SIAM J. OPTIMIZATION
Vol. 6, No. 4, pp. 1040–1058, November 1996

A REFLECTIVE NEWTON METHOD FOR MINIMIZING A QUADRATIC FUNCTION SUBJECT TO BOUNDS ON SOME OF THE VARIABLES*

THOMAS F. COLEMAN† AND YUYING LI†

Abstract. We propose a new algorithm, a reflective Newton method, for the minimization of a quadratic function of many variables subject to upper and lower bounds on some of the variables. The method applies to a general (indefinite) quadratic function for which a local minimizer subject to bounds is required and is particularly suitable for the large-scale problem. Our new method exhibits strong convergence properties and global and second-order convergence and appears to have significant practical potential. Strictly feasible points are generated. We provide experimental results on moderately large and sparse problems based on both sparse Cholesky and preconditioned conjugate gradient linear solvers.

Key words. interior Newton method, interior-point method, quadratic programming

AMS subject classifications. 65K05, 90C20, 90C06

1. Introduction.

In this paper we propose a new algorithm for solving the box-constrained quadratic programming problem

$$\min_x \left\{ q(x) \overset{def}{=} c^T x + \frac{1}{2} x^T H x : \quad l \le x \le u \right\}. \tag{1.1}$$

The matrix H is symmetric and, in general, indefinite; $l \in \{\Re \cup \{-\infty\}\}^n$, $u \in \{\Re \cup \{\infty\}\}^n$, $l < u$. We denote the feasible region $\mathcal{F} = \{x : \ l \le x \le u\}$ and the strict interior, $\text{int}(\mathcal{F}) = \{x : \ l < x < u\}$. When H is indefinite we are interested in locating a local minimizer.

The purpose of this paper is to motivate and describe our new approach, the reflective Newton method, and to present the results of computational experiments designed to investigate practical viability. A companion report [9] proffers supporting theory and analysis.

Problem (1.1) arises as a subproblem when minimizing general nonlinear functions subject to bounds, and as a problem in its own right. The box-constrained quadratic programming problem represents an important class of optimization problems and has been the subject of considerable recent work (e.g., [1, 4, 11, 14, 13, 17, 19, 20, 22, 24, 26, 27, 30]). A special subclass deserves mention: the box-constrained least-squares problem

$$\min_x \{\|Ax - b\|_2 : \quad l \le x \le u\}, \tag{1.2}$$

where A is a rectangular m-by-n matrix with $m > n$. Our proposed algorithm can of course be applied to this special case if we use $H = A^T A$ and $c = -A^T b$; moreover, it

* Received by the editors August 16, 1994; accepted for publication (in revised form) June 6, 1995. This research was partially supported by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under grant DE-FG02-86ER25013.A000, by the Computational Mathematics Program of the National Science Foundation under grant DMS-8706133, and by the Advanced Computing Research Institute, a unit of the Cornell Theory Center which receives major funding from the National Science Foundation and IBM Corporation, with additional support from New York State and members of its Corporate Research Institute.

† Computer Science Department and Center for Applied Mathematics, Cornell University, Ithaca, NY 14850 (coleman@cs.cornell.edu).

is possible to implement a conjugate gradient version of our method without explicitly forming $H = A^T A$.

We propose a new approach, a reflective Newton algorithm. The algorithm generates a sequence of strictly feasible iterates, $\{x_k\}$, which converges under standard assumptions to a local solution of (1.1), x_*, at a quadratic convergence rate. Coleman and Li [9] establish theoretical properties of the reflective Newton approach applied to the general nonlinear box-constrained problem—these results apply directly to the reflective Newton procedure proposed here for the quadratic minimization problem (1.1).

This new approach has potential advantages over existing methods. The main advantage appears to be the ability to achieve accurate solutions to large problems in relatively few iterations. The global and quadratic convergence properties established in [9] account, in some measure, for this good behaviour. Our numerical results, reported in §4, support the notion that for a given problem class, the number of required iterations increases only modestly with problem size. The numerical results also indicate a remarkable insensitivity to problem conditioning and the degree of near degeneracy. This is partially explained by the choice of scaling matrix inherent in our approach; we discuss this subsequently.

The sequence $\{x_k\}$ generated by the algorithm is strictly feasible; therefore, the algorithm can be regarded as an "interior-point" algorithm. However, this may be a misleading classification. The algorithm differs markedly from methods commonly referred to as "interior-point" algorithms. For example, the proposed algorithm does not use a barrier function to help ensure strict feasibility. The algorithm generates descent directions for $q(x)$ and then follows a piecewise linear path, reflecting off constraints as they are encountered. Most interior-point methods, on the other hand, generate descent directions (for some function) and then restrict the step, along this straight-line direction, to ensure strict feasibility. The proposed algorithm does not require the delicate choice or adjustment of a penalty or barrier parameter—no penalty or barrier function is used—nor does it require that a "central path" be followed. Indeed the proposed method is robust with respect to starting point.

The algorithm most similar to our current proposal is probably the recent method due to Coleman and Hulbert [5]. (There is also a strong connection to previous work by Coleman and Li [6, 7, 8] and Li [21] on various norm minimization problems.) Both are driven by the nonlinear system of equations representing first-order optimality conditions. Both methods require piecewise quadratic minimization. The methods differ in that our new algorithm is more general: positive definiteness of the symmetric matrix H is not required and it is not necessary to have finite upper and lower bounds on all the variables—the Coleman–Hulbert method requires both of these restrictive properties. Finally, the Coleman–Hulbert method is an exterior-point method, requiring strict decrease in a piecewise quadratic "dual" function, whereas the new method generates feasible iterates, requiring strict decrease in the original quadratic function q.

We conclude the introduction with a short review of optimality conditions for problem (1.1). Define $D = D(x)$ to be a diagonal matrix with the ith diagonal component equal to $|v_i(x)|^{\frac{1}{2}}$. Vector $v_i(x)$ is defined in Figure 1.1, where

$$g(x) \overset{def}{=} \nabla q(x) = Hx + c.$$

How to compute $v(x)$:

- If $g_i < 0$ and $u_i < \infty$ then $v_i = x_i - u_i$.
- If $g_i \geq 0$ and $l_i > -\infty$ then $v_i = x_i - l_i$.
- If $g_i < 0$ and $u_i = \infty$ then $v_i = -1$.
- If $g_i \geq 0$ and $l_i = -\infty$ then $v_i = 1$.

FIG. 1.1. *Definition of* $v(x)$.

The first-order optimality conditions can now be written. If a feasible point x_* is a local minimizer of (1.1) then

$$D_*^2 g_* = 0. \tag{1.3}$$

Let $Free_*$ denote the set of indices corresponding to "free" variables at point x_*:

$$Free_* = \{i : l_i < (x_*)_i < u_i\}.$$

Second-order necessary conditions can be written.[1] If a feasible point x_* is a local minimizer of (1.1) then $D_*^2 g_* = 0$ and $H^{Free_*} \geq 0$, where H^{Free_*} is the submatrix of H corresponding to the index set $Free_*$.

These conditions are necessary but not sufficient. To state practical sufficiency conditions we first need a definition of degeneracy.

DEFINITION 1.1. *A point* $x \in \Re^n$ *is* nondegenerate *if for each index i*

$$g_i = 0 \implies l_i < x_i < u_i.$$

With this definition we can state *second-order sufficiency conditions.* If a nondegenerate feasible point x_* satisfies $D_*^2 g_* = 0$ and $H^{Free_*} > 0$, then x_* is a local minimizer of (1.1).

2. Motivation.

In this section we motivate and explain the basis of the reflective Newton method. Additional intuition can be gleaned from [9] where greater attention is paid to the theoretical underpinnings. We begin at the end.

2.1. The end game.

As indicated by (1.3), a local solution to (1.1) is a zero of the nonlinear system

$$D^2(x)g(x) = 0, \tag{2.1}$$

where $g(x) \overset{def}{=} Hx + c$. The system (2.1) is nondifferentiable when either some $v_i = 0$ or $g_i = 0$. Let x_* be a nondegenerate feasible point satisfying the second-order sufficiency conditions. If a feasible point x is sufficiently close to x_* then a Newton step, $s^N(x)$, can be defined with respect to system (2.1):

$$s^N(x) \;=\; -(D^2H + JD^g)^{-1}D^2g, \tag{2.2}$$

where $D^g = D^g(x) \overset{def}{=} \text{diag}(|g|)$. Each diagonal element of the diagonal Jacobian matrix J is defined as follows: If v_i equals either $x_i - u_i$ or $x_i - l_i$, then $J_{ii} = 1$; otherwise, $J_{ii} = 0$. In [9] it is established that the damped local Newton sequence,

$$x_{k+1} = x_k + \alpha_k s_k^N \tag{2.3}$$

[1] Notation: If a matrix A is a symmetric matrix then we write $A > 0$ to mean A is positive definite; $A \geq 0$ means A is positive semidefinite.

is a locally quadratically convergent sequence provided $\{x_k\}$ is strictly feasible and

$$|1-\alpha_k| = O(\|x_k - x_*\|). \tag{2.4}$$

A crucial observation, proven in [9], is that the distance to the boundary from x_k, along s_k^N, satisfies (2.4). In particular, for each $j \notin Free_*$, $|1-(\beta_k^N)_j| = O(\|x_k-x_*\|)$, where $(\beta_k^N)_j = \frac{|(v_k(x))_j|}{|(s_k^N(x))_j|}$. The positive number $(\beta_k^N)_j$ is the distance from x_k to the approaching bound, one of u_j, l_j, in the direction s_k^N. Also, for $j \in Free_*$, $(\beta_k^N)_j \to \infty$ as $x_k \to x_*$. Therefore, it is possible to choose α_k to satisfy (2.4) and produce feasible iterates via (2.3). For example, let $(\beta_k^N)_{\min} = \min\{(\beta_k^N)_j\}$ and set $\alpha_k = (\beta_k^N)_{\min} - \epsilon_k$, where ϵ_k is a positive number satisfying $\epsilon_k \le \chi_\alpha \|D_k g_k\|$ for any positive constant χ_α. It immediately follows that α_k satisfies (2.4) and that $\{x_k\}$, generated by (2.3), is strictly feasible. Local and quadratic convergence ensues [9].

We maintain strict feasibility $l < x_k < u$ at each iteration. Hence, a good way to compute s_k^N is to first form $\bar{M}_k = D_k H_k D_k + J_k D_k^g$, then solve

$$\bar{M}_k \bar{s}_k = -D_k g_k, \tag{2.5}$$

and finally set $s_k = D_k \bar{s}_k$. Notice that $\bar{M}_k$ is symmetric; moreover, as shown in [9], $\bar{M}_k$ is positive definite and bounded in a feasible neighbourhood of x_*. Furthermore, it can be argued that the conditioning of $\bar{M}_k$ in a feasible neighbourhood of x_* reflects the conditioning of the original problem (1.1) around x_*.

So we have all the ingredients necessary to construct a robust, interior, local, and quadratic Newton process for (1.1).

Algorithm 1

```
For $k = 1 : \infty$
    1. Form $\bar{M}_k = D_k H_k D_k + J_k D_k^g$.
    2. Solve $\bar{M}_k \bar{s}_k^N = -D_k g_k$. Set $s_k^N = D_k \bar{s}_k^N$.
    3. Compute $\beta_{k_{min}}^N$; set $\alpha_k = \beta_{k_{min}}^N - \epsilon_k$, where $\epsilon_k \le \chi_\alpha \|D_k g_k\|$.
    4. Set $x_{k+1} = x_k + \alpha_k s_k^N$.
```

FIG. 2.1. *A local (strictly) feasible Newton method.*

Algorithm 1 (Figure 2.1) represents a practical interior Newton process. However, it is clear that this is a local procedure only; the challenge is to extend this approach to a global method.

2.2. The middle game. Our objective is to devise a global method that smoothly evolves into Algorithm 1, the fast local procedure discussed above. A common globalization technique is to generate a descent direction at the current point for a suitable "merit" function and then take a step along this direction. The most natural merit function for a strictly feasible algorithm is the objective function itself, $q(x) = c^T x + \frac{1}{2} x^T H x$. It is easy to see that the Newton step s_k^N, defined by (2.5) or (2.2), is a descent direction for q in a feasible neighbourhood of a nondegenerate second-order point x_*. The matrix $\bar{M}(x)$ is positive definite in a feasible neighbourhood of x_*. Moreover, in [9] we show that the stepsize $\alpha = 1$ yields sufficient decrease of $q(x)$ in a feasible neighbourhood of x_*. It follows that the calculation of the step length α described in Algorithm 1 also yields sufficient decrease of $q(x)$ in a feasible neighbourhood of x_*. However, when $\|x - x_*\|$ is large, $\bar{M}(x)$ is not necessarily positive definite, and $s^N(x)$ is not necessarily a descent direction for q at x. Indeed, $\bar{M}(x)$ may be singular and therefore $s^N(x)$ is not necessarily well defined at points remote from x_*.

Our solution to this problem is founded on a highly successful technique used in unconstrained nonlinear minimization: minimize a quadratic approximation subject to an elliptical constraint. For example, a descent direction s_k can be determined by solving

$$\min_s \left\{ s^T g_k + \frac{1}{2} s^T B_k s : \; \|S_k^{-1} s\|_2 \le \Delta_k \right\} \tag{2.6}$$

or, more generally,

$$\min_s \left\{ s^T g_k + \frac{1}{2} s^T B_k s : \; \|S_k^{-1} s\|_2 \le \Delta_k, \; s \in \mathcal{S}_k \right\}, \tag{2.7}$$

where $\mathcal{S}_k$ is a subspace of $\mathcal{R}^n$, S_k is a positive diagonal scaling matrix, B_k is symmetric, and $\Delta_k > 0$. Appropriate choices of B_k, S_k, and $\mathcal{S}_k$ yield a descent direction for $q(x)$ at x_k. However, a most interesting question is how to choose these quantities to enable a smooth tie-in to Algorithm 1, the local procedure above. Appropriate choices may surprise. Choose $S = D$ and $B = M = [H + JD^g D^{-2}]$. (Recall that D is a diagonal matrix with the ith diagonal element defined to be $|v_i(x)^{\frac{1}{2}}|$, $i = 1 : n$.) Notice that with this selection (2.6) becomes

$$\min_{\bar{s}} \left\{ \bar{s}^T \bar{g}_k + \frac{1}{2} \bar{s}^T \bar{M}_k \bar{s} : \; \|\bar{s}\|_2 \le \Delta_k \right\}, \tag{2.8}$$

with $s_k = D_k \bar{s}_k$. Therefore, provided the ball constraint $\|\bar{s}\|_2 \le \Delta_k$ is inactive near the solution, problem (2.8) yields $s_k = s_k^N$ in a feasible neighbourhood of the solution. Moreover the solution to (2.7) is also $s_k = s_k^N$ near the solution provided $\mathcal{S}_k$ contains s_k^N and the elliptical constraint $\|D_k^{-1} s\|_2 \le \Delta_k$ is inactive.

Global convergence properties of optimization algorithms usually require "sufficient decrease" guarantees [12]. While (2.6) and (2.7) can yield a descent direction s_k at any feasible point x_k, it may not be possible to sufficiently decrease $q(x)$ along s_k. The problem is that the feasibility requirement may restrict α_k to be very small when $x_{k+1} = x_k + \alpha_k s_k$. Sufficient decrease in $q(x)$ may not be possible. Note that, as indicated above, this inhibiting behaviour cannot occur in a feasible neighbourhood of x_*, but it certainly can occur when x_k is far from x_* but close to the boundary.

Our solution to this problem has two parts. The first part involves a search along a piecewise linear path, a reflective path; the second part restricts the descent directions to have certain desirable properties. We discuss the reflective path idea first.

Given a feasible point x and an initial descent direction s we define the reflective path $p(\alpha)$ as follows. First, determine the array BR of breakpoint distances:

$$BR(i) = \max\{(l_i - x_i) \,/\, s_i, \;\; (u_i - x_i) \,/\, s_i)\}. \tag{2.9}$$

Component $BR(i)$ records the distance from x to the breakpoint corresponding to variable x_i in the positive direction s. The array BR can now be used in the definition of the piecewise linear (reflective) path $p(\alpha)$ given by Algorithm 2 in Figure 2.2 and (2.10): for $\beta^{i-1} \le \alpha < \beta^i$,

$$p(\alpha) = b^{i-1} + (\alpha - \beta^{i-1}) p^i. \tag{2.10}$$

```
Algorithm 2 [Let $\beta^0 = 0$, $p^1 = s$, set $b^0 = x_k$.]

For $i = 1, \ldots$
    1. Let $\beta^i$ be the distance to the nearest breakpoint along $p^i$:
           $\beta^i = \min(BR)$.
    2. Define $i$th breakpoint: $b^i = b^{i-1} + (\beta^i - \beta^{i-1})p^i$.
    3. Reflect to get new dir'n and update $BR$:
        (a) $p^{i+1} = p^i$
        (b) For each $j$ such that $(b^i)_j = u_j$ (or $(b^i)_j = l_j$)
            • $BR(j) = BR(j) + |\frac{(u_j - l_j)}{s_j}|$.
            • $(p^{i+1})_j = -(p^i)_j$.
```

FIG. 2.2. *Determine the linear reflective path p.*

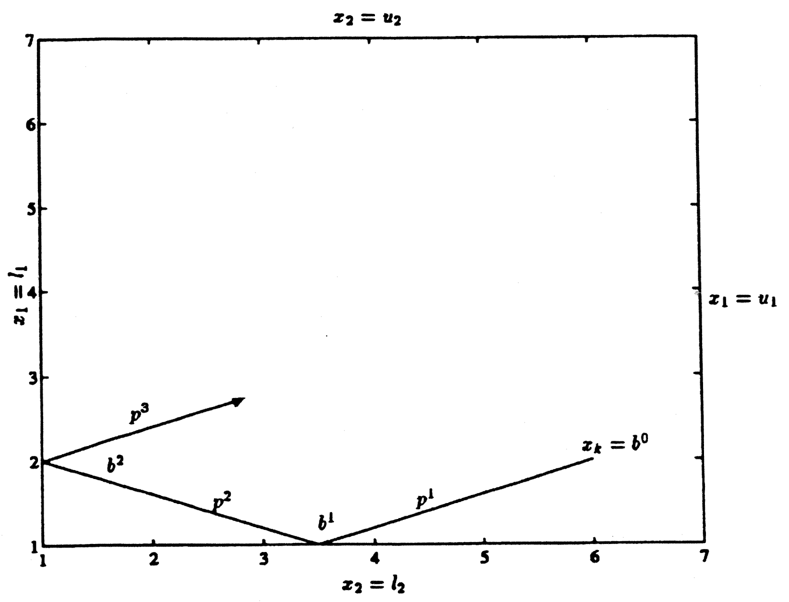

FIG. 2.3. *A reflective path.*

An example of a reflective path is given in Figure 2.3.

If we let $x_{k+1} = x_k + p_k(\alpha_k)$ then our sufficient decrease conditions along s_k are as follows: given $0 < \sigma_l < \sigma_u < 1$ and a descent direction s_k, α_k satisfies our approximate line search conditions if

$$q(x_{k+1}) < q(x_k) + \sigma_l \left(\alpha_k g_k^T s_k + \frac{1}{2}\alpha_k^2 \min(s_k^T H s_k, 0) \right) \tag{2.11}$$

and

$$q(x_{k+1}) > q(x_k) + \sigma_u \left(\alpha_k g_k^T s_k + \frac{1}{2}\alpha_k^2 \min(s_k^T H s_k, 0) \right), \tag{2.12}$$

where $g_k = \nabla q(x_k)$. These conditions are analogous to those suggested by Goldfarb [18] for use in the unconstrained nonlinear setting. As established in Theorem 2 of

[9], if boundedness of $q(x)$ in $\mathcal{F}$ is assumed, then an interval (α_l^k, α_u^k), $\alpha_l^k < \alpha_u^k$, can always be found such that if $\alpha \in (\alpha_l^k, \alpha_u^k)$ then $q(x_k + p_k(\alpha))$ satisfies (2.11) and (2.12). Condition (2.11) implies an upper bound on α; condition (2.12) gives a lower bound.

A basic reflective path algorithm for problem (1.1) can now be stated, as Algorithm 3 (Figure 2.4). To allow for flexibility, especially with regard to Newton steps, we do not always require that both (2.11) and (2.12) be satisfied. Instead, we demand that either both of these conditions are satisfied or (2.11) is satisfied and α_k is bounded away from zero, e.g., $\alpha_k > \rho > 0$. The latter conditions are used to allow for the liberal use of Newton steps and do not weaken the global convergence results.

Note that Algorithm 3 generates *strictly* feasible points; i.e., since $x_1 \in \text{int}(\mathcal{F})$, it follows that $x_k \in \text{int}(\mathcal{F})$.

```
Algorithm 3 [ $\rho$ is a positive scalar.]
    Choose $x_1 \in \text{int}(\mathcal{F})$.
    For $k = 1, 2, ...$
      1. Determine an initial descent dir'n $s_k$ for $q$ at $x_k$. Note that the piecewise
         linear path $p_k$ is defined by $x_k, s_k$.
      2. Perform an approximate piecewise line minimization of $q(x_k + p_k(\alpha))$,
         with respect to $\alpha$, to determine $\alpha_k$ such that
         (a) $\alpha_k$ does not correspond to a breakpoint,
         (b) condition (2.11) is satisfied,
         (c) either
               i. $\alpha_k$ satisfies condition (2.12), or
              ii. $\alpha_k > \rho > 0$.
      3. $x_{k+1} = x_k + p_k(\alpha_k)$.
```

FIG. 2.4. *A reflective path algorithm satisfying line search conditions.*

Satisfaction of the line search conditions in Algorithm 3 is not sufficient to ensure a first-order convergence result. The convergence analysis given in [9] requires two additional properties of the search direction sequence $\{s_k\}$: "constraint compatibility" and "consistency."

DEFINITION 2.1. *A sequence of vectors $\{s_k\}$ is* constraint compatible *if the sequence $\{D_k^{-2} s_k\}$ is bounded.*

DEFINITION 2.2. *A sequence of vectors $\{s_k\}$ satisfies the* consistency condition *if $\{s_k^T g_k\} \to 0$ implies $\{D_k g_k\} \to 0$.*

Consistency is a standard notion which insists that first-order descent, represented by the term $g_k^T s_k$, be consistent with first-order convergence. This is a generalization of the condition that the angle of the gradient and the descent direction are bounded away from 90° in unconstrained minimization [12, p. 123].

Next we illustrate that the constraint compatibility of $\{s_k\}$ enables a sufficiently long step along s_k. Consider the following simple result.

LEMMA 2.3. *If $\{s_k\}$ is a constraint-compatible sequence then $\{BR_k(j) : BR_k(j) = \frac{|(v_k)_j|}{|(s_k)_j|}\}$ is bounded away from zero.*

Constraint compatibility avoids running directly into a bound by ensuring that the stepsizes to breakpoints, corresponding to "correct sign conditions," remain strictly bounded away from zero. Specifically, if $\{s_k\}$ is constraint compatible then the positive distance to constraint j along s_k, $BR_k(j) = \max\{\frac{l_j - (x_k)_j}{(s_k)_j}, \frac{u_j - (x_k)_j}{(s_k)_j}\}$, is strictly bounded away from zero for any j with the correct sign condition. The correct sign condition implies a consistency between v_j and $\max\{\frac{l_j-(x_k)_j}{(s_k)_j}, \frac{u_j-(x_k)_j}{(s_k)_j}\}$. The correct sign condition holds when $(s_k)_j(g_k)_j < 0$, and so $BR_k(j) = \frac{|(v_k)_j|}{|(s_k)_j|}$.

When the "sign condition" is violated, i.e., when $(s_k)_j(g_k)_j > 0$, then a bound may be hit after only a short step along s_k. This is when the reflective line search idea steps forward: the reflective line search guarantees that the new direction passing this breakpoint will maintain descent if the bound is encountered soon enough (since $(s_k)_j(g_k)_j > 0$).

Several examples of descent directions satisfying both constraint compatibility and consistency are given in [9]. It is particularly noteworthy that the solution of the trust region problem (2.6) yields a direction s_k satisfying both conditions.

3. A practical algorithm.

In this section we discuss two specifics of our approach that turn Algorithm 3 into a viable approach for large-scale problems. The first aspect is the line search; the second aspect is a particular way to determine the directions s_k via (2.6) by choosing a low-dimensional subspace $\mathcal{S}_k$ appropriately.

3.1. The line search.

The most natural way to implement the piecewise line search is in a left-to-right fashion. With some cleverness an exact line search can be accomplished, moving left-to-right, requiring $O(n)$ work per breakpoint crossed. Unfortunately, even this linear order of work is too expensive for large-scale problems with many activities at the solution—the number of breakpoints crossed is proportional to the number of activities at the solution.

A viable solution to this problem is to use an approximate bisection algorithm, first attempting a unit step, $\alpha = 1$, to see if condition (2.11) is satisfied. (It is not necessary to test condition (2.12) since clearly a unit step is bounded away from zero!)

An important observation is that the point $x + p(\alpha)$ can easily be evaluated, no need to compute BR or apply Algorithm 2. To see this assume, for simplicity, that for each i all finite lower bounds equal zero and all finite upper bounds equal unity. Suppose we wish to evaluate $x + p(\alpha)$. Define $y = x + \alpha s$ and define R to be the reflective transformation, $R(y) = x + p(\alpha)$. The computation of R is described in Figure 3.1.

```
To evaluate $z_i = R(y)_i$
    Case 1: $l_i = 0,\ u_i = 1$. Set $w_i = |y_i| \bmod 2$, set $z_i = \min(w_i, 2 - w_i)$.
    Case 2: $l_i = 0,\ u_i = \infty$. Set $z_i = |y_i|$.
    Case 3: $l_i = -\infty,\ u_i = 1$. If $y_i \le 1$, set $z_i = y_i$; else, set $z_i = 2 - y_i$.
    Case 4: $l_i = -\infty,\ u_i = \infty$. Set $z_i = y_i$.
```

FIG. 3.1. *The reflective transformation R.*

A bisection (piecewise) line search algorithm which yields a point x_{k+1} satisfying (2.11) and (2.12) can be implemented as follows. Assume x is the current feasible point and s is the current descent direction at x. As usual, $p(\alpha)$ denotes the piecewise linear path defined in (2.10). The simple bisection algorithm is given in Figure 3.2.

3.2. Subspace selection.

We propose to implement step 1 of Algorithm 3, i.e., determine the descent direction s_k, by solving a problem of the form (2.7):

$$\min_s \left\{ s^T g_k + \frac{1}{2} s^T M_k s : \ \|D_k^{-1} s\|_2 \le \Delta_k, \ s \in \mathcal{S}_k \right\}. \tag{3.1}$$

```
Algorithm 4
    • If $R(x + s)$ satisfies (2.11), set $\hat{\alpha} = 1$.
    • Otherwise
        – Set $\alpha_L = 0$, $\alpha_R = 1$, $y = x + \frac{1}{2}s$.
        – Repeat until $R(y)$ satisfies (2.11) and (2.12)
            * If $R(y)$ violates (2.11), set $\alpha_R = y$;
              else if $R(y)$ violates (2.12), set $\alpha_L = y$, end
            * Set $\hat{\alpha} = (\alpha_L + \alpha_R)/2$, $y = x + \hat{\alpha}s$.
    • If $\hat{\alpha}$ corresponds to a breakpoint, set $\alpha = \hat{\alpha} - \epsilon$, where $\epsilon \leq \chi_\alpha \|Dg\|$; otherwise,
      set $\alpha = \hat{\alpha}$.
```

FIG. 3.2. *The piecewise linear bisection algorithm.*

A subspace $\mathcal{S}_k$ must be chosen. If we choose $\mathcal{S}_k = \Re^n$ then strong convergence results follow [9]; however, this choice is impractical for large problems.

To meet global convergence requirements, the inclusion of the scaled gradient $D_k^2 g$ in $\mathcal{S}_k$ (or a direction with a nonzero projection onto the scaled gradient) is useful. Moreover, asymptotic second-order convergence requires that a good approximation to the Newton step s_k^N be included in $\mathcal{S}_k$ when $\bar{M}_k$ is determined to be positive definite. Finally, when $\bar{M}_k$ is determined to be non-positive definite, it is desirable to include a direction of negative curvature in $\mathcal{S}_k$ if available. That is, include $D_k w_k$ in $\mathcal{S}_k$ when $w_k^T \bar{M}_k w_k < 0$. These subspace ideas are similar to those explored in [3] for unconstrained minimization problems. Coleman and Li [9] show that the consistency and constraint-compatibility conditions can be satisfied using the two-dimensional subspace framework.

Thus our practical algorithm has the structure given in Figure 3.3.

```
Algorithm 5 [ $\rho$ is a positive scalar.]
    Choose $x_1 \in \text{int}(\mathcal{F})$.
    For $k = 1, 2, \ldots$
      1. Determine a two-dimensional subspace $\mathcal{S}_k$: if a nonzero vector $w_k$ is
         found such that $w_k^T \bar{M}_k w_k \leq 0$, set $\mathcal{S}_k = < D_k^2 sgn(g_k), D_k w_k >$; other-
         wise, set $\mathcal{S}_k = < D_k^2 g_k, D_k \tilde{s}_k^N >$, where $\tilde{s}_k^N$ approximately solves (2.5).
      2. Determine an initial descent dir'n $s_k$ for $q$ at $x_k$: solve (2.6) to determine
         $s_k$. Note that the piecewise linear path $p_k$ is defined by $x_k, s_k$.
      3. Apply Algorithm 4, the approximate bisection minimization of $q(x_k +
         p_k(\alpha))$, with respect to $\alpha$, to determine $\alpha_k$ such that
         (a) $\alpha_k$ does not correspond to a breakpoint,
         (b) condition (2.11) is satisfied,
         (c) either
               i. $\alpha_k$ satisfies condition (2.12), or
              ii. $\alpha_k > \rho > 0$.
      4. $x_{k+1} = x_k + p_k(\alpha_k)$.
```

FIG. 3.3. *A practical reflective path algorithm satisfying line search conditions.*

Algorithm 5 will locate a second-order point, under reasonable assumptions, when a constraint-compatible direction sequence $\{w_k\}$ of sufficient negative curvature, relative to $\lambda_{min}(\bar{M}_k)$, is guaranteed [9]. The computational results we present in this paper correspond to implementations that do not guarantee that sufficient negative curvature is found. However, our implementations are globally first-order convergent and if there is convergence to a second-order point then, under second-order sufficiency assumptions, a second-order convergence rate can be achieved [9].

We consider two variations of Algorithm 5 in our experimental testing. The two variations represent different ways to compute the subspace $\mathcal{S}_k$ in step 1 of Algorithm 5; both variations have applicability to large-scale problems.

Variation Cholesky. A sparse Cholesky factorization of a permutation of $\bar{M}_k$ is attempted. If this factorization completes, then $\bar{M}_k$ is positive definite and $\tilde{s}_k^N = -\bar{M}_k^{-1}\bar{g}_k$ is computed. Otherwise, a nonzero vector w_k is found such that $w_k^T \bar{M}_k w_k \leq 0$. In our experiments we use the symmetric minimum degree ordering to define a permutation of $\bar{M}_k$ to reduce fill in the Cholesky factor, e.g., [15]. Note that the ordering step, and determination of potential fill, needs to be done only once for a given instance of problem (1.1).

Variation conjugate gradients (PCG). We apply a preconditioned conjugate gradient (PCG) process to the system $\bar{M}_k \tilde{s}_k^N = \bar{g}_k$ until either a vector $\tilde{s}_k^N$ is found such that $\|\bar{M}_k \tilde{s}_k^N - \bar{g}_k\| \leq \epsilon_k$ for a given tolerance $\epsilon_k > 0$ or a nonzero vector w_k is generated such that $w_k^T \bar{M}_k w_k \leq 0$ [29].

Both variations exhibit a first-order global convergence result; moreover, assuming convergence to a point satisfying second-order sufficiency conditions, variation Cholesky will generate a quadratically convergent sequence, whereas variation PCG will yield a superlinearly convergent sequence provided ϵ_k goes to zero sufficiently fast [2].

4. Numerical experiments.

We have implemented our algorithm in MATLAB 4.0, a version of MATLAB which allows for *sparse* matrix data structures [16]. In this section we present some preliminary numerical results.

For each problem we record the number of required iterations for each of our two variations of Algorithm 5, Cholesky and PCG. In the PCG approach we use a simple diagonal preconditioner where the diagonal elements correspond to the 2-norms of the columns of the corresponding matrix $\bar{M}_k$. Our purpose here is not to compare the overall relative efficiencies of these two variations per se but rather to explore the use of each approach within the context of the reflective Newton method, Algorithm 5. Is the number of major iterations required by the reflective Newton method sensitive to the use of Cholesky versus PCG? How does the number of major iterations grow with problem size and is this growth dependent on the choice of PCG versus Cholesky? How does the reflective Newton approach depend on problem characteristics, and is this dependence related to the choice of PCG versus Cholesky? Of course ultimately the relative efficiencies of the Cholesky version versus the PCG depend also on the work required to "solve" each linear system. This, in turn, depends on the usual factors: sparsity and distribution of eigenvalues. Except for a few remarks at the end of this section, we do not explore the latter question in this paper.

All experiments were performed on Sun Sparc workstations in the (sparse) MATLAB environment [23].

Starting and stopping. In all the experiments reported in this paper the starting value of x, i.e., x_1, is as follows. For component j where both upper and lower bounds are finite, choose the midpoint, $(x_1)_j = \frac{l_j + u_j}{2}$. If both upper and lower bound corresponding to component j are infinite in size, choose $(x_1)_j = 0$. If l_j is finite and $u_j = \infty$, choose $(x_1)_j = l_j + 1$; if $l_j = -\infty$ and u_j is finite, choose $(x_1)_j = u_j - 1$.

(Note: The reflective Newton approach is not particularly sensitive to starting value. For example, we repeated many of the experiments reported here using a random (strictly) feasible starting point; very little difference in behaviour was detected.)

Choosing a robust stopping rule in optimization is never easy. Our primary stopping rule is based on the relative difference in function value. This is reasonable partly because strict feasibility is always maintained and partly because usually the real objective in practical optimization is to achieve a point of relatively low function value. Specifically, our primary stopping rule is

$$q(x_k) - q(x_{k+1}) \leq tol * (1 + |q(x_k)|). \tag{4.1}$$

We choose $tol = 100 * \mu$ where μ is unit roundoff (machine epsilon); i.e., in MATLAB on a Sun Sparc workstation, $\mu = 2.2204 * 10^{-16}$. We do have secondary stopping criteria as well, designed to determine when progress is deemed too slow. This secondary rule tends to kick in when solving degenerate or ill-conditioned problems and a very flat region around the solution has been entered.

Of course the stopping criterion mentioned above does not guarantee a minimizer. In our numerical experiments reported in this paper, independent verification of optimality was performed whenever feasible. The optimal point was known, a priori, for the positive definite problems. In this case optimality was verified by comparing the computed function value at the computed solution to the known value (or, in some cases, the computed function value at the known minimizer). Optimality is much more difficult to verify for indefinite problems. For these problems, first-order optimality conditions were verified, and in some cases, when computationally feasible, second-order optimality was confirmed.

Parameter settings. There are a few preset parameters in the algorithm. Here are the settings we used in our experiments:

- σ_l: used in the line search, see (2.11): we use $\sigma_l = .1$.
- σ_u: used in the line search, see (2.12): we use $\sigma_u = .9$.
- ρ: a lower bound on the stepsize, see Algorithm 5: we use $\rho = .1$.
- χ_α: if the line search produces a unit step which turns out to be a breakpoint, this point is perturbed by an amount bounded by $\chi_\alpha \|D_k g_k\|$; see [9]: we use $\chi_\alpha = 1$.
- ϵ_k: used with the PCG variation; we set $\epsilon_k = .1$ for all k.

4.1. Positive definite problems. We have generated a number of quadratic test problems with certain properties. In the first set of results we concentrate on the case where H is symmetric positive definite. In the results reported below we use sparse matrices H with sparsity patterns representing a three-dimensional grid using a seven-point difference scheme. The Moré–Toraldo [24] QP-generator was adapted to generate problems with a given sparsity pattern (see also [5]). We will not review the generator characteristics here: our generator is a straightforward adaptation of the Moré–Toraldo scheme to the sparse setting using several sparse MATLAB functions (e.g., "sprandsym," "sprand").

In Tables 4.1–4.3, the dimension of the test problems is $n = 1000$ in each case. The parameter "pctbnd" indicates the percentage of variables tight at the solution, approximately evenly divided between upper and lower bounds. Parameter "deg" reflects the degree to which the solution is (nearly) degenerate; the larger the value of deg, the greater the amount of (near) degeneracy. Specifically, at the solution the gradient g satisfies $|g_i| \leq 10^{-\deg}$ for some of the indices i corresponding to components of x which are tight at the solution, i.e., $x_i = u_i$ or $x_i = l_i$. Additional discussion of this concept is contained in [5]. Parameter "cond" reflects the conditioning of the matrix H; the condition number of H is approximately 10^{cond}.

TABLE 4.1
Positive definite problems, pctbnd = .1, $n = 1000$.

		Cholesky		PCG	
deg	cond	max	avg	max	avg
3	3	15	14	16	14.7
6	3	16	15.6	17	16.3
9	3	15	15	16	16
3	6	14	12.7	15	14
6	6	16	15.3	17	16
9	6	15	15	17	16.7
3	9	13	12.7	14	13.7
6	9	16	15.3	17	16
9	9	16	15.7	17	16.3

TABLE 4.2
Positive definite problems, pctbnd = .5, $n = 1000$.

		Cholesky		PCG	
deg	cond	max	avg	max	avg
3	3	15	15	17	16
6	3	17	17	19	18.3
9	3	17	16.7	18	18
3	6	16	15.3	17	16
6	6	18	17.3	19	17.3
9	6	17	17	19	18.7
3	9	15	14.3	17	16
6	9	17	17	18	18
9	9	17	16.3	18	17.7

TABLE 4.3
Positive definite problems, pctbnd = .9, $n = 1000$.

		Cholesky		PCG	
deg	cond	max	avg	max	avg
3	3	17	16.7	18	17.7
6	3	18	17.3	19	18
9	3	17	16.3	19	18.7
3	6	16	15.7	18	16.7
6	6	18	17.3	19	18.3
9	6	18	17.3	18	16.7
3	9	16	15.3	16	15.7
6	9	17	17	18	17
9	9	17	16.7	18	17.7

The upper and lower bound vectors u and l were generated as follows. Approximately 75% of the components of l were chosen to be finite and assigned the value of zero; the index assignment was made in a random fashion. Similarly, approximately 75% of the components of u were chosen to be finite and assigned the value of unity. Again, the index assignment was made in a random fashion, independent of the assignment of l.

Each row of Tables 4.1–4.3 reflects the results of three independent runs with the same parameter settings. The columns labeled "max" indicate the maximum number

TABLE 4.4
Positive definite problems: Timing breakdown.

	Cholesky			PCG		
n	it	totM	totls	it	totM	totls
512	18	11.8	1.7	15	4.5	1.8
1000	15	38.5	3.9	16	8.4	3.8
1728	17	133.5	6.1	16	16.4	6.7
2744	16	533	11.4	17	33.3	12.1
8000	15	5369	29.4	17	95.2	33.6

of iterations required over the set of three independent runs to achieve the stopping criteria. The column labelled "avg" records the average number of iterations required to reach the stopping criteria over the three problems.

Full accuracy, in terms of function value, was achieved for every problem run. That is, the computed value of the objective function $q(x)$ at the computed solution matched the computed value of $q(x)$ at the true (known) solution to 15 decimal places.

Observations on Tables 4.1–4.3. First, we observe the remarkable consistency of both versions of the reflective Newton method in these problems. In terms of iterations required to achieve the stopping criteria and accuracy attained in the function value, there is apparently very little sensitivity to degeneracy, conditioning, or number of variables tight at the solution. Of course we do not claim that accuracy in x is independent of condition/degeneracy—it surely is not. However, it is usually acceptable in optimization to locate a point with nearly optimal function value, and we have been quite successful in that (on this test collection).

Second, the absolute number of iterations required to obtain a very accurate solution (in terms of the function value q) is modest in every case for both reflective Newton variations, i.e., less than 20. In general the PCG version required 1–2 additional iterations compared with the Cholesky version.

It is important to know where an algorithm spends its time. To this end we generated larger problems, with the same structure, and we have broken down the timing information. In Table 4.4 we consider a representative positive definite problem with "average characteristics," i.e., deg = 6, cond = 6, pctbnd = .5, and we vary the problem dimension n. (The sparsity structure remains the same.) The second column, labelled "it," records the number of iterations required to achieve the stopping criteria; "totM" records the total number of mega-flops used in the matrix operations corresponding to each linear system $\bar{M}_k s_k = \bar{g}_k$; "totls" records the number of mega-flops used in the approximate line search algorithm. More than 95% of the total flop count on these problems is represented by the sum of the totM and totls columns. The remaining work in the algorithm, such as the two-dimensional trust region solution, is negligible in comparison.

Observations on Table 4.4. First, there is no significant growth in number of iterations as the problem dimension n increases for either version. In the Cholesky variation, the sparse matrix factorization work totM increases relative to the line search cost totls as n increases; however, this trend is not evident in the PCG version. For large n the cost of the line search PCG is comparable to the linear solve whereas in Cholesky the line search cost is dominated by the matrix factorization cost. Therefore, in the Cholesky variation, speedup of the sparse Cholesky factorization aspect of the algorithm (e.g., use of parallelism, exploitation of specific particular structure) will have significant impact on the overall computing time. Conversely, improving the approximate line search (in terms of cost) in the PCG version is as crucial as decreasing the cost of the matrix operations in the linear solve.

In addition to these randomly generated, but structured, positive definite problems, we have experimented with three specific test cases. Two of these problems are from the literature (e.g., [11, 25]) and the third example is new. In Tables 4.5 and 4.6 we report on the "obstacle" problem. In the first case there are lower bounds only; in the second case there are lower and upper bounds. In defining the specific example used we have chosen the same parameter settings and specific functions used in [25]. Table 4.7 reports on the elastic-plastic torsion problem. Again we used the same parameters as reported in [24] to define the problem; the parameter "c," as defined in [24], was assigned $c = 5$ in our experiments. The columns labelled "norm" record the logarithm (base 10) of the reciprocal of the final 2-norms of the vector D^2g, a first-order measure or optimality; see (1.3). For example, if $\|D^2g\| = 10^{-7}$ then the corresponding norm table entry equals 7.

TABLE 4.5
Obstacle problem: Lower bounds only.

		Cholesky		PCG	
m	n	its	norm	its	norm
30	900	14	15	17	8
40	1600	14	15	14	7
50	2500	15	15	21	8
60	3600	16	14	18	7
100	10,000	15	16	17	6

TABLE 4.6
Obstacle problem: Lower and upper bounds.

		Cholesky		PCG	
m	n	its	norm	its	norm
30	900	12	10	12	7
40	1600	12	16	12	7
50	2500	13	16	13	7
60	3600	13	16	15	9
100	10,000	14	9	14	8

TABLE 4.7
Elastic-plastic torsion problem.

		Cholesky		PCG	
m	n	its	norm	its	norm
30	900	10	15	11	6
40	1600	11	13	11	6
50	2500	11	13	16	9
60	3600	11	14	10	5
100	10,000	10	14	12	7

In Table 4.8 we report on a linear spline approximation problem. This type of problem arises, for example, in a particle method approach to turbulent combustion simulation [28]. The problem results in a large sparse least-squares problem subject to nonnegativity constraints on the variables. To set up a sample problem we assume an m-by-m-by-m three-dimensional grid. Within each cell is a set of particles randomly located. (We use approximately 10 particles per cell in our experiments.) Each

TABLE 4.8
Linear spline approximation.

		Cholesky		PCG	
m	n	its	norm	its	norm
30	900	15	13	14	6
40	1600	15	15	15	6
50	2500	15	15	17	7
60	3600	15	15	17	7
100	10,000	16	15	16	6

particle p has a known function value $\phi(p)$. Associate with each grid intersection point a linear basis function and determine the best set of coefficients, x, for the basis functions in the least-squares sense subject to nonnegativity constraints on x. The function ϕ we used in our experiments is defined as follows. Given a point in 3-space, $p = (p_1, p_2, p_3)$, define

$$\phi(p) = .3\sin(9.2p_1)\sin(9.3p_2)\sin(9.4p_3).$$

Observations on Tables 4.5–4.8. The most noteworthy observation is the apparent insensitivity of the reflective Newton method to problem size for each of these problems. The number of iterations does not grow significantly for a given problem class as the dimension of the problem increases. For example, for the linear spline problem, 15 iterations are required by the Cholesky variation when $n = 900$; 16 iterations are required when $n = 10,000$. Similarly, the PCG variation requires 14 iterations when $n = 900$ and 16 iterations when $n = 10,000$. In general, PCG required a few more iterations to satisfy the stopping criteria than did Cholesky. Moreover, the accuracy achieved by PCG, in terms of the final norm of D^2g, is considerably worse than Cholesky. The total amount of work is another matter; see Table 4.4 and subsequent comments.

4.2. Indefinite problems. We have adapted the Moré–Toraldo QP generation scheme, in combination with sparse matrix functions in MATLAB 4.0, to generate large sparse indefinite matrices, each with a given sparsity pattern and preassigned set of approximate eigenvalues. In the indefinite case we chose finite upper and lower bound vectors, $l = 0$, $u = 1$. (We use all finite bounds in the generation of indefinite problems to avoid the generation of unbounded problems.)

In each of the problems in Tables 4.9–4.11 roughly 10% of the eigenvalues of H are negative. (In each case the optimality conditions were verified to hold at the final point.)

Observations on Tables 4.9–4.11. Iteration counts indicate that the reflective Newton method is not quite as consistent or efficient on indefinite problems compared with the performance on positive definite problems. Still, the overall efficiency seems very good—the average number of iterations required for any problem category is always less than 23 for Cholesky and less than 26 for PCG.

The difference between PCG and Cholesky, in terms of number of iterations to satisfy the optimality conditions, is greater on the set of indefinite problems than on the positive definite set. It is not clear why this is the case.

In Table 4.12 we indicate where the algorithm spends its time on indefinite problems by considering a representative example and increasing the dimension.

Remark on Table 4.12. We see little growth in required iterations for either variation as n increases. Clearly the totM column dominates the totls column as

TABLE 4.9
Indefinite problems, pctbnd = .1, n = 1000.

		Cholesky		PCG	
deg	cond	max	avg	max	avg
3	3	18	16.7	21	20
6	3	19	17	22	21
9	3	23	19.3	22	21.3
3	6	14	13.7	19	18.3
6	6	32	22.7	25	22.7
9	6	26	21.7	24	22.7
3	9	15	14	24	21
6	9	16	15.7	24	22
9	9	16	15.7	25	21.7

TABLE 4.10
Indefinite problems, pctbnd = .5, n = 1000.

		Cholesky		PCG	
deg	cond	max	avg	max	avg
3	3	17	15.7	24	20
6	3	19	18	24	22.3
9	3	18	16.7	25	25
3	6	15	13.3	21	19
6	6	18	17.3	28	23.3
9	6	19	17.7	25	23.6
3	9	14	11.3	28	23
6	9	16	15.7	19	18.3
9	9	25	18.3	23	21.7

n increases for the Cholesky version; in the PCG variation the totM and totls are comparable. Recall that totM represents the matrix flop count for the linear solve while totls represents the number of total mega-flops required by the line search procedure. Again, for large n the cost of the line search PCG is comparable to the linear solve, whereas in Cholesky the line search cost is dominated by the matrix factorization cost. Therefore, to obtain further improvements in efficiency of the Cholesky variation it is best to focus on the matrix factorization aspect of the overall procedure. Equal attention must be given to both aspects, line search and linear solve, in the PCG variation.

Overall remarks on computational experiments. As mentioned above, the purpose of the experimental study is not to compare the PCG variation with the Cholesky variation in terms of overall efficiency. Rather, the purpose is to study the numerical behaviour of two possible implementations of the reflective Newton strategy. Our numerical results indicate that either implementation, PCG or Cholesky, is credible within this framework. The best choice between these two variations is problem dependent. In most of our experiments Cholesky required slightly fewer iterations but often significantly more time—witness Tables 4.4 and 4.12. On the other hand, there certainly are situations where the Cholesky variation is more economical. For example, in Table 4.13 we present the result of our experiments with a quadratic problem based on the evaluation of the Hessian matrix in [10]. In this case Cholesky is the clear winner in terms of operation count.

In the PCG variation, the fundamental subtask in both the line search and the linear solve operations is multiplication between the Hessian matrix and several vectors. The number of Hessian–vector products for our implementation of the PCG variation

TABLE 4.11
Indefinite problems. pctbnd = .9, $n = 1000$.

		Cholesky		PCG	
deg	cond	max	avg	max	avg
3	3	16	13.3	22	21
6	3	18	16	26	21.3
9	3	16	11	33	24
3	6	13	12	20	17.7
6	6	14	13	19	17
9	6	16	14.3	30	25.7
3	9	12	11.3	19	18
6	9	15	13.7	17	16.7
9	9	15	12.7	19	16.7

TABLE 4.12
Indefinite problems: Timing breakdown.

	Cholesky			PCG		
n	it	totM	totls	it	totM	totls
512	17	10.8	2.1	18	2.5	2.3
1000	19	38.2	4.9	25	7.4	6.8
1728	22	115.5	9.1	19	9.4	8.2
2744	24	396.6	16.4	24	18.3	16.7
8000	32	5030	69	31	78.4	67.8

averaged about $10 * it$ plus the total number of conjugate gradient iterations. Our implementation of the Cholesky variation averaged $10 * it$ Hessian–vector products.

5. Concluding remarks.

The reflective Newton idea represents a new interior approach to box-constrained minimization problems. A convergence theory establishing global and second-order convergence results is presented in [9]; in the present paper we have provided a motivation of this approach and given results of computational experiments for sparse quadratic (indefinite) objective functions. The numerical results strongly support the notion that this approach is attractive for solving large sparse box-constrained quadratic programs.

Two variations of the reflective Newton method for box-constrained quadratic programs have been proposed in this paper. The Cholesky variation attempts a sparse Cholesky factorization in each iteration. The matrix to be factored has the same structure as the original Hessian matrix H and ultimately turns positive definite in the neighbourhood of a strong local minimizer. If the factorization fails then the matrix is not positive definite and a direction of negative curvature is usually available. The reflective Newton method uses this direction to help locate the next point. The PCG variation attempts to solve a local "Newton" system, each major iteration k, via PCG iterations. If negative curvature is found in the process, which may happen far from the solution (but not in a neighbourhood of the solution), then the PCG process is aborted and the reflective Newton method uses the negative direction to help locate the next point.

Regardless of which linear solver is used, the reflective Newton method then proceeds by solving a two-dimensional trust region problem to locate an initial feasible descent direction. A simple one-dimensional reflective path search is then performed to locate an improved point. We show how to efficiently implement an approximate search along this piecewise linear path.

Note that this approach does not involve an artificial "merit" function such as a

TABLE 4.13
Problem var.

	Cholesky		PCG	
n	it	m-flops	it	m-flops
1000	16	6.1	16	14.1
5000	17	32.9	14	62.7
10,000	17	65.9	17	225.3

barrier function—no delicate choice of barrier or penalty parameter is needed—nor does a "central path" need to be followed. These are potential advantages of this proposed new method. Moreover, our computational experiments to date indicate that this approach has considerable promise as a robust and efficient way to solve large box-constrained quadratic programs.

REFERENCES

[1] A. BJORCK, *A direct method for sparse least squares problems with lower and upper bounds*, Numer. Math., 54 (1988), pp. 19–32.

[2] M. A. BRANCH, T. F. COLEMAN, AND Y. LI, *A Subspace, Interior, and Conjugate Gradient Method for Large-Scale Bound-Constrained Minimization Problems*, Tech. report TR95-1525, Department of Computer Science, Cornell University, Ithaca, NY, 1995.

[3] R. H. BYRD, R. B. SCHNABEL, AND G. SHULTZ, *Approximate solution of the trust region problem by minimization over two-dimensional subspaces*, Math. Programming, 40 (1988), pp. 247–263.

[4] T. F. COLEMAN AND L. A. HULBERT, *A direct active set algorithm for large sparse quadratic programs with simple bounds*, Math. Programming, 45 (1989), pp. 373–406.

[5] ———, *A globally and superlinearly convergent algorithm for convex quadratic programs with simple bounds*, SIAM J. Optim., 3 (1993), pp. 298–321.

[6] T. F. COLEMAN AND Y. LI, *A quadratically-convergent algorithm for the linear programming problem with lower and upper bounds*, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., SIAM, Philadelphia, PA, 1990, pp. 49–57.

[7] ———, *A global and quadratically-convergent method for linear l_∞ problems*, SIAM J. Numer. Anal., 29 (1992), pp. 1166–1186.

[8] ———, *A globally and quadratically convergent affine scaling method for linear l_1 problems*, Math. Programming, 56 (1992), pp. 189–222.

[9] ———, *On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds*, Math. Programming, 67 (1994), pp. 189–224.

[10] A. R. CONN, N. I. M. GOULD, AND P. L. TOINT, *Testing a class of methods for solving minimization problems with simple bounds on the variables*, Math. Comp., 50 (1988), pp. 399–430.

[11] R. S. DEMBO AND U. TULOWITZKI, *On the Minimization of Quadratic Functions Subject to Box Constraints*, Tech. report B 71, School of Organization and Management, Yale University, New Haven, CT, 1983.

[12] J. E. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[13] F. FACCHINEI AND S. LUCIDI, *A class of penalty functions for optimization problems with bound constraints*, Optimization, 26 (1992), pp. 239–259.

[14] R. FLETCHER AND M. P. JACKSON, *Minimization of a quadratic function of many variables subject only to lower and upper bounds*, J. Inst. Math. Appl., 14 (1974), pp. 159–174.

[15] A. GEORGE AND J. W.-H. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewoods Cliffs, NJ, 1981.

[16] J. GILBERT, C. MOLER, AND R. SCHREIBER, *Sparse matrices in* MATLAB*: Design and implementation*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 333–356.

[17] P. GILL AND W. MURRAY, *Minimization Subject to Bounds on the Variables*, Tech. report NAC 71, National Physical Laboratory, England, 1976.

[18] D. GOLDFARB, *Curvilinear path steplength algorithms for minimization algorithms which use directions of negative curvature*, Math. Programming, 18 (1980), pp. 31–40.

[19] C.-G. HAN, P. M. PARDALOS, AND Y. YE, *Computational aspects of an interior point algo-*

rithm for quadratic programming problems with box constraints, in Large-Scale Numerical Optimization, T. F. Coleman and Y. Li, eds., SIAM, Philadelphia, PA, 1990, pp. 92–112.

[20] J. J. JÚDICE AND F. M. PIRES, *Direct methods for convex quadratic programs subject to box constraints*, Investigacao Peracional, 9 (1989), pp. 23–56.

[21] Y. LI, *A globally convergent method for l_p problems*, SIAM J. Optim., 3 (1993), pp. 609–629.

[22] P. LOTSTEDT, *Solving the minimal least squares problem subject to bounds on the variables*, BIT, 24 (1984), pp. 206–224.

[23] C. B. MOLER, J. LITTLE, S. BANGERT, AND S. KLEIMAN, *ProMatlab User's Guide*, MathWorks, Sherborn, MA, 1987.

[24] J. J. MORÉ AND G. TORALDO, *Algorithms for bound constrained quadratic programming problems*, Numer. Math., 55 (1989), pp. 377–400.

[25] ———, *On the solution of large quadratic programming problems with bound constraints*, SIAM J. Optim., 1 (1991), pp. 93–113.

[26] D. P. O'LEARY, *A generalized conjugate gradient algorithm for solving a class of quadratic programming problems*, Linear Algebra Appl., 34 (1980), pp. 371–399.

[27] U. ÖREBORN, *A direct method for sparse nonnegative least squares problems*, Ph.D. thesis, Department of Mathematics, Linköping University, Linköping, Sweden, 1986.

[28] S. POPE, *Application of the velocity-dissipation PDF model to inhomogeneous turbulent flows*, Phys. Fluids A, 3 (1991), pp. 1947–1957.

[29] T. STEIHAUG, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.

[30] E. K. YANG AND J. W. TOLLE, *A class of methods for solving large convex quadratic programs subject to box constraints*, Math. Programming, 51 (1991), pp. 223–228.