QUADRATIC PROGRAMMING WITH M-MATRICES*

by

Franklin T. Luk
and
Marcello Pagano

TR 79-396

Department of Computer Science
Cornell University
Ithaca, NY 14853

Harvard University
Sidney Farber Cancer Institute
Boston, MA 02115

*This waork supported in part by U.S. Army Research Office Grant DAAG 29-78-G01079.

QUADRATIC PROGRAMMING

WITH M-MATRICES

by

Franklin T. Luk

Department of Computer Science
Cornell University
Ithaca, NY 14853

and

Marcello Pagano
Harvard University
Sidney Farber Cancer Institute
44 Binney Street
Boston, MA 02115

Accepted for publication in Linear Algebra and its Applications.

This work supported in part by U.S. Army Research Office Grant DAAG 29-78-G0179.

Abstract:

In this paper, we study the problem of quadratic programming with M-matrices. We describe (1) an effective algorithm for the case where the variables are subject to a lower bound constraint, and (2) an analogous algorithm for the case where the variables are subject to lower and upper bounds constraints. We demonstrate the special monotone behavior of the iterate and gradient vectors. The result on the gradient vector is new. It leads us to consider a simple updating procedure which preserves the monotonicity of both vectors. The procedure uses the fact that an M-matrix has a non-negative inverse. Two new algorithms are then constructed by incorporating this updating procedure into the two given algorithms. We give numerical examples which show that the new methods can be more efficient than the original ones.

QUADRATIC PROGRAMMING WITH M-MATRICES

1. Introduction

In this paper we address the lower and upper bounds quadratic program

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.1)$$

$$\text{subject to} \quad \underset{\sim}{c} \leq \underset{\sim}{x} \leq \underset{\sim}{d} ,$$

where A is an $n \times n$ M-matrix and $\underset{\sim}{b}$, $\underset{\sim}{c}$ and $\underset{\sim}{d}$ are n-vectors. An important special case of (1.1) is the linear complementarity problem, in which $\underset{\sim}{c} = \underset{\sim}{0}$ and $\underset{\sim}{d} = \underset{\sim}{\infty}$:

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.2)$$

$$\text{subject to} \quad \underset{\sim}{x} \geq \underset{\sim}{0} .$$

We assume that the matrix A is large and sparse. The problems (1.1) and (1.2) find applications in the numerical solution of free boundary problems for elliptic partial differential equations. Such problems include various types of Dirichlet problems with obstacles ([7] and [10]), and models of the journal bearing [5] and of the application of torsion to a bar [1].

We define an M-matrix as follows.

DEFINITION 1.1 [11, p. 85]. A real square matrix $A = (a_{ij})$ with $a_{ij} \leq 0$ for all $i \neq j$ is an M-matrix if A is nonsingular, and $A^{-1} \geq 0$.

We lose no generality by restricting our attention to symmetric M-matrices, for we can replace the matrix A by its symmetric part $(A + A^t)/2$ in the quadratic form of (1.1) or (1.2) without changing the value of the quadratic form. The next lemma states that a symmetric M-matrix is positive definite. Thus, the problem (1.1) or (1.2) always has a unique solution.

DEFINITION 1.2 [11, p. 85]. A real square matrix $A = (a_{ij})$ with $a_{ij} \leq 0$ for all $i \neq j$ is a Stieltjes matrix if A is symmetric and positive definite.

LEMMA 1.1 [11, p. 85 and 87]. A symmetric M-matrix is a Stieltjes matrix and vice versa.

There are many good algorithms for solving problems (1.1) and (1.2) when the matrix A is positive definite (see [4]). However, it is possible to exploit the special properties of an M-matrix to obtain more efficient special algorithms. Chandrasekaran [2] proposed an algorithm for the linear complementarity problem (1.2), and Pang [8] developed an analogous algorithm for the lower and upper bounds problem (1.1). Cottle and Cohoen [4] performed an extensive comparison of the latter method with four other well known algorithms. Their results indicate that Pang's method is very efficient. They also described a preprocessing scheme that could be used with the method. Their scheme would identify some of the variables which will be at their bounds in

the optimal solution. Those variables could then be eliminated from further consideration.

In this paper we study the algorithms of Chandrasekaran and Pang. We demonstrate the special monotone behavior of the iterate and gradient vectors. The result on the gradient vector is new. It leads us to consider a simple updating procedure which preserves the monotonicity of both vectors. The procedure uses the fact that an M-matrix has a nonnnngative inverse. Two new algorithms are then constructed by incorporating this updating procedure into the two described algorithms. Numerical tests show that our new algorithms can be more efficient than the original methods (see Section 5).

We also consider the extension of problem (1.1) to the case when some components of the bounds are infinite. We show how we can compute finite a priori bounds on all the variables and reduce the extended problem to one with finite bounds (see Section 4). This result is new.

Let

$$v(\underset{\sim}{x}) = \frac{1}{2}\underset{\sim}{x}^t A\underset{\sim}{x} - \underset{\sim}{x}^t\underset{\sim}{b} \tag{1.3}$$

and

$$\underset{\sim}{g}(\underset{\sim}{x}) = A\underset{\sim}{x} - \underset{\sim}{b}\ , \tag{1.4}$$

i.e., $v(\underset{\sim}{x})$ is the value of the quadratic form at $\underset{\sim}{x}$ and $\underset{\sim}{g}(\underset{\sim}{x})$ is the gradient vector of the quadratic form at $\underset{\sim}{x}$. The solution $\underset{\sim}{x}$ for problem (1.1) must satisfy the optimality conditions:

$$\begin{aligned} &\text{if } x_i = c_i \quad \text{then } g_i(\underset{\sim}{x}) \geq 0 \\ &\text{if } c_i < x_i < d_i \quad \text{then } g_i(\underset{\sim}{x}) = 0 \\ &\text{if } x_i = d_i \quad \text{then } g_i(\underset{\sim}{x}) \leq 0\ , \end{aligned} \tag{1.5}$$

for $i = 1,2,\ldots,n$. The solution $\underset{\sim}{x}$ for problem (1.2) must satisfy the optimality conditions:

$$\begin{aligned} &\text{if } x_i = 0 \quad &&\text{then} \quad g_i(\underset{\sim}{x}) \geq 0 \\ &\text{if } x_i > 0 \quad &&\text{then} \quad g_i(\underset{\sim}{x}) = 0 , \end{aligned} \tag{1.6}$$

for $i = 1,2,\ldots, n$.

The following lemma and notations are used in this paper.

DEFINITION 1.3 [11, p. 30]. A principal submatrix of an $n \times n$ matrix A is any matrix obtained by crossing out any j rows and the corresponding j columns of A, where $1 \leq j < n$.

LEMMA 1.2 [11, p. 86]. Any principal submatrix of an M-matrix is an M-matrix.

NOTATIONS.

(1) $N \equiv \{1,2,\ldots,n\}$.

(2) $\bar{J} \equiv N \setminus J$ for any subset J of N, i.e. $\bar{J}$ is the complement of J in N.

(3) $\underset{\sim}{x}_J \equiv (x_{j_1}, x_{j_2}, \ldots, x_{j_p})$, where $J = \{j_1, j_2, \ldots, j_p\} \subseteq N$ and $j_1 < j_2 < \cdots < j_p$.

(4) $A_{JK} \equiv (c_{rs})$, where

$c_{rs} = a_{j_r k_s}$, $J = \{j_1, j_2, \ldots, j_p\} \subseteq N$,

$K = \{k_1, k_2, \ldots, k_q\} \subseteq N$, $j_1 < j_2 < \cdots < j_p$ and $k_1 < k_2 < \cdots < k_q$.

2. Linear Complementarity Problem

In this section, we consider the linear complementarity problem

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \tag{1.2}$$

$$\text{subject to} \qquad \underset{\sim}{x} \geq \underset{\sim}{0} ,$$

where A is a symmetric M-matrix. We assume that $\underset{\sim}{b}$ contains both positive and negative components to eliminate two trival cases.

LEMMA 2.1.

(1) If $\underset{\sim}{b} \leq \underset{\sim}{0}$, then $\underset{\sim}{x} = \underset{\sim}{0}$ solves (1.2).

(2) If $\underset{\sim}{b} \geq \underset{\sim}{0}$, then $\underset{\sim}{x} = A^{-1}\underset{\sim}{b}$ solves (1.2).

Proof.

(1) Assume $\underset{\sim}{x}$ solves (1.2) with $\underset{\sim}{x}_P > \underset{\sim}{0}$ and $\underset{\sim}{x}_{\bar{P}} = \underset{\sim}{0}$ for some nonempty set $P \subseteq N$. The optimality conditions (1.6) give

$$g_P(\underset{\sim}{x}) \equiv A_{PP}\underset{\sim}{x}_P - \underset{\sim}{b}_P = \underset{\sim}{0} .$$

Hence

$$\underset{\sim}{x}_P = A_{PP}^{-1}\underset{\sim}{b}_P \leq \underset{\sim}{0} . \quad \text{because} \quad A_{PP}^{-1} \geq 0 \quad \text{and} \quad \underset{\sim}{b}_P \leq \underset{\sim}{0} .$$

Contradiction.

(2) We have

$$\underset{\sim}{x} = A^{-1}\underset{\sim}{b} \geq \underset{\sim}{0} \quad \text{because} \quad A^{-1} \geq 0 \quad \text{and} \quad \underset{\sim}{b} \geq \underset{\sim}{0} ,$$

and

$$g(\underset{\sim}{x}) = A\underset{\sim}{x} - \underset{\sim}{b} = \underset{\sim}{0} .$$

Thus, $\underset{\sim}{x}$ solves (1.2). □

We proceed to prove two more lemmas and an important theorem.

LEMMA 2.2. Suppose we have a Stieltjes matrix A and a nonempty subset I of N. The matrix

$$V \equiv A_{II} - A_{I\bar{I}} A_{\bar{I}\bar{I}}^{-1} A_{\bar{I}I}$$

is a Stieltjes matrix.

Proof. The matrix V is symmetric because $A_{I\bar{I}} = A_{\bar{I}I}^{t}$. Let $W \equiv A^{-1}$. Since A is a positive definite matrix, so is W. A fortiori, so are W_{II} and W_{II}^{-1}. But $V = W_{II}^{-1}$ [6, p. 99]. The matrix $A_{\bar{I}\bar{I}}$ is an M-matrix by Lemma 1.2. Hence $A_{\bar{I}\bar{I}}^{-1} \geq 0$. The off-diagonal elements of V are therefore nonpositive because the off-diagonal elements of A_{II} are nonpositive, and $A_{I\bar{I}} \leq 0$, $A_{\bar{I}\bar{I}}^{-1} \geq 0$ and $A_{\bar{I}I} \leq 0$. □

LEMMA 2.3. Suppose we have vectors $\underset{\sim}{x}$ and $\underset{\sim}{y}$ with $\underset{\sim}{x}_J = \underset{\sim}{y}_J$ and $\underset{\sim}{x}_{\bar{J}} \leq \underset{\sim}{y}_{\bar{J}}$ for some nonempty subset J of N. Then $\underset{\sim}{g}_J(\underset{\sim}{x}) \geq \underset{\sim}{g}_J(\underset{\sim}{y})$.

Proof.

$$\begin{aligned}
\underset{\sim}{g}_J(\underset{\sim}{x}) &= A_{JJ}\underset{\sim}{x}_J + A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J \\
&= A_{JJ}\underset{\sim}{y}_J + A_{J\bar{J}}\underset{\sim}{y}_{\bar{J}} - \underset{\sim}{b}_J + A_{J\bar{J}}(\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_{\bar{J}}) \\
&= \underset{\sim}{g}_J(\underset{\sim}{y}) + A_{J\bar{J}}(\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_{\bar{J}}) \\
&\geq \underset{\sim}{g}_J(\underset{\sim}{y})
\end{aligned}$$

because $A_{J\bar{J}} \leq 0$ and $\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_{\bar{J}} \leq \underset{\sim}{0}$. □

THEOREM 2.1. Assume $A_{JJ}^{-1}b_J > \underset{\sim}{0}$ for some nonempty subset J of N. Define a vector $\underset{\sim}{x}$ with $\underset{\sim}{x}_J = A_{JJ}^{-1}b_J$ and $\underset{\sim}{x}_{\bar{J}} = \underset{\sim}{0}$. Let K and L be two sets partitioning $\bar{J}$ so that $g_K(\underset{\sim}{x}) < \underset{\sim}{0}$ and $g_L(\underset{\sim}{x}) \geq \underset{\sim}{0}$.

If the set K is empty, then

(1) $\underset{\sim}{x}$ solves (1.2),

else

(2) let $Q := J \cup K$. Construct $\underset{\sim}{y}$ with $\underset{\sim}{y}_Q = A_{QQ}^{-1}b_Q$ and $\underset{\sim}{y}_L = \underset{\sim}{0}$.

We get

(a) $\underset{\sim}{y}_J \geq \underset{\sim}{x}_J > \underset{\sim}{0}$, $\underset{\sim}{y}_K > \underset{\sim}{0}$,

(b) $g_L(\underset{\sim}{y}) \leq g_L(\underset{\sim}{x})$,

and

(c) $\nu(\underset{\sim}{y}) < \nu(\underset{\sim}{x})$. (Remember $\nu(\underset{\sim}{z}) \equiv \frac{1}{2}\underset{\sim}{z}^t A \underset{\sim}{z} - \underset{\sim}{z}^t \underset{\sim}{b}$).

Proof.

(1) The optimality conditions (1.6) are satisfied because $g_J(\underset{\sim}{x}) = \underset{\sim}{0}$ and $g_{\bar{J}}(\underset{\sim}{x}) \geq \underset{\sim}{0}$.

(2a) From the system

$$A_{JJ}\underset{\sim}{y}_J + A_{JK}\underset{\sim}{y}_K = \underset{\sim}{b}_J ,$$

$$A_{KJ}\underset{\sim}{y}_J + A_{KK}\underset{\sim}{y}_K = \underset{\sim}{b}_K ,$$

we obtain

$$\underset{\sim}{y}_K = (A_{KK} - A_{KJ}A_{JJ}^{-1}A_{JK})^{-1}\,(\underset{\sim}{b}_K - A_{KJ}A_{JJ}^{-1}\underset{\sim}{b}_J) .$$

But

$$\underset{\sim}{b}_K - A_{KJ}A_{JJ}^{-1}\underset{\sim}{b}_J = \underset{\sim}{b}_K - A_{KJ}\underset{\sim}{x}_J$$

$$= -g_K(\underset{\sim}{x}) > \underset{\sim}{0}$$

and

$$(A_{KK} - A_{KJ}A_{JJ}^{-1}A_{JK})^{-1} \geq 0 \quad \text{by Lemma 2.2.}$$

Thus,

$$\underset{\sim}{y}_K > \underset{\sim}{0} .$$

Also,

$$\begin{aligned} \underset{\sim}{y}_J &= A_{JJ}^{-1}\underset{\sim}{b}_J - A_{JJ}^{-1}A_{JK}\underset{\sim}{y}_K \\ &\geq A_{JJ}^{-1}\underset{\sim}{b}_J \qquad \text{because } A_{JJ}^{-1} \geq 0,\ A_{JK} \leq 0 \text{ and } \underset{\sim}{y}_K > \underset{\sim}{0} \\ &= \underset{\sim}{x}_J \\ &> \underset{\sim}{0} . \end{aligned}$$

(2b) We have shown in (2a) that $\underset{\sim}{y}_J \geq \underset{\sim}{x}_J$ and $\underset{\sim}{y}_K > \underset{\sim}{0} = \underset{\sim}{x}_K$. Now $\bar{L} = J \cup K$ and $\underset{\sim}{y}_L = \underset{\sim}{x}_L = \underset{\sim}{0}$. The conditions of Lemma 2.3 are satisfied and therefore $g_L(\underset{\sim}{y}) \leq g_L(\underset{\sim}{x})$.

$$\begin{aligned} (2c)\quad 2\cdot\nu(\underset{\sim}{y}) &= \underset{\sim}{y}_Q^t A_{QQ}\underset{\sim}{y}_Q - 2\underset{\sim}{y}_Q^t\underset{\sim}{b}_Q \\ &= -\underset{\sim}{b}_Q^t A_{QQ}^{-1}\underset{\sim}{b}_Q < (\underset{\sim}{x}_Q - A_{QQ}^{-1}\underset{\sim}{b}_Q)^t\, A_{QQ}(\underset{\sim}{x}_Q - A_{QQ}^{-1}\underset{\sim}{b}_Q) - \underset{\sim}{b}_Q^t A_{QQ}^{-1}\underset{\sim}{b}_Q \\ &= \underset{\sim}{x}_Q^t A_{QQ}\underset{\sim}{x}_Q - 2\underset{\sim}{x}_Q^t\underset{\sim}{b}_Q \\ &= 2\cdot\nu(\underset{\sim}{x}). \qquad \square \end{aligned}$$

Let Z denote the index set of the constrained (zero) components of the iterate vector and P the index set of the unconstrained (positive) components. We can describe the lower bound algorithm in [2] as follows.

ALGORITHM 2.1 (Chandrasekaran's method). Let

$$P := \{i \in N | b_i > 0\} \quad \text{and} \quad Z := \bar{P}.$$

Define $\underset{\sim}{x}$ so that $\underset{\sim}{x}_P = A_{PP}^{-1}\underset{\sim}{b}_P$ and $\underset{\sim}{x}_Z = \underset{\sim}{0}$. Let J be a nonempty set.

Repeat until either set Z or J is empty:

1. Compute $\underset{\sim}{g}_Z := A_{ZP}\underset{\sim}{x}_P - \underset{\sim}{b}_Z$
2. Let $J := \{j \in Z | g_j < 0\}$.
3. If set J is nonempty, then
   (a) Let $P := P \cup J$ and $Z := Z \setminus J$.
   (b) Reconstruct $\underset{\sim}{x}$ so that $\underset{\sim}{x}_P = A_{PP}^{-1}\underset{\sim}{b}_P$ and $\underset{\sim}{x}_Z = \underset{\sim}{0}$. □

Part (2a) of Theorem 2.1 says that the iterate vector is nondecreasing in value. Thus, a positive component stays positive and no element leaves the index set P. We enlarge set P when we release variables with negative gradients from their constraints. If no such variables exist, then we have computed the solution (Part (1) of Theorem 2.1). Hence Algorithm 2.1 always terminates. It is a descent method by Part (2c) of the same theorem.

Most of the work in an iteration of Algorithm 2.1 is spent in the solution of a matrix equation. We can use an iterative method. The conjugate gradient method (cf. [9]) or its generalized variant (cf. [3]) is particularly effective for solving a large sparse set of linear equations when the coefficient matrix is symmetric and positive definite. Furthermore, we can construct a good starting vector from the solution to the matrix equation in the previous iteration. A straightforward strategy is to place zeros in the new positions. In many cases, the quadratic program (1.2) arises from an approximation to a continuous problem and some sort of interpolation and extrapolation schemes can be successfully exploited.

It is obvious that we do not need to solve the matrix equations to full accuracy in any but the last iteration. This gives another reason for using an iterative method instead of a direct method for solving the matrix equations.

The intent of solving the matrix equation is to go from the minimum on one subspace S_1 to the minimum on a larger subspace S_2 that contains S_1. But the cost involved can be substantial. Thus, we are interested in the possibility of using some simple computing process that lets us enlarge the index set P of unconstrained variables with very little work. It will take more iterations to determine the final index set P, but the total cost may be lower because of the smaller amount of work per iteration.

Suppose we are at the start of an iteration of Algorithm 2.1. Let $\underset{\sim}{x}$ and $g(\underset{\sim}{x})$ be the iterate and gradient vectors, respectively. Let P be the index set of positive components of $\underset{\sim}{x}$ and J be the index set of zero components with negative gradients, i.e.,

$$P := \{j \in Z_n | x_j > 0\} \quad \text{and} \quad J := \{j \in \bar{P} | g_j < 0\} .$$

Assume set J is nonempty. Let $Q := P \cup J$. We construct the new iterate $\hat{\underset{\sim}{x}}$ so that $\hat{\underset{\sim}{x}}_{\bar{Q}} = \underset{\sim}{x}_{\bar{Q}}$ and $\hat{\underset{\sim}{x}}_Q = \underset{\sim}{y}$, where $\underset{\sim}{y}$ solves the matrix equation

$$A_{QQ}\underset{\sim}{y} = \underset{\sim}{b}_Q . \tag{2.1}$$

Let us consider a cheaper way to construct $\hat{\underset{\sim}{x}}$: update only the j-th components, where $j \in J$. In other words, $\hat{\underset{\sim}{x}}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\hat{\underset{\sim}{x}}_J = \underset{\sim}{z}$, where $\underset{\sim}{z}$ solves the matrix equation

$$B\underset{\sim}{z} = -\underset{\sim}{g}_J \tag{2.2}$$

and B is an M-matrix of order equal to the size of J. Thus $\underset{\sim}{z} > \underset{\sim}{0}$. We can choose B as A_{JJ}, but a better strategy is to choose B so that the matrix equation (2.2) is easily solvable. Regardless of which B we choose, it is very important that the new iterate $\hat{\underset{\sim}{x}}$ satisfies the inequality

$$\hat{\underset{\sim}{x}}_Q \leq A_{QQ}^{-1}\underset{\sim}{b}_Q . \tag{2.3}$$

If (2.3) does not hold, then the gradients of some constrained variables

may have decreased too much in value (cf. Lemma 2.3) and those variables may be erroneously released from their constraints.

We propose to choose B as C_{JJ}, where C is an $n \times n$ M-matrix so that $C \geq A$. The next theorem shows that the inequality (2.3) always holds for these choices of B and C. We now give a simple way to construct the matrix C.

LEMMA 2.4 [11, p. 85]. Let A be an M-matrix, and let C be any matrix obtained from A by setting certain off-diagonal entries of the matrix to zero. Then, C is also an M-matrix.

A possible choice for C is therefore a (block) diagonal matrix with the same (block) diagonal part as A.

ASSUMPTION 2.1. We have chosen an M-matrix C such that $C \geq A$ and that matrix equations with C or a principal submatrix of C as the coefficient matrix are easily solvable.

LEMMA 2.5. Suppose there is a vector $\underset{\sim}{x} \geq \underset{\sim}{0}$ such that $\underset{\sim}{x}_J = \underset{\sim}{0}$ and $\underset{\sim}{g}_J(\underset{\sim}{x}) < \underset{\sim}{0}$ for some nonempty subset J of N. Construct a vector $\underset{\sim}{y}$ so that $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\underset{\sim}{y}_J = -C_{JJ}^{-1}\underset{\sim}{g}_J(\underset{\sim}{x})$. Then $\underset{\sim}{g}_J(\underset{\sim}{y}) \leq \underset{\sim}{0}$.

Proof. C_{JJ} is an M-matrix by Lemma 1.2. Hence $C_{JJ}^{-1} \geq 0$. But $A_{JJ} \leq C_{JJ}$. Thus,

$$A_{JJ}C_{JJ}^{-1} \leq C_{JJ}C_{JJ}^{-1} = I ,$$

which implies

$$I - A_{JJ}C_{JJ}^{-1} \geq 0 .$$

Finally,

$$\begin{aligned} g_J(y) &= A_{JJ}y_J + A_{J\bar{J}}y_{\bar{J}} - b_J \\ &= -A_{JJ}C_{JJ}^{-1}g_J(x) + A_{J\bar{J}}x_{\bar{J}} - b_J \\ &= -A_{JJ}C_{JJ}^{-1}g_J(x) + g_J(x) \qquad \text{because} \quad x_J = 0 \\ &= (I - A_{JJ}C_{JJ}^{-1})\, g_J(x) \\ &\leq 0 \qquad \text{because} \quad g_J(x) < 0 \,. \end{aligned}$$ □

LEMMA 2.6. Suppose there is a nonempty subset J of N and two vectors x and y such that $x_{\bar{J}} = 0$, $g_J(x) = 0$, $y_{\bar{J}} = 0$ and $g_J(y) \leq 0$. Then $x_J \geq y_J$.

Proof.

$$\begin{aligned} A_{JJ}(x_J - y_J) &= A_{JJ}x_J - b_J - (A_{JJ}y_J - b_J) \\ &= g_J(x) - g_J(y) \\ &\geq 0 \,. \end{aligned}$$

Hence

$$x_J - y_J \geq 0 \,.$$ □

THEOREM 2.2. Assume we have a vector $x \geq 0$. Let $P := \{j \in N \mid x_j > 0\}$ and $J := \{j \in \bar{P} \mid g_j(x) < 0\}$. Suppose that both index sets P and J are nonempty, and that $g_P(x) \leq 0$. Construct a vector y such that $y_J = -C_{JJ}^{-1}g_J(x)$ and $y_{\bar{J}} = x_{\bar{J}}$. Let $K := \bar{P} \setminus J$. Then

(1) $g_P(\underset{\sim}{y}) \leq g_P(\underset{\sim}{x}) \leq \underset{\sim}{0}$, $g_J(\underset{\sim}{y}) \leq \underset{\sim}{0}$ and $g_K(\underset{\sim}{y}) \leq g_K(\underset{\sim}{x})$.

(2) $\underset{\sim}{z}_Q \leq A_{QQ}^{-1}\underset{\sim}{b}_Q$, where $Q = P \cup J$.

(3) $v(\underset{\sim}{y}) < v(\underset{\sim}{x})$.

Proof.

(1) By Lemma 2.5, $g_J(\underset{\sim}{y}) \leq \underset{\sim}{0}$. Since $\underset{\sim}{y}_J > \underset{\sim}{0} = \underset{\sim}{x}_J$ and $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$, we get $g_{\bar{J}}(\underset{\sim}{y}) \leq g_{\bar{J}}(\underset{\sim}{x})$ by Lemma 2.3. But $\bar{J} = P \cup K$.

(2) Define a vector $\underset{\sim}{z}$ so that $\underset{\sim}{z}_Q = A_{QQ}^{-1}\underset{\sim}{b}_Q$ and $\underset{\sim}{z}_{\bar{Q}} = \underset{\sim}{y}_{\bar{Q}} = \underset{\sim}{0}$. Hence $g_Q(\underset{\sim}{z}) = \underset{\sim}{0}$. From Part (1), $g_Q(\underset{\sim}{y}) \leq \underset{\sim}{0}$. Thus, $\underset{\sim}{z}_Q \geq \underset{\sim}{y}_Q$ by Lemma 2.6.

(3) $$v(\underset{\sim}{y}) - v(\underset{\sim}{x}) = \tfrac{1}{2}\underset{\sim}{y}_J^t A_{JJ}\underset{\sim}{y}_J + \underset{\sim}{y}_J^t A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_J^t\underset{\sim}{b}_J$$

$$= \tfrac{1}{2}\underset{\sim}{y}_J^t A_{JJ}(-C_{JJ}^{-1}g_J(\underset{\sim}{x})) + \underset{\sim}{y}_J^t(A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J)$$

$$= \underset{\sim}{y}_J^t(I - \tfrac{1}{2}A_{JJ}C_{JJ}^{-1})\, g_J(\underset{\sim}{x}) \ .$$

We have shown that $I - A_{JJ}C_{JJ}^{-1} \geq 0$. Hence $(I - (A_{JJ}C_{JJ}^{-1})/2) \geq 0$ and equality is not possible. Since $\underset{\sim}{y}_J > \underset{\sim}{0}$ and $g_J(\underset{\sim}{x}) < \underset{\sim}{0}$, we get $v(\underset{\sim}{y}) < v(\underset{\sim}{x})$. □

We now have the tool to modify Algorithm 2.1. Instead of going from one constrained minimum to another through solving matrix equations involving the unconstrained variables, we take descent steps through solving very simple matrix equations involving only those just released variables. We call our technique "partial updating". It lowers the

gradients of some constrained variables and we can again release those variables with negative gradients from the lower bound using the same technique. The process is repeated until the gradients of the constrained variables are all nonnegative. Then we compute the constrained minimum by solving the matrix equation

$$A_{PP}\underset{\sim}{y} = \underset{\sim}{b}_P \qquad (2.4)$$

and defining the new iterate $\underset{\sim}{x}$ so that $\underset{\sim}{x}_P = \underset{\sim}{y}$ and $\underset{\sim}{x}_{\bar{P}} = \underset{\sim}{0}$. The iterate vector from the "partial updating" is usually a very good initial vector for solving (2.4) using an iterative method.

ALGORITHM 2.2. (Modified Chandrasekaran's method). Let

$$\underset{\sim}{x} := \underset{\sim}{0}\ ,\ \underset{\sim}{g} := -\underset{\sim}{b} \text{ and}$$

$J := \{j \in N | g_j < 0\}$. Set $P := J$ and $Z := \bar{J}$.

Repeat until either set Z or J is empty:

1. Repeat until either set Z or J is empty:
   (a) Compute $\underset{\sim}{x}_J := -C_{JJ}^{-1}\underset{\sim}{g}_J$.
   (b) Update $\underset{\sim}{g}_Z := \underset{\sim}{g}_Z + A_{ZJ}\underset{\sim}{x}_J$.
   (c) Redefine $J := \{j \in Z | g_j < 0\}$.
   (d) Let $P := P \cup J$ and $Z := Z \setminus J$.
2. Compute $\underset{\sim}{x}_P := A_{PP}^{-1}\underset{\sim}{b}_P$.
3. Compute $\underset{\sim}{g}_Z := A_{ZP}\underset{\sim}{x}_P - \underset{\sim}{b}_Z$.
4. Redefine $J := \{j \in Z | g_j < 0\}$.
5. Let $P := P \cup J$ and $Z := Z \setminus J$. □

We can easily extend Algorithm 2.2 to solve the general lower-bound quadratic program

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (2.5)$$

$$\text{subject to} \quad \underset{\sim}{x} \geq \underset{\sim}{c} .$$

ALGORITHM 2.3. Let

$$\underset{\sim}{x} := \underset{\sim}{c} , \quad \underset{\sim}{g} := A\underset{\sim}{c} - \underset{\sim}{b} \quad \text{and}$$

$J := \{j \in N \mid g_j < 0\}$. Set $P := J$ and $Z := \bar{J}$.

Repeat until either set Z or J is empty:

1. Repeat until either set Z or J is empty:
   (a) Compute $\underset{\sim}{x}_J := \underset{\sim}{c}_J - C_{JJ}^{-1} \underset{\sim}{g}_J$.
   (b) Update $\underset{\sim}{g}_Z := \underset{\sim}{g}_Z + A_{ZJ} \underset{\sim}{x}_J$.
   (c) Redefine $J := \{j \in Z \mid g_j < 0\}$.
   (d) Let $P := P \cup J$ and $Z := Z \setminus J$.
2. Compute $\underset{\sim}{x}_P := A_{PP}^{-1}(\underset{\sim}{b}_P - A_{PZ} \underset{\sim}{c}_Z)$.
3. Compute $\underset{\sim}{g}_Z := A_{ZP} \underset{\sim}{x}_P + A_{ZZ} \underset{\sim}{c}_Z - \underset{\sim}{b}_Z$.
4. Redefine $J := \{j \in Z \mid g_j < 0\}$.
5. Let $P := P \cup J$ and $Z := Z \setminus J$. □

3. Lower and Upper Bounds Problem

In this section, we consider the lower and upper bounds quadratic program

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.1)$$

$$\text{subject to} \quad \underset{\sim}{c} \leq \underset{\sim}{x} \leq \underset{\sim}{d} .$$

We introduce an index set U denoting the variables constrained at the upper bound $\underset{\sim}{d}$. The index sets P and Z denote the unconstrained variables and the variables constrained at the lower bound $\underset{\sim}{c}$, respectively.

Our lower and upper bounds algorithm starts with the iterate vector $\underset{\sim}{x}$ at the upper bound $\underset{\sim}{d}$ and the index set U equal to N. We examine the gradient vector $\underset{\sim}{g}(\underset{\sim}{x}) \equiv A\underset{\sim}{x} - \underset{\sim}{b}$, and release those variables whose gradients are positive. Let P denote the just released variables and let $U := \bar{P}$. We solve the lower bound quadratic program

$$\min_{\underset{\sim}{y}} \frac{1}{2} \underset{\sim}{y}^t A_{PP} \underset{\sim}{y} - \underset{\sim}{y}^t (\underset{\sim}{b}_P - A_{PU} \underset{\sim}{d}_U) \qquad (3.1)$$

$$\text{subject to} \quad \underset{\sim}{y} \geq \underset{\sim}{c}_P .$$

The iterate vector $\underset{\sim}{x}$ is redefined so that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$ and $\underset{\sim}{x}_P = \underset{\sim}{y}$, the solution to (3.1). The indices of the components of $\underset{\sim}{x}$ at the lower bound are moved from set P to set Z. We start a new iteration by releasing those variables at the upper bound whose gradients have now become positive.

THEOREM 3.1. Suppose we have a vector $\underset{\sim}{x}$ and two nonempty index sets U and P partitioning N such that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$ and $\underset{\sim}{x}_P = \underset{\sim}{y} < \underset{\sim}{c}_P$, where $\underset{\sim}{y}$ solves (3.1). Let $J := \{j \in U | g_j(\underset{\sim}{x}) > 0\}$.

If the index set J is empty, then

1. $\underset{\sim}{x}$ solves the lower and upper bounds quadratic program (1.1), else
2. let $P := \{j \in P | x_j > c_j\}$, $Z := \{j \in P | x_j = c_j\}$, $K := P \cup J$ and $L := U \setminus J$. Define a vector $\underset{\sim}{y}$ so that $\underset{\sim}{y}_L = \underset{\sim}{d}_L$, $\underset{\sim}{y}_Z = \underset{\sim}{c}_Z$ and $\underset{\sim}{y}_K = \underset{\sim}{p}$, where $\underset{\sim}{p}$ solves the lower bound quadratic program

$$\min_{\underset{\sim}{p}} \frac{1}{2} \underset{\sim}{p}^t A_{KK} \underset{\sim}{p} - \underset{\sim}{p}^t (\underset{\sim}{b}_K - A_{KL} \underset{\sim}{d}_L)$$

subject to $\underset{\sim}{p} \geq \underset{\sim}{c}_K$

Then

(a) $\underset{\sim}{y}_J < \underset{\sim}{d}_J$ and $\underset{\sim}{y}_P \leq \underset{\sim}{x}_P$.

(b) $\underset{\sim}{g}_L(\underset{\sim}{y}) \geq \underset{\sim}{g}_L(\underset{\sim}{x})$ and $\underset{\sim}{g}_Z(\underset{\sim}{y}) \geq \underset{\sim}{g}_Z(\underset{\sim}{x}) \geq \underset{\sim}{0}$.

(c) $v(\underset{\sim}{y}) < v(\underset{\sim}{x})$.

Proof.

(1) The optimality conditions (1.5) are satisfied because $\underset{\sim}{g}_U(\underset{\sim}{x}) \leq \underset{\sim}{0}$ and $\underset{\sim}{x}_P$ solves (3.1).

(2a) The inequalities hold trivially for those components of $\underset{\sim}{y}_K$ that are at the lower bound. The gradients of the other components equal zero and we can prove the inequalities using a technique similar to that of Part (2a) of Theorem 2.1.

(2b) We have shown in Part (2a) that $\underset{\sim}{y}_K \leq \underset{\sim}{x}_K$. As $\underset{\sim}{y}_{\bar{K}} = \underset{\sim}{x}_{\bar{K}}$ by construction, we get $\underset{\sim}{g}_{\bar{K}}(\underset{\sim}{y}) \geq \underset{\sim}{g}_{\bar{K}}(\underset{\sim}{x})$ from Lemma 2.3. But $\bar{K} = L \cup Z$.

(2c) Since $g_Z(y) \geq 0$ by Part (2b), the vector $q = y_{K \cup Z}$ solves the lower bound quadratic program

$$\min_{q} \frac{1}{2} q^t A_{QQ} q - q^t (b_Q - A_{QL} d_L)$$

$$\text{subject to} \quad q \geq c_Q ,$$

where $Q = K \cup Z$. But $L = Q$, $y_L = x_L$ and $y \neq x$. Hence $v(y) < v(x)$. □

ALGORITHM 3.1. (Pang's method). Set $x := d$ and compute $g := Ad - b$. Let $J := \{j \in N | g_j > 0\}$, $P := J$, $U := \bar{J}$ and $Z := \emptyset$.

Repeat until set U or P or J is empty:

1. Reconstruct x so that $x_U = d_U$, $x_Z = c_Z$ and $x_P = y_P$, where y solves the lower bound quadratic program (3.1) (we may use Algorithm 2.3).
2. Set $K := \{k \in P | x_K = c_K\}$. Let $P := P \setminus K$ and $Z := Z \cup K$.
3. Compute $g_U := A_{UU} d_U + A_{UP} x_P + A_{UZ} c_Z - b_U$.
4. Redefine $J := \{j \in U | g_j > 0\}$. Set $P := P \cup J$ and $U := U \setminus J$. □

Part (2a) of Theorem 3.1 says that the iterate vector is non-increasing in value. Hence once a variable leaves its upper bound it never returns, and once a variable enters its lower bound it never exits. Since we release variables from their upper bounds when their gradients become positive, there is a flow of variables from the upper bound to the unconstrained region, and then to the lower bound. If

there is no outflow of variables from the upper bound, then the iterate vector is the desired solution (Part (1) of Theorem 3.1). Thus, Algorithm 3.1 always terminates. It is a descent method by Part (2c) of the same theorem. Algorithm 3.1 is essentially the algorithm proposed by Pang [8], except that he releases variables from their upper bounds when their gradients are nonnegative. The difference is so minor in real arithmetic that we expect both algorithms to produce identical results for almost all problems. Indeed, the two algorithms behaved identically in all our test examples.

We are interested in a "partial updating" technique that is similar to the one in the previous section. Such a technique may save many expensive solutions of lower bound quadratic programs. However, the presence of a lower bound restricts our choice of a "partial updating" matrix to a diagonal matrix. Recall that a positive definite matrix has a positive diagonal.

ASSUMPTION 3.1. We have chosen a diagonal matrix D such that $D \geq A$.

NOTATION. If $\underset{\sim}{v} = (v_1, v_2, \ldots, v_p)^t$, then $\underset{\sim}{v}_+ = (w_1, w_2, \ldots, w_p)^t$, where $w_j = \max(v_j, 0)$ for $j = 1,2,\ldots,p$.

LEMMA 3.2. Suppose there is a vector $\underset{\sim}{x}$ with $\underset{\sim}{c} \leq \underset{\sim}{x} \leq \underset{\sim}{d}$ such that $\underset{\sim}{x}_J = \underset{\sim}{d}_J$ and $g_J(\underset{\sim}{x}) > 0$ for some nonempty subset J of N. Construct a vector $\underset{\sim}{y}$ so that $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\underset{\sim}{y}_J = \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - D_{JJ}^{-1} g_J(\underset{\sim}{x})]_+$. Then $g_J(\underset{\sim}{y}) \geq 0$.

Proof. There exists a diagonal matrix $\hat{D} \geq D_{JJ}$ such that

$$\underset{\sim}{y}_J = \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - \hat{D}^{-1} g_J(\underset{\sim}{x})] .$$

Hence $\hat{D} \geq A_{II}$ and this lemma can be proved in the same way as Lemma 2.5. □

Note that Lemma 3.2 may not hold if we replace D by a non-diagonal M-matrix E with $E \geq A$.

THEOREM 3.2. Suppose we have a vector $\underset{\sim}{x}$ with $\underset{\sim}{c} < \underset{\sim}{x} \leq \underset{\sim}{d}$ and two nonempty index sets U and P partitioning N such that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$, $\underset{\sim}{x}_P < \underset{\sim}{d}_P$ and $g_P(\underset{\sim}{x}) \geq 0$. Let $J := \{j \in U | g_j(\underset{\sim}{x}) > 0\}$ and assume that it is not empty. Construct a vector $\underset{\sim}{y}$ such that $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\underset{\sim}{y}_J = \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - D_{JJ}^{-1} g_J(\underset{\sim}{x})]_+$. Let $K := U \setminus J$. Then

(1) $g_K(\underset{\sim}{y}) \geq g_K(\underset{\sim}{x})$, $g_J(\underset{\sim}{y}) \geq 0$ and $g_P(\underset{\sim}{y}) \geq g_P(\underset{\sim}{x}) \geq \underset{\sim}{0}$.

(2) $\underset{\sim}{y}_P \geq \underset{\sim}{g}$, where $\underset{\sim}{g}$ solves the lower bound quadratic program (3.1).

(3) $\nu(\underset{\sim}{y}) < \nu(\underset{\sim}{x})$.

Proof.

(1) By Lemma 3.2, $g_J(\underset{\sim}{y}) \geq \underset{\sim}{0}$. Since $\underset{\sim}{y}_J < \underset{\sim}{d}_J = \underset{\sim}{x}_J$ and $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$, we get $g_{\bar{J}}(\underset{\sim}{y}) \geq g_{\bar{J}}(\underset{\sim}{x})$ by Lemma 2.3. But $\bar{J} = K \cup P$.

(2) The inequality is trivial for those components of $\underset{\sim}{g}$ that are at the lower bound. Since the gradients of the other components of $\underset{\sim}{c}$ equal zero and $g_P(\underset{\sim}{y}) \geq \underset{\sim}{0}$ from Part (1), we can complete the proof using a technique similar to the one in Lemma 2.6.

(3) Let $\hat{D} \geq D_{JJ}$ be the diagonal matrix such that

$$\begin{aligned} y_J &= c_J + [d_J - c_J - \hat{D}^{-1} g_J(x)] \\ &= d_J - \hat{D}^{-1} g_J(x). \end{aligned}$$

Let $z = \hat{D}^{-1} g_J(x)$. Then

$$y_J = x_J - z$$

and

$$\begin{aligned} \nu(y) - \nu(x) &= \tfrac{1}{2} z^t A_{JJ} z - z^t A_{JJ} x_J - z^t A_{J\bar{J}} x_{\bar{J}} + z^t b_J \\ &= \tfrac{1}{2} z^t A_{JJ} \hat{D}^{-1} g_J(x) - z^t g_J(x) \\ &= -z^t (I - \tfrac{1}{2} A_{JJ} \hat{D}^{-1}) \, g_J(x) \, . \end{aligned}$$

It is easy to show that $I - (A_{JJ}\hat{D}^{-1})/2 \geq 0$, equality excluded. Since $z > 0$ and $g_J(x) > 0$, we get $\nu(y) - \nu(x) < 0$. □

ALGORITHM 3.2. (Modified Pang's method). Set $x := d$ and compute $g := Ad - b$. Let $J := \{j \in N | g_j > 0\}$, $P := J$, $U := \bar{J}$ and $Z := \emptyset$.

Repeat until set U or P or J is empty:

1. Repeat until set U or J is empty:

   (a) Update $x_J := c_J + [d_J - c_J - D_{JJ}^{-1} g_J]_+$.

   (b) Let $K := \{k \in J | x_k = c_k\}$, $Z := Z \cup K$ and $P := P \cup (J \setminus K)$.

   (c) Update $g_U := g_U - A_{UJ}(d_J - x_J)$.

   (d) Redefine $J := \{j \in U | g_j > 0\}$. Let $U := U \setminus J$.

2. Reconstruct $\underset{\sim}{x}$ so that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$, $\underset{\sim}{x}_Z = \underset{\sim}{c}_Z$ and $\underset{\sim}{x}_P = \underset{\sim}{y}$, where $\underset{\sim}{y}$ solves the lower bound quadratic program (3.1) (we may use Algorithm 2.3).

3. Set $K := \{k \in P | x_k = c_k\}$. Let $P := P \setminus K$ and $Z := Z \cup K$.

4. Compute $\underset{\sim}{g}_U := A_{UU}\underset{\sim}{d}_U + A_{UP}\underset{\sim}{x}_P + A_{UZ}\underset{\sim}{c}_Z - \underset{\sim}{b}_U$.

5. Redefine $J := \{j \in U | g_j > 0\}$. Let $P := P \cup J$ and $U := U \setminus J$. □

4. Problem with Non-finite Bounds

In this section, we consider the lower and upper bounds quadratic program

$$\min_{\underset{\sim}{x}} \frac{1}{2}\underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.1)$$

$$\text{subject to} \quad \underset{\sim}{c} \le \underset{\sim}{x} \le \underset{\sim}{d} ,$$

where some components of the bounds are infinite. Let us first assume that the upper bound is infinite. The next lemma indicates how finite lower bounds can be determined for all variables.

LEMMA 4.1. Consider the quadratic program (1.1) with $\underset{\sim}{d} = \underset{\sim}{\infty}$. Assume $\underset{\sim}{c}_J = -\underset{\sim}{\infty}$ and $\underset{\sim}{c}_{\bar{J}} > -\underset{\sim}{\infty}$ for some nonempty subset J of N. Let $\underset{\sim}{x}$ be the solution vector. Then

$$\underset{\sim}{x}_J \ge A_{JJ}^{-1}(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{c}_{\bar{J}}) .$$

Proof. Define a vector $\underset{\sim}{p}$ with $\underset{\sim}{p}_J = A_{JJ}^{-1}(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{c}_{\bar{J}})$ and $\underset{\sim}{p}_{\bar{J}} = \underset{\sim}{c}_{\bar{J}}$. By construction, $\underset{\sim}{g}_J(\underset{\sim}{p}) = \underset{\sim}{0}$. Define another vector $\underset{\sim}{q}$ with $\underset{\sim}{q}_J = \underset{\sim}{p}_J$ and $\underset{\sim}{q}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$. Since $\underset{\sim}{q}_{\bar{J}} \ge \underset{\sim}{p}_{\bar{J}}$, we get $\underset{\sim}{g}_J(\underset{\sim}{q}) \le \underset{\sim}{g}_J(\underset{\sim}{p})$ from Lemma 2.3.

Also $g_J(\underset{\sim}{x}) = \underset{\sim}{0}$ by the optimality conditions (1.5). Thus,

$$\begin{aligned} A_{JJ}(\underset{\sim}{x}_J - \underset{\sim}{p}_J) &= A_{JJ}\underset{\sim}{x}_J + A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J - (A_{JJ}\underset{\sim}{p}_J + A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J) \\ &= g_J(\underset{\sim}{x}) - g_J(\underset{\sim}{q}) \\ &\geq 0 . \end{aligned}$$

Hence

$$\underset{\sim}{x}_J - \underset{\sim}{p}_J \geq \underset{\sim}{0} . \qquad \square$$

We now suppose that some upper bound components are finite. The next lemma shows how we can compute finite upper bounds for all variables.

LEMMA 4.2. For problem (1.1), assume $\underset{\sim}{d}_J = \underset{\sim}{\infty}$ and $\underset{\sim}{d}_{\bar{J}} < \underset{\sim}{\infty}$ for some nonempty subset J of N. Let $\underset{\sim}{x}$ be the solution to (1.1) and $\underset{\sim}{y}$ be the solution to the lower bound quadratic program

$$\begin{aligned} \min_{\underset{\sim}{y}} \ & \frac{1}{2}\underset{\sim}{y}^t A_{JJ}\underset{\sim}{y} - \underset{\sim}{y}^t(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{d}_{\bar{J}}) \\ \text{subject to} \ & \underset{\sim}{y} \geq \underset{\sim}{c}_J . \end{aligned} \tag{4.1}$$

Then

$$\underset{\sim}{x}_J \leq \underset{\sim}{y} .$$

Proof. The inequality is trivial for those components of $\underset{\sim}{x}_J$ that are at the lower bound. Let $K := \{k \in J | x_k > c_k\}$. Then $g_K(\underset{\sim}{x}) = \underset{\sim}{0}$.

Now define a vector $\underset{\sim}{p}$ with $\underset{\sim}{p}_J = \underset{\sim}{y}$ and $\underset{\sim}{p}_{\bar{J}} = \underset{\sim}{d}_{\bar{J}}$. As $A_{JJ}\underset{\sim}{y} - (\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{d}_{\bar{J}}) \geq \underset{\sim}{0}$, we get that $g_J(\underset{\sim}{p}) \geq \underset{\sim}{0}$. Define another vector $\underset{\sim}{q}$ with $\underset{\sim}{q}_J = \underset{\sim}{y}$ and $\underset{\sim}{q}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$. Since $\underset{\sim}{q}_{\bar{J}} \leq \underset{\sim}{p}_{\bar{J}}$, we get $\underset{\sim}{g}_J'(\underset{\sim}{q}) \geq \underset{\sim}{g}_J'(\underset{\sim}{p})$ by Lemma 2.3. It follows that $g_K(\underset{\sim}{q}) \geq 0$.

Let $L := J \setminus K$. Then $\underset{\sim}{x}_L = \underset{\sim}{c}_L \leq \underset{\sim}{p}_L$, or equivalently, $\underset{\sim}{x}_L - \underset{\sim}{p}_L \leq \underset{\sim}{0}$. Thus,

$$\begin{aligned} A_{KK}(\underset{\sim}{x}_K - \underset{\sim}{p}_K) &= (A_{KK}\underset{\sim}{x}_K + A_{KL}\underset{\sim}{x}_L + A_{K\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_K) \\ &\quad -(A_{KK}\underset{\sim}{p}_K + A_{KL}\underset{\sim}{p}_L + A_{K\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_K) - A_{KL}(\underset{\sim}{x}_L - \underset{\sim}{p}_L) \\ &= g_K(\underset{\sim}{x}) - g_K(\underset{\sim}{q}) - A_{KL}(\underset{\sim}{x}_L - \underset{\sim}{p}_L) \\ &\leq 0 \quad \text{because } A_{KL} \leq 0 . \end{aligned}$$

Hence

$$\underset{\sim}{x}_K - \underset{\sim}{p}_K \leq \underset{\sim}{0} . \quad \square$$

Note that we can use Lemma 4.1 to compute finite lower bounds for all variables for the lower bound quadratic program (4.1), since $\underset{\sim}{d}_J = \underset{\sim}{\infty}$. We can then apply Algorithm 2.3 to solve (4.1). As an upper bound problem is reducible to a lower bound one by negating all its variables, the following lemmas are direct consequences of the preceding ones.

LEMMA 4.3. Consider the quadratic program (1.1) with $\underset{\sim}{c} = \underset{\sim}{-\infty}$. Assume that $\underset{\sim}{d}_J = \underset{\sim}{\infty}$ and $\underset{\sim}{d}_{\bar{J}} < \underset{\sim}{\infty}$ for some nonempty subset J of N. Let $\underset{\sim}{x}$ be the solution vector. Then

$$\underset{\sim}{x}_J \leq A_{JJ}^{-1}(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{d}_{\bar{J}}) . \quad \square$$

LEMMA 4.4. For problem (1.1), assume that $c_J = -\infty$ and $c_{\bar{J}} > -\infty$ for some nonempty subset J of N. Let x be the solution to (1.1) and χ be the solution to the upper bound quadratic program

$$\min_{\chi} \tfrac{1}{2} \chi^t A_{JJ} \chi - \chi^t (b_J - A_{J\bar{J}} c_{\bar{J}}) \tag{4.2}$$

$$\text{subject to} \quad \chi \le d_J \ .$$

Then

$$x_J \ge \chi \ . \qquad \square$$

The upper bound problem (4.2) can be trivially reduced to a lower bound problem, which can then be solved in the same manner as the quadratic program (4.1).

5. Numerical Examples

We have chosen four representative problems. The goal is to examine the effectiveness of our "partial updating" technique in reducing the number of matrix equation solutions. The conjugate gradient method has been selected as the linear system solver due to its general applicability to sparse equations. It is obvious that other choices are possible, e.g. the square-root-free Cholesky method is probably more appropriate for the tridiagonal systems of equations arising in Examples 1 and 3.

Our programs were written in FORTRAN and run on an IBM 370/168 computer at the Stanford Linear Accelerator Center. The codes were compiled with optimization level 2 of the H EXTENDED compiler.

EXAMPLE 1. Let us consider the linear complementarity problem

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.2)$$

$$\text{subject to} \qquad \underset{\sim}{x} \geq \underset{\sim}{0} .$$

The matrix A is chosen as

$$A = \begin{pmatrix} 2 & -1 & & \bigcirc \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 & -1 \\ \bigcirc & & & -1 & 2 \end{pmatrix}_{n \times n}$$

and the vector $\underset{\sim}{b}$ is generated by

$$b_i = 8 - 20 r_i \qquad \text{for} \quad i = 1,2,\ldots, n ,$$

where r_i is a random number from a uniform distribution in the open interval $(0,1)$. The "partial updating" matrix C is chosen as

$$C = \begin{pmatrix} 2 & & \bigcirc \\ & 2 & \\ & & \ddots \\ \bigcirc & & & 2 \end{pmatrix}_{n \times n} .$$

We define an iteration to be a sequence of "partial updates" followed by a solution of a matrix equation with coefficient matrix A_{PP}. The scalar $|P|$ gives the number of elements in the index set P at the end of an iteration. The other scalar u gives the number of "partial updates" in an iteration. Time is machine execution time in seconds.

n = 1000

Iteration	Algorithm 2.1 |P|	Algorithm 2.2 u, |P|
1	382	2,456
2	499	2,524
3	535	1,539
4	539	1,542
5	542	
Time	2.67	2.40

n = 1500

Iteration	Algorithm 2.1 |P|	Algorithm 2.2 u. |P|
1	585	2,692
2	763	2,806
3	824	3,838
4	837	2,848
5	845	2,851
6	848	1,853
7	850	
8	852	
9	853	
Time	8.43	6.49

n = 2000

Iteration	Algorithm 2.1 |P|	Algorithm 2.2 u, |P|
1	782	2,923
2	1020	2,1074
3	1098	3,1118
4	1119	2,1131
5	1128	2,1136
6	1133	1,1138
7	1135	
8	1137	
9	1138	
Time	11.79	9.92

EXAMPLE 2. This example is the same as the previous one, except that the matrix A has been chosen as the Laplacian 5-point finite difference operator:

$$A = \begin{pmatrix} B & -I & & \\ -I & B & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & B & -I \\ & & & -I & B \end{pmatrix}_{m^2 \times m^2},$$

where

$$B = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}_{m \times m}.$$

Let $n = m^2$. We consider two different choices of the "partial updating" matrix:

(1) $$C = D \equiv \begin{pmatrix} 4 & & & \\ & 4 & & \\ & & \ddots & \\ & & & 4 \end{pmatrix}_{n \times n},$$

and

(2) $$C = T \equiv \begin{pmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{pmatrix}_{n \times n}.$$

n = 300

Iteration	Algorithm 2.1	Algorithm 2.2 (C = D)	Algorithm 2.2 (C = T)
	|P|	u, |P|	u, |P|
1	349	2,416	2,416
2	450	2,462	2,462
3	463	1,465	1,465
4	465	1,466	1,466
5	466		
Time	7.54	6.73	6.70

n = 1500

Iteration	Algorithm 2.1	Algorithm 2.2 (C = D)	Algorithm 2.2 (C = T)
	|P|	u, |P|	u, |P|
1	689	2,730	2,730
2	793	2,823	1,828
3	837	1,842	1,842
4	842		
Time	17.08	15.29	15.41

We observe that both choices of C have produced essentially identical results. This is not surprising if we look at the index set J of the variables eligible for "partial updating." Rarely do we find two consecutive indices in J. Thus, the tridiagonal "updating" matrix reduces to a diagonal "updating" matrix in almost all cases.

We construct a different example with n = 900 and another choice of the vector $\underset{\sim}{b}$:

$$b_i = \begin{cases} -6r_i & \text{for } i = 301, 302, \ldots, 600 \\ 8r_i & \text{otherwise,} \end{cases}$$

where r_i is a random number from a uniform distribution in the open interval (0,1).

n = 900

Iteration	Algorithm 2.1 |P|	Algorithm 2.2 (C = D) u, |P|	Algorithm 2.2 (C = T) u, |P|
1	300	2,307	2,307
2	366	2,380	3,386
3	396	2,401	2,405
4	406	1,406	1,407
5	407	1,407	
Time	24.97	25.08	23.80

This is also an example where the "partial updating" technique is not particularly efficient due to the very special structure of the vector $\underset{\sim}{b}$.

EXAMPLE 3. We address the lower and upper bounds problem

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b}$$
$$\text{subject to} \quad \underset{\sim}{0} \leq \underset{\sim}{x} \leq \underset{\sim}{d} . \tag{5.1}$$

The matrix A is chosen as the tridiagonal matrix in Example 1. The vectors $\underset{\sim}{b}$ and $\underset{\sim}{d}$ are generated by

$$b_i = 11 - 20r_i, \qquad d_i = 7s_i, \qquad \text{for } i = 1,2,\ldots, n,$$

where r_i and s_i are random numbers from a uniform distribution in the open interval (0,1). The "partial updating" matrix C is the diagonal matrix with the same diagonal as A. We use Algorithm 2.1 (2.2) to solve the lower bound quadratic program in an iteration of Algorithm 3.1 (3.2).

An iteration is defined to be a sequence of "partial updates" followed by a solution of a lower bound quadratic program with the matrix A_{pp}. The 3-tuple $(|Z|,|P|,|U|)$ gives the number of elements in the index sets Z, P and U, respectively, at the end of an iteration. The scalar u gives the number of "partial updates" in the iteration.

n = 1000

Iteration	Algorithm 3.1 (|Z|, |P|, |U|)	Algorithm 3.2 u, (|Z|, |P|, |U|)
1	(234, 216, 550)	3, (307, 275, 418)
2	(310, 283, 407)	1, (311, 282, 397)
3	(313, 291, 396)	1, (313, 291, 396)
Time	1.80	1.63

n = 1500

Iteration	Algorithm 3.1 (|Z|, |P|, |U|)	Algorithm 3.2 u, (|Z|, |P|, |U|)
1	(366, 324, 810)	3, (489, 396, 615)
2	(492, 408, 600)	1, (495, 416, 589)
3	(495, 415, 590)	
4	(495, 416, 589)	
Time	4.13	2.09

n = 2000

Iteration	Algorithm 3.1 (|Z|, |P|, |U|)	Algorithm 3.2 u, (|Z|, |P|, |U|)
1	(452, 452, 1096)	3, (590, 560, 819)
2	(602, 579, 819)	2, (609, 602, 793)
3	(605, 602, 793)	1, (605, 603, 792)
4	(605, 603, 792)	
Time	7.51	5.01

EXAMPLE 4. This example is the same as Example 3, except that (1) the matrix A is chosen as the discretized Laplacian operator of Example 2, and (2) the upper bound $\underset{\sim}{d}$ is generated by

$$d_i = 3s_i \quad \text{for} \quad i = 1,2,\ldots,n .$$

We have changed the upper bound so that the three index sets are of roughly equal size in the final solution.

We use Algorithm 2.1 (2.2) to solve the lower bound quadratic program in an iteration of Algorithm 3.1 (3.2). Both choices: (1) $C = D$ and (2) $C = T$ (see Example 2) are considered for Algorithm 2.2.

$n = 900$

	Algorithm 3.1	Algorithm 3.2
Iteration	$(\|Z\|,\|P\|,\|U\|)$	$u,(\|Z\|,\|P\|,\|U\|)$
1	(274,168,458)	3,(307,223,370)
2	(325,277,298)	2,(325,294,281)
3	(325,294,281)	1,(325,296,279)
4	(325,296,279)	
Time	6.79	5.97 (C = D), 5.94 (C = T)

$n = 1600$

	Algorithm 3.1	Algorithm 3.2
Iteration	$(\|Z\|,\|P\|,\|U\|)$	$u,(\|Z\|,\|P\|,\|U\|)$
1	(478,277,845)	3,(552,430,618)
2	(582,487,531)	2,(585,524,497)
3	(585,520,495)	1,(585,526,489)
4	(585,526,489)	
Time	17.72	15.35 (C = D), 15.28 (C = T)

We note that the "partial updating" technique is particularly effective for examples 1 and 3 where the test matrix is tridiagonal (-1, 2, -1). The impressive gains are probably due to the fact that conjugate gradients is a terrible way to solve linear systems related to this matrix. We believe that the improvement would be much smaller if a more appropriate solver, such as square-root-free Cholesky, were used.

6. Conclusion

It is evident from the work of Cottle and Goheen [4] that the algorithms of Chandrasekaran [2] and Pang [8] can be very effective for solving lower and upper bounds quadratic programs associated with M-matrices. We have seen in the last section how our "partial updating" technique can cut the execution time of the two algorithms. Thus, our new schemes (Algorithms 2.2 and 3.2) are highly competitive for solving this important class of quadratic programming problems.

Acknowledgments

The authors acknowledge the generous help and support of Professor Gene Golub. We are also grateful to an anonymous referee for his valuable comments which have greatly improved this paper.

REFERENCES

[1] Cea, J., and Glowinski, R., "Sur des methodes d'optimisation par relaxation," RAIRO R-3 (1973), 5-32.

[2] Chandrasekaran, R., "A special case of the complementary pivot problem," Opsearch 7 (1970), 263-268.

[3] Concus, P., Golub. G.H., and O'Leary, D.P., "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," in Sparse Matrix Computations (J. R. Bunch and D.J. Rose, Ed.) Academic Press, New York (1976), 309-332.

[4] Cottle, R. and Goheen, M., "A special class of large quadratic programs," Report SOL-76-7, Systems Optimization Lab., Stanford University (1976).

[5] Cryer, C.W., "A survey of trial free-boundary methods for the numerical solution of free-boundary problems," MRC Report 1693, Mathematics Research Center, University of Wisconsin, Madison (1976).

[6] Fröberg, C.E., Introduction to Numerical Analysis, Addison-Wesley, Reading, Massachusetts (1965).

[7] Levati, G., Scarpini, F., and Volpi, G., "Sul trattemento numerico di alcuni problemi variazionali di tipo unilaterale," L.A.N. Pub. 82 (1974).

[8] Pang, J.S., "On a class of least-element complementarity problems," Report SOL-76-10, Systems Optimization Lab., Stanford University (1976).

[9] Reid, J.K., "On the method of conjugate gradients for the solution of large sparse systems of linear equations," in Large Sparse Sets of Linear Equations, (J.K. Reid, Ed.), Academic Press, New York (1971), 231-254.

[10] Scarpini, F., "Some algorithms solving the unilateral Dirichlet problems with two constraints," Calcolo 12 (1975), 113-149.

[11] Varga, R.S., Matrix Iterative Analysis, Prentice Hall, Englewood Cliffs, New Jersey (1962).