOINT, P. L., *A Globally Convergent Augmented Lagrangian Algorithm for Optimization with General Constraints and Simple Bounds*, SIAM Journal on Numerical Analysis, Vol. 28, pp. 545–572, 1991.
42. POLAK, E., *On the Global Stabilization of Locally Convergent Algorithms*, Automatica, Vol. 12, pp. 337–342, 1976.
43. BURKE, J., and HAN, S. P., *A Robust Sequential Quadratic Programming Method*, Mathematical Programming, Vol. 43, pp. 277–303, 1989.

44. COLVILLE, A. R., *A Comparative Study on Nonlinear Programming Codes*, Report No. 320-2949, IBM New York Scientific Center, 1968.
45. HIMMELBLAU, D. M., *Applied Nonlinear Programming*, McGraw-Hill, New York, New York, 1972.
46. HOCK, W., and SCHITTKOWSKI, K., *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, Berlin, Germany, 1980.
47. GILL, P. E., MURRAY, W., SAUNDERS, M. A., and WRIGHT, M. H., *User's Guide for NPSOL (Version 4.0): A Fortran Package for Nonlinear Programming*, Report 86-2, Department of Operations Research, Stanford University, Stanford, California, 1986.
48. WOLFE, P., *Methods of Nonlinear Programming*, Nonlinear Programming, Edited by J. Abadie, Interscience, John Wiley, New York, New York, pp. 97–131, 1967.
49. DAVIDON, W. C., *Variable Metric Method for Minimization*, SIAM Journal on Optimization, Vol. 1, pp. 1–17, 1991.
50. MURTAGH, B. A., and SAUNDERS, M. A., *Large-Scale Linearly Constrained Optimization*, Mathematical Programming, Vol. 14, pp. 41–72, 1978.
51. MURTAGH, B. A., and SAUNDERS, M. A., *A Projected Lagrangian Algorithm and Its Implementation for Sparse Nonlinear Constraints*, Mathematical Programming Study, Vol. 16, pp. 84–117, 1982.
52. ROBINSON, S. M., *Strongly Regular Generalized Equations*, Mathematics of Operations Research, Vol. 5, pp. 43–62, 1980.