A Computational Method for the Indefinite Quadratic Programming Problem

James R. Bunch*
Department of Mathematics
University of California at San Diego
La Jolla, California 92093

and

Linda Kaufman
Bell Laboratories
Murray Hill, New Jersey 07974

Submitted by Hans Schneider

ABSTRACT

We present an algorithm for the quadratic programming problem of determining a local minimum of $f(\mathbf{x})=\frac{1}{2}\mathbf{x}^TQ\mathbf{x}+\mathbf{c}^T\mathbf{x}$ such that $A^T\mathbf{x}\geqslant\mathbf{b}$ where Q is a symmetric matrix which may not be positive definite. Our method combines the active constraint strategy of Murray with the Bunch-Kaufman algorithm for the stable decomposition of a symmetric matrix. Under the active constraint strategy one solves a sequence of equality constrained problems, the equality constraints being chosen from the inequality constraints defining the original problem. The sequence is chosen so that $f(\mathbf{x})$ continues to decrease and $\mathbf{x}$ remains feasible. Each equality constrained subproblem requires the solution of a linear system with the projected Hessian matrix, which is symmetric but not necessarily positive definite. The Bunch-Kaufman algorithm computes a decomposition which facilitates the stable determination of the solution to the linear system. The heart of this paper is a set of algorithms for updating the decomposition as the method progresses through the sequence of equality constrained problems. The algorithm has been implemented in a FORTRAN program, and a numerical example is given.

1. INTRODUCTION

The classical approach to minimizing a quadratic form $\mathbf{x}^TQ\mathbf{x}$ was the reduction of that form to diagonal form proposed by Lagrange in 1759 [12].

*This author's research was supported by NSF grants MCS75-06510 and MCS76-03139.

Lagrange's method was generalized slightly by Bunch and Parlett [4]. They reduced the quadratic form to a block diagonal form with diagonal blocks of order 1 and 2; the blocks of order 2 replaced the rotations in Lagrange's method. This gives a decomposition of a symmetric matrix Q,

$$Q = MDM^T, \tag{1.1}$$

where D is block diagonal with blocks of order 1 or 2, and M is the product of permutations and block elementary transformations.

Bunch [2] has developed a complete pivoting form of the algorithm to compute (1.1) which is as stable as Gaussian elimination with complete pivoting; Bunch and Kaufman [3] later developed a more efficient partial pivoting form which is as stable as Gaussian elimination with partial pivoting. These algorithms can be used on all symmetric matrices, whether positive definite or not.

The stable, efficient decomposition in (1.1) of symmetric matrices is a natural decomposition in several minimization contexts. In [15] and [16] it has been used for general unconstrained minimization when the Hessian of the function to be minimized is indefinite. These algorithms can certainly be generalized to handle linear inequality constraints using the decomposition to handle the projected Hessian. The decomposition is also natural in the simpler constrained problem discussed in this paper: solving quadratic programming problems having a symmetric, but not necessarily positive definite, quadratic form. This problem we define as follows: Given an $n \times n$ symmetric matrix Q, which need not be positive definite, an $n \times m$ matrix A, an n-vector $\mathbf{c}$, and an m-vector $\mathbf{b}$, find a local minimum of

$$f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x} \tag{1.2}$$

subject to the m linear inequality constraints

$$A^T \mathbf{x} \geqslant \mathbf{b}. \tag{1.3}$$

If Q is positive definite (having all positive eigenvalues), all local minima are global minima of the problem. If Q is not positive definite, the problem may have unbounded solutions and local minima which are not global minima.

Indefinite quadratic programming problems may arise naturally or as part of a nonlinear programming algorithm. Several people (see [6]) have proposed solving the general nonlinear problem by posing a sequence of problems involving second order approximations to the function and first order approximations to the constraints.

Our method follows the active constraint strategy of Murray [13]; in fact, when Q is positive definite, the two are identical. Under the active constraint strategy one solves a sequence of equality constrained problems, the equality constraints being chosen from the inequality constraints defining the original problem. (An inequality constraint which is being used as an equality constraint is said to be active.) The sequence is chosen so that the function continues to decrease. Only one of the equality constraints is changed each time.

If the solution of the current equality constrained problem violates one of the other inequality constraints, a new constraint is included. Of course, if the solution is unbounded, a new constraint is included if there is a bounded solution to the original problem. If the solution of the current equality constrained problem is feasible but is not a solution of the inequality constrained problem, then a constraint is deactivated, which extends the search space for a lower function value. Eventually, the solution of a particular equality constrained problem will be the solution of the inequality constrained problem originally posed. Thus the active constraint strategy poses two subproblems:

(A) Given any $n\times t$ $(t\leqslant n)$ matrix $\hat{A}$ and a vector $\hat{\mathbf{b}}$, minimize $f(\mathbf{x})$ such that

$$\hat{A}^T\mathbf{x}=\hat{\mathbf{b}}.$$

(B) If $\hat{A}^T$ are rows of A^T, and $\hat{\mathbf{b}}$ are the corresponding elements of $\mathbf{b}$, determine whether the solution of subproblem (A) also minimizes $f(\mathbf{x})$ subject to

$$A^T\mathbf{x}\geqslant\mathbf{b}.$$

In our algorithm only one constraint is deactivated at a time, and it is deactivated only after the current equality constrained problem has been solved. For some problems this strategy might entail wasting time searching in subspaces which do not contain a solution. On the other hand the strategy could prevent some unnecessary zigzagging between sets of active constraints and insures the termination of the algorithm in a finite number of steps. Since there are only a finite number of constraints and hence only a finite number of possible equality constrained problems, the method must eventually find a local minimum.

Gill and Murray [9] have recently proposed an algorithm for solving the indefinite quadratic programming problem. Their algorithm also uses an active constraint strategy, and when Q is positive definite, our algorithm and theirs follow the same path to the solution. The main difference between the two algorithms lies in the basic linear algebraic decomposition used to solve subproblem (A) when Q is indefinite. Their decomposition forces an added restriction on the initial point, which often means that the user of their algorithm must first solve a linear programming problem to find a vertex of the feasible region or add extra variables or bounds so that the initial point is at a vertex. Admittedly we require the solution of a preproblem when the user cannot specify a feasible point, but we do not insist that the initial point be a vertex of the prescribed region.

The principal aim of this paper is to indicate the use of the MDM^T decomposition in the context of the quadratic programming problem. Since the decomposition will prove useful for more general problems, it is important to demonstrate its utility in the quadratic programming problem. We propose a method as an alternative to the Gill-Murray approach, and realize that for some problems their algorithm might be more efficient, for some ours will be, and for still others both are equally satisfactory.

In Sec. 2 we outline the main algorithm, and in Sec. 3 we consider the special case when subproblem (A) does not have a bounded solution. In Secs. 4 and 5 we describe the updating of matrices needed to solve subproblem (A) when the equality constraint matrix is changed. Section 6 deals with the case where some of the constraints are simply upper and lower bounds on the variables, and Sec. 7 describes the initialization of the main algorithm. A numerical example is given in Sec. 8, and our conclusions in Sec. 9.

Throughout this paper uppercase letters denote matrices. The ith column of a matrix A is written as $\mathbf{a}_i$, and the (i,j)th element as a_{ij}. The transpose of A is denoted by A^T. The ith element of a vector $\mathbf{x}$ is written as x_i. Scalar quantities which are not elements of vectors and matrices are given Greek letters.

2. THE MAIN ALGORITHM

The algorithm proceeds by solving a sequence of equality constrained problems. At each stage one is given an $n\times t$ $(t<n)$ matrix $\hat{A}$ of full rank, a vector $\hat{\mathbf{b}}$, and a vector $\mathbf{x}$ such that

$$\hat{A}^T\mathbf{x}=\hat{\mathbf{b}}, \tag{2.1}$$

and one wishes to find $\mathbf{x}'$, the solution to subproblem (A).

Let Z be an $n\times(n-t)$ matrix which spans the null space of $\hat{A}^T$. Certainly if (2.1) is satisfied and

$$\hat{A}^T\mathbf{x}'=\hat{\mathbf{b}},$$

then there exist vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{v}'$ such that

$$\mathbf{x}=\hat{A}\mathbf{u}+Z\mathbf{v}$$

and

$$\mathbf{x}'=\hat{A}\mathbf{u}+Z\mathbf{v}'.$$

Thus

$$\mathbf{x}'=\mathbf{x}+Z(\mathbf{v}'-\mathbf{v}).$$

If $\mathbf{x}'$ solves subproblem (A), then $\mathbf{v}'$ must minimize

$$s(\mathbf{v})\equiv f(\mathbf{x})=\tfrac{1}{2}(\hat{A}\mathbf{u}+Z\mathbf{v})^T Q(\hat{A}\mathbf{u}+Z\mathbf{v})+\mathbf{c}^T(\hat{A}\mathbf{u}+Z\mathbf{v}).$$

Let $D=Z^TQZ$, the constrained Hessian.

If D is positive definite, then $s(\mathbf{v})$ is minimized ($\nabla s=0$) at

$$\begin{aligned}\mathbf{v}'&=-D^{-1}Z^T(Q\hat{A}\mathbf{u}+\mathbf{c})\\&=-D^{-1}Z^T(Q[\mathbf{x}-Z\mathbf{v}]+\mathbf{c})\\&=-D^{-1}Z^T(Q\mathbf{x}+\mathbf{c})+D^{-1}Z^TQZ\mathbf{v},\end{aligned}$$

which implies

$$\mathbf{v}'-\mathbf{v}=-D^{-1}Z^T(Q\mathbf{x}+\mathbf{c}).$$

Hence given an $\mathbf{x}$ satisfying (2.1), if Z^TQZ is positive definite, the solution to subproblem (A) is given by

$$\begin{aligned}\mathbf{x}'&=\mathbf{x}-ZD^{-1}Z^T(Q\mathbf{x}+\mathbf{c})\\&=\mathbf{x}+Z\mathbf{w},\end{aligned}\tag{2.2}$$

where $\mathbf{w}=-D^{-1}Z^T(Q\mathbf{x}+\mathbf{c})$.

The computation of $\mathbf{x}'$ by (2.2) obviously is based on the availability of Z and D; the efficiency of the computation is based on the structure of D. Murray's algorithm [13] computes Z such that D is always diagonal. When D

is not positive definite, his algorithm sometimes entails solving small eigenvalue problems. We choose Z such that D is block diagonal with blocks of order 1 and 2. Each 2×2 block corresponds to a positive-negative eigenvalue pair, so if D is positive definite, D is diagonal. The matrix Z can be easily determined using elementary transformations, and it can be proven that D is bounded (Bunch and Kaufman [3]).

If $\mathbf{x}'$ does not violate any of the inactive constraints (rows of A^T which are not in $\hat{A}^T$), then subproblem (B) should be solved to determine if the solution to the original quadratic programming problem has been found. If $\mathbf{u}$ is the solution of

$$\hat{A}\mathbf{u} = Q\mathbf{x}' + \mathbf{c} = \nabla f$$

and $u_j < 0$, then it is easy to show (see [8]) that $f(\mathbf{x})$ can be decreased further if the jth row is deleted from $\hat{A}^T$, since $\nabla f^T\mathbf{p} < 0$ where $A^T\mathbf{p} = \mathbf{e}_j$, the jth column of the identity matrix. In our algorithm the kth row is deleted from $\hat{A}^T$, where $u_k = \min\{u_j \leqslant 0\}$ and Z and D are modified accordingly.

If $\mathbf{x}'$ violates an inactive constraint, the new trial solution is set to

$$\mathbf{x} + \lambda Z\mathbf{w}, \tag{2.3}$$

where λ is the distance to the nearest inactive constraint along $Z\mathbf{w}$, and that constraint is activated. Note that for $\lambda \leqslant 1$, $f(\mathbf{x})$ decreases as λ increases.

When D is not positive definite, subproblem (A) does not have a bounded minimum, and it is possible to define $\mathbf{w}$ such that $f(\mathbf{x}+\lambda Z\mathbf{w})$ continues to decrease as λ increases. Methods for obtaining $\mathbf{w}$ are given in Sec. 3 and are dependent on the structure of D. If the original quadratic programming problem has only bounded solutions, an inactive constraint must be encountered and activated for some value of λ.

In summary, at each step of the algorithm one of six situations arises:

(1) D is positive definite, the solution to subproblem (A) is feasible, and it is the solution to the whole problem.

(2) D is positive definite, and the solution to subproblem (A) is feasible, but the function can be further decreased by dropping a constraint.

(3) D is not positive definite, and an unbounded but feasible solution is found to the whole problem.

(4) The proposed solution for subproblem (A) is infeasible, an inactive constraint is activated, and the new point $\mathbf{x}+\lambda Z\mathbf{w}$ is not an extreme point of the feasible region.

(5) An inactive constraint is activated, the new point $\mathbf{x}+\lambda Z\mathbf{w}$ is an extreme point of the feasible region, and the function can be further decreased by dropping a constraint.

(6) An inactive constraint is activated, the new point $\mathbf{x}+\lambda Z\mathbf{w}$ is an extreme point of the feasible region, and the algorithm has arrived at a solution to the whole problem.

For cases (2), (3), (4) and (6) a constraint is either activated or deactivated. Only in case (5) is one constraint activated and another deactivated. At the beginning of each iteration except the first, the matrix $\hat{A}$ is changed and the matrices Z and D are changed accordingly.

We shall now formalize the algorithm described above. Each step of the iteration is similar to Murray's method [13] except that we are using the Bunch-Kaufman algorithm instead of symmetric Gaussian elimination in the reduction. We now describe a typical iteration. Let $\hat{A}$ be an $n\times t$ and $\bar{A}$ be an $n\times(m-t)$ matrix; let $\hat{\mathbf{b}}$, $\bar{\mathbf{b}}$, and $\mathbf{x}$ be vectors of length t, $m-t$, and n, respectively, such that

$$A=[\hat{A} \,\vdots\, \bar{A}],\quad \mathbf{b}=\begin{bmatrix}\hat{\mathbf{b}}\\ \cdots \\ \bar{\mathbf{b}}\end{bmatrix},\quad \hat{A}^T\mathbf{x}=\hat{\mathbf{b}},\text{ and } \bar{A}^T\mathbf{x}\geqslant\bar{\mathbf{b}}.$$

Let Z be an $n\times(n-t)$ matrix such that $\hat{A}^TZ=0$ and $Z^TQZ=D$; D is an $(n-t)\times(n-t)$ block diagonal matrix with blocks of order 1 or 2, whose ith block is denoted D_i. Let $\mathbf{r}$ be the residual vector of length $m-t$: $\mathbf{r}=\bar{A}^T\mathbf{x}-\bar{\mathbf{b}}$.

Then the steps of one iteration are as follows.

Step 1:

Compute $\mathbf{g}=Q\mathbf{x}+\mathbf{c}$ and $\mathbf{y}=-Z^T\mathbf{g}$. If $\mathbf{y}=0$ and D is positive semidefinite, go to step 4.

The vector $\mathbf{y}$ is the negative of the projected gradient.

If D is positive definite, solve $D\mathbf{w}=\mathbf{y}$, else compute $\mathbf{w}$ as in Sec. 3.

Set $\mathbf{p}=Z\mathbf{w}$.

The vector $\mathbf{p}$ is the search direction and is orthogonal to $\hat{A}$.

Step 2:

Let

$$\gamma_q=\min_{t+1\leqslant j\leqslant m}\left\{\frac{-r_j}{\bar{\mathbf{a}}_j^T\mathbf{p}}:\bar{\mathbf{a}}_j^T\mathbf{p}<0\right\}.$$

Let $\mathbf{x}\leftarrow\mathbf{x}+\lambda\mathbf{p}$, where

$$\lambda=\begin{cases}\min(1,\gamma_q) & \text{if } D \text{ is positive definite,}\\ \gamma_q & \text{otherwise.}\end{cases}$$

Note that γ_q is the distance to the nearest inactive constraint along $\mathbf{p}$.
Let $\mathbf{r} \leftarrow \mathbf{r} + \lambda \bar{A}^T \mathbf{p}$.

Step 3:

If D is positive definite and $\lambda = 1$, go to step 4.
Add the qth column of $\bar{A}$ to $\hat{A}$, and change Z and D as in Sec. 4.
Let $r_j \leftarrow r_{j+1}$ for $j = q, q+1, \ldots, m-t-1$.
Increase t by 1.
If $t < n$, go to 1.

Step 4:

Solve $\hat{A}\mathbf{u} = \mathbf{g}$.
If $\mathbf{u} > 0$, the solution has been found. Otherwise let $u_s = \min_i u_i$.
Drop the sth constraint from $\hat{A}$, and change Z and D as in Sec. 5. Update the residual vector as follows:
Let $r_{m-t+1} \leftarrow 0$. Decrease t by 1 and go to step 1.
In step 4 it is necessary to solve $\hat{A}\mathbf{u} = \mathbf{g}$ for $\mathbf{u}$. The vector $\mathbf{u}$ may be found by solving $LL^T\mathbf{u} = \hat{A}^T(Q\mathbf{x} + \mathbf{c})$, where $LL^T = \hat{A}^T\hat{A}$ and L is lower triangular. However, $\mathbf{u}$ may also be found by solving $R\mathbf{u} = U(Q\mathbf{x} + \mathbf{c})$, where U has orthogonal columns,

$$U\hat{A} = R, \tag{2.5}$$

and R is nonsingular and upper triangular. The matrices Z, Q, U and the strictly upper triangular portion R' of R may be stored in two $n \times n$ arrays:

$$[\, U^T \,\vdots\, Z \,] \quad \text{and} \quad \begin{pmatrix} \diagdown & R' \\ Q & \diagdown \end{pmatrix}.$$

As $\hat{A}$ gains a column, U^T gains a column and Z loses a column. Similarly, as $\hat{A}$ loses a column, U^T loses a column and Z gains a column. In the following sections we shall assume that U and R are available. Methods for updating U and R when $\hat{A}$ is changed are discussed in [7].

The algorithm proposed by Gill and Murray [9] also uses the projected Hessian matrix Z^TQZ. However, they choose Z such that Z has orthogonal columns, and they update the LDL^T factorization of Z^TQZ where L is lower unit triangular and D is diagonal. When Z^TQZ is indefinite, this factorization may not exist. If Z^TQZ has only one negative eigenvalue, their algorithm manages to bypass the consequences of the indefiniteness of the projected Hessian, but they cannot handle a projected Hessian with more than one negative eigenvalue. Thus a user must provide an initial feasible point at which the projected Hessian has at most one negative eigenvalue. When the projected Hessian is positive definite, both our algorithm and theirs generate the same search direction. When the projected Hessian is indefinite, their method and any of those listed in Sec. 3 generate directions of negative curvature; however, the algorithm in Gill and Murray is tailored to use the fact that the projected Hessian has only one negative eigenvalue. Ours are all designed to allow a more general eigenstructure of the projected Hessian.

3. DETERMINING A SEARCH DIRECTION WHEN D IS NOT POSITIVE DEFINITE

When D is not positive definite, it is possible to determine a search direction $\mathbf{p}=Z\mathbf{w}$ such that $f(\mathbf{x}+\lambda\mathbf{p})$ will not increase as λ increases. Since we assume f has a bounded minimum, there is a value of λ for which $\mathbf{x}+\lambda\mathbf{p}$ first violates an inactive constraint.

To obtain the search direction we consider the fact that

$$\begin{aligned} f(\mathbf{x}+\lambda Z\mathbf{w}) &= f(\mathbf{x})+\lambda\mathbf{w}^TZ^T\nabla f(\mathbf{x})+\frac{\lambda^2}{2}\mathbf{w}^TZ^TQZ\mathbf{w} \\ &= f(\mathbf{x})-\lambda\mathbf{w}^T\mathbf{y}+\frac{\lambda^2}{2}\mathbf{w}^TD\mathbf{w}, \end{aligned} \tag{3.1}$$

where $\mathbf{y}=-Z^T\nabla f(\mathbf{x})$, as in step 1 of the main algorithm. Certainly $f(\mathbf{x}+\lambda Z\mathbf{w})\leqslant f(\mathbf{x})$ if

$$\mathbf{w}^T\mathbf{y}\geqslant 0 \tag{3.2a}$$

and

$$\mathbf{w}^TD\mathbf{w}\leqslant 0. \tag{3.2b}$$

If $\mathbf{w}$ is given by $\bar{D}^{-1}\mathbf{y}$ where $\bar{D}$ is a block diagonal matrix, these conditions become

$$\mathbf{y}^T\bar{D}^{-T}\mathbf{y}\geqslant 0 \tag{3.3a}$$

and

$$\mathbf{y}^T\bar{D}^{-T}D\bar{D}^{-1}\mathbf{y}\leqslant 0. \tag{3.3b}$$

Our problem reduces to finding $\bar{D}$ which satisfies (3.3), although in a few trivial cases we shall neglect $\bar{D}$ and compute a vector w explicitly which satisfies (3.2).

The explanation of the construction of $\bar{D}$ is complicated because some of the diagonal blocks of D can be 2×2. In order to simplify the situation, let D_j be the jth *block* of D, partition y into vectors of length 1 and 2 according to the block structure of D, and let y_j be the jth vector of y. Thus $\mathrm{y}^T D\mathrm{y} = \Sigma_i \mathrm{y}_j^T D_j \mathrm{y}_j$. The matrix $\bar{D}$ will have the same block structure as D, and its blocks will be called $\bar{D}_j$.

The following index sets will be useful in our construction:

J_0 = set of indices for which $D_j = 0$,
J_1 = set of indices for which

$$\mathrm{y}_j^T D_j^{-1} \mathrm{y}_j < 0, \qquad \mathrm{y}_j^T D_j \mathrm{y}_j < 0,$$

J_2 = set of indices for which

$$\mathrm{y}_j^T D_j^{-1} \mathrm{y}_j < 0, \qquad \mathrm{y}_j^T D_j \mathrm{y}_j \geqslant 0,$$

J_3 = set of indices for which

$$\mathrm{y}_j^T D_j^{-1} \mathrm{y}_j \geqslant 0, \qquad \mathrm{y}_j^T D_j \mathrm{y}_j < 0,$$

J_4 = set of indices for which

$$\mathrm{y}_j^T D_j \mathrm{y}_j > 0, \qquad D_j \text{ is } 1\times 1,$$

and
J_5 = set of indices for which

$$\mathrm{y}_j^T D_j \mathrm{y}_j \geqslant 0, \qquad \mathrm{y}_j^T D_j^{-1} \mathrm{y}_j \geqslant 0, \qquad D_j \text{ is } 2\times 2.$$

To avoid difficulties later, we shall immediately dispose of one exceptional case. If D is positive semidefinite and $y_j = 0$ whenever D_j is 0, then set

$$w_j = \begin{cases} y_j / D_j & \text{if } j \in J_4, \\ 0 & \text{if } j \in J_0. \end{cases}$$

and in steps 2 and 3 treat D as positive definite.

When one is not in the exceptional case, the matrix $\bar{D}$ and w can still be constructed in several ways. Three methods, which do not exhaust the possibilities, are given below. The first method tries to take the Newton direction in the space of positive curvature. The third ignores the space of positive curvature, while the second is a compromise direction.

METHOD 1. Let

$$\alpha = \sum_{j\in(J_1\cup J_2)} \mathbf{y}_j^T D_j^{-1}\mathbf{y}_j + \sum_{j\in J_3} \mathbf{y}_j^T D_j \mathbf{y}_j + \sum_{j\in J_5} (y_2^{(j)}, -y_1^{(j)}) D_j (y_2^{(j)}, -y_1^{(j)})^T$$

and

$$\gamma = \sum_{j\in J_4} \mathbf{y}_j^T D_j^{-1}\mathbf{y}_j.$$

Note that $\alpha \leqslant 0$ and $\gamma \geqslant 0$. If $\alpha < 0$, then let

$$\beta = \begin{cases} 1 & \text{if } \gamma = 0, \\ \dfrac{-\alpha(1+2\epsilon)}{\gamma} & \text{if } \gamma \neq 0, \end{cases}$$

where ϵ is the machine precision, and define $\bar{D}$ as follows:

$$\bar{D}_j = \begin{cases} \beta I & \text{if } j\in(J_0\cup J_3), \\ -\beta D_j & \text{if } j\in J_1, \\ -\beta D_j & \text{if } j\in J_2, \\ D_j & \text{if } j\in J_4, \\ \beta\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} & \text{if } j\in J_5. \end{cases}$$

The scalar γ isolates the positive portion of $\mathbf{y}^T\bar{D}^{-T}D\bar{D}^{-1}\mathbf{y}$. The scalar α isolates the negative contribution if β is 1. The quantity β is designed so that (3.3b) holds. The sign of the $\bar{D}_j$'s is determined so that (3.3a) holds.

If $\alpha = 0$, which could happen if all the nonzero $\bar{\mathbf{y}}_j$'s corresponded to positive semidefinite D_j's, the positive portion of $\mathbf{y}^T\bar{D}^{-T}D\bar{D}^{-1}\mathbf{y}$ cannot be canceled. In this case, which would occur if one started at a saddle point, our implementation uses Method 3; otherwise, it uses Method 1.

The construction of $\bar{D}_j$ when $j \in J_5$ deserves some explanation. Let

$$s_j = y_1^{(j)2} d_{22}^{(j)} + y_2^{(j)2} d_{11}^{(j)} - 2y_1^{(j)} y_2^{(j)} d_{12}^{(j)}.$$

Note that $y_j^T D_j^{-1} y_j = s_j / (d_{11}^{(j)} d_{22}^{(j)} - d_{12}^{(j)2})$. Because $y_j^T D_j^{-1} y_j \geqslant 0$ and because D_j is 2×2, $s_j < 0$. Since $y_j^T \bar{D}_j^{-T} D_j \bar{D}_j^{-1} y_j = s_j$ and $y_j^T \bar{D}_j^{-T} y_j = 0$ for $j \in J_5$, our choice for $\bar{D}_j$ for this case satisfies (3.3a) and (3.3b).

The method is expressly designed not to lead one toward a saddle point, because of the definition of $\bar{D}_j$ when $j \in (J_1 \cup J_2)$. For example, the problem, in which $\mathbf{x}$ must satisfy

$$x_2 \geqslant -20$$

and

$$-2x_1 + x_2 \geqslant -10,$$

with

$$\mathbf{c} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad Q = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

has a saddle point at $\mathbf{x} = (-1, +1)^T$ where $f(\mathbf{x}) = 2$. If one were at $(0,0)$, Method 1 would generate the direction $(-1, -(1+2\epsilon))^T$ and one would stop at about $\mathbf{x} = (-20, -20)^T$, giving $f(\mathbf{x}) \approx -40$. Thus the search direction is almost orthogonal to the direction leading to the saddle point. For the same problem, Method 2 and Method 3 below would generate $\mathbf{w} = (0, -1)^T$, producing the solution $\mathbf{x} = (0, -10)^T$, at which $f(\mathbf{x}) = -20$.

METHOD 2. Define $\bar{D}^{-1}$ as

$$\bar{D}_j^{-1} = \begin{cases} I & \text{if} \quad j \in (J_0 \cup J_1 \cup J_3), \\ 0 & \text{if} \quad j \in (J_2 \cup J_4 \cup J_5). \end{cases}$$

Obviously this method is simpler than Method 1, but it usually gives poorer results. Moreover, once it has been determined to which set j belongs, that information may as well be used more explicitly.

Certainly for this method one can construct examples for which $\mathbf{w} = \bar{D}^{-1}\mathbf{y}$ is identically 0. Therefore, Method 3 is suggested whenever $\mathbf{w} = \bar{D}^{-1}\mathbf{y} \equiv 0$.

METHOD 3. Here, we construct a vector w which satisfies (3.2). In our description, w is considered as a vector partitioned into vectors w_i according to the block structure of D. Define

$$w_i = \begin{cases} \operatorname{sign}(y_i) & \text{if } D_i \text{ is } 1\times 1 \text{ and } D_i \leqslant 0, \\ 0 & \text{if } D_i \text{ is } 1\times 1 \text{ and } D_i > 0, \\ \left(d_{12}^{(i)}, \delta\right)^T \operatorname{sign}\left(d_{12}^{(i)} y_1^{(i)} + \delta y_2^{(i)}\right) & \text{otherwise,} \end{cases}$$

where

$$\delta = \frac{d_{22}^{(i)} - d_{11}^{(i)}}{2} - \left\{ \left(\frac{d_{11}^{(i)} - d_{22}^{(i)}}{2} \right)^2 + \left(d_{12}^{(i)}\right)^2 \right\}^{1/2}$$

and

$$\operatorname{sign}(y) = \begin{cases} +1 & \text{if } y \geqslant 0, \\ -1 & \text{if } y < 0. \end{cases}$$

When D_i is 2×2, w_i is an eigenvector corresponding to the negative eigenvalue of D_i.

With Method 3, as long as D is not positive semidefinite, the quadratic form $w^T D w$ will be negative and hence $f(x+\lambda Zw) < f(x)$ for every nonzero λ even if $y = 0$, i.e., even if one is at a saddle point.

The advantage of this method is that there are no restrictions on it. The disadvantage is that it does not use very much of the available information about the function.

Without considering the proximity of the constraints, it is difficult to determine whether any direction of positive curvature should be included in w or whether any one direction of negative curvature should be emphasized. If all the constraints are rather distant, the largest decrease in f would be obtained if the last term in (3.1) were made as large as possible per unit step. Thus a direction similar to those given in Method 2 and Method 3 would be preferable. On the other hand, if all the constraints were close, a steepest descent direction in some appropriate metric would be preferable. If close initial points are assumed, then a direction like that in Method 1, which is Newton-like, avoids saddle points, and continues to decrease the function as one proceeds further in the direction, seems most desireable. However, because indefinite problems usually arise when a process is not well understood or under control, assumptions of closeness are probably not as viable in this case as they are in other minimization problems.

4. MATRIX UPDATE TECHNIQUES WHEN A CONSTRAINT IS ACTIVATED

Let $\hat{A}^T$ denote the current $t\times n$ matrix of active constraints. Let Z be an $n\times(n-t)$ matrix of rank t such that $\hat{A}^T\mathrm{Z}=0$ and $\mathrm{Z}^TQ\mathrm{Z}=D$, a block diagonal matrix of order $n-t$. Assume a vector a, independent of the columns of $\hat{A}$, is added to the constraint matrix. Let

$$A'^T \equiv \begin{bmatrix} \hat{A}^T \\ \mathbf{a}^T \end{bmatrix}.$$

In this section we shall outline the construction of D' and Z', the new D and Z matrices respectively. The constructions, in both this section and Sec. 5, are basically the two step process:

(1) update Z to $\bar{\mathrm{Z}}$ such that $A'^T\bar{\mathrm{Z}}=0$.

(2) update $\bar{\mathrm{Z}}^TQ\bar{\mathrm{Z}}$ to $D'=M^T\bar{\mathrm{Z}}^TQ\bar{\mathrm{Z}}M$ such that D' is block diagonal; $\mathrm{Z}'=\bar{\mathrm{Z}}M$.

The construction of Z' is based on the fact that

$$A'^T\mathrm{Z}=\begin{bmatrix} \hat{A}^T \\ \mathbf{a}^T \end{bmatrix}\mathrm{Z}=\begin{bmatrix} 0 \\ \mathbf{y}^T \end{bmatrix}, \qquad \text{where} \quad y_i=\mathbf{a}^T\mathbf{z}_i.$$

However, there are nonsingular matrices S of order $n-t$ such that

$$\mathbf{y}^T\mathrm{S}=\beta\mathbf{e}_{n-t}^T$$

for some constant β. For example, if $\mathbf{y}=0$, take $\beta=0$ and S to be any nonsingular matrix of order $n-t$. If $\mathbf{y}=y_{n-t}\mathbf{e}_{n-t}\neq 0$, take β to be arbitrary and $\mathrm{S}=\mathrm{diag}\{1,\ldots,1,\beta/y_{n-t}\}$.

If $\mathbf{y}\neq y_{n-t}\mathbf{e}_{n-t}$, then

(a) there exists an elementary reflector S such that $\mathbf{y}^T\mathrm{S}=\beta\mathbf{e}_{n-t}^T$, where $\beta=-\|\mathbf{y}\|_2$; this is always stable [14].

(b) there exists a permutation P such that $\mathbf{x}=P\mathbf{y}$ and $x_{n-t}\neq 0$, and there exists an elementary transformation E such that $E\mathbf{x}=x_{n-t}\mathbf{e}_{n-t}$, i.e., E zeros out the other elements of x; now take $\mathrm{S}=P^TE^T$. We can make S into a stable transformation by choosing P such that $|x_{n-t}|=\max_{1\leqslant i\leqslant n-t}|y_i|$.

Now, for any X of order $n-t-1$,

$$A'^T\mathrm{ZS}\begin{bmatrix} X \\ \cdot\;\cdot \\ 0^T \end{bmatrix}=0;$$

for simplicity, take $X = I_{n-t-1}$. However, the matrix

$$ZS\begin{bmatrix} I_{n-t-1} \\ \cdots\cdots \\ 0^T \end{bmatrix}$$

is a candidate for Z' only if

$$D' \equiv (I_{n-t-1} \,\vdots\, 0) S^T D S (I_{n-t-1} \,\vdots\, 0)^T$$

is block diagonal with blocks of order 1 or 2. If we construct S by (a) or (b) above, then D' will not necessarily have this block diagonal structure. With (a), D' would usually be full; with (b), the permutations would cause D' to fill in.

Two methods are given below for the construction of an S such that

$$\mathbf{y}^T S = \beta \mathbf{e}_{n-t}^T \tag{4.1a}$$

for some constant β,

$$Z' \equiv ZS(I_{n-t-1} \,\vdots\, 0)^T \tag{4.1b}$$

and

$$D' \equiv Z'^T Q Z' = (I_{n-t-1} \,\vdots\, 0) S^T D S (I_{n-t-1} \,\vdots\, 0)^T \tag{4.1c}$$

is block diagonal with blocks of order 1 and 2. Both methods use a sequence of permutations and elementary transformations, and both bound D' and take advantage of the block structure of D. The first method is easier to explain and to implement, while the second is more efficient and is the one we implemented in our codes.

METHOD i. This method first constructs a matrix S such that (4.1) holds and

$$D_1 = S^T D S$$

is 5-diagonal. The top $n-t-1$ submatrix of D_1 is then reduced to block diagonal form.

More specifically the algorithm is as follows.

(i.1) Partition y into vectors of length 1 and 2 according to the block structure of D, and call the jth vector y_j. Let α_j denote $\|\mathrm{y}_j\|_\infty$.

(i.2) Permute the columns of Z, the blocks of D, and the vectors y_j (and hence the elements of y) so that $\alpha_j \leqslant \alpha_{j+1}$. Also permute the elements of each y_j so that $|y_1^{(j)}| = \alpha_j$. Perform the same permutations on the columns of Z and on the rows and columns of D to form D_0, which is still block diagonal.

(i.3) Using permutations if necessary, apply $n-t-1$ stabilized elementary transformations in the planes $(1,2)$, $(2,3),\ldots,(n-t-1,n-t)$ to annihilate the first $n-t-1$ elements of y. Apply these transformations to the columns of Z and to the columns and rows of D_0 to form D_1.

If the matrix D_0 had the form

$$\begin{bmatrix} \mathrm{x} & \mathrm{x} & & & & & & \\ \mathrm{x} & \mathrm{x} & & & & & & \\ & & \mathrm{x} & \mathrm{x} & & & & \\ & & \mathrm{x} & \mathrm{x} & & & & \\ & & & & \mathrm{x} & & & \\ & & & & & \mathrm{x} & & \\ & & & & & & \mathrm{x} & \mathrm{x} \\ & & & & & & \mathrm{x} & \mathrm{x} \end{bmatrix},$$

then after step (i.3), the matrix D_1 would look like

$$\begin{bmatrix} \mathrm{x} & \mathrm{x} & & & & & & \\ \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & & & & \\ & \mathrm{x} & \mathrm{x} & \mathrm{x} & & & & \\ & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} & & & \\ & & & \mathrm{x} & \mathrm{x} & \mathrm{x} & & \\ & & & & \mathrm{x} & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ & & & & & \mathrm{x} & \mathrm{x} & \mathrm{x} \\ & & & & & \mathrm{x} & \mathrm{x} & \mathrm{x} \end{bmatrix}.$$

The permutations determined in step (i.2) were performed so that D_1 would have at most 5-diagonals.

(i.4) Use the algorithm of Bunch and Kaufman for 5-diagonal matrices [3] to reduce the top $n-t-1$ submatrix of D_1 to block diagonal form where the blocks are of order 1 and 2. This is the new matrix D'. Apply the transformations determined by the algorithm to the columns of Z. The first $n-t-1$ columns of the resulting matrix compose the matrix Z'.

The whole method requires at most $\frac{7}{2}(n-t)$ comparisons and $\frac{7}{2}n(n-t)$ multiplications and additions to determine Z'.

METHOD ii. This method eliminates steps (i.1) and (i.2) and combines steps (i.3) and (i.4) of Method i. It is based on the fact that the increase in the bandwidth of the D matrix in step (i.3) can be limited if unwanted off diagonal elements are annihilated as one goes along. In fact the largest dense matrix which must be considered is a 4×4 five-diagonal matrix. The block diagonality of the original D matrix is primarily responsible for the small matrices which are treated.

The method will be explained using snapshots of the zero structure of the relevant portion of the matrix D and vector y. All transformations are assumed to be stabilized elementary (i.e., using pivoting if necessary). The last step in most cases involves reducing a tridiagonal matrix of order 2, 3, or 4 to block diagonal form with blocks of order 1 or 2. This can be done without pivoting by using the algorithm outlined in Bunch and Kaufman [3]. In Fig. 1, + will denote the next element to be annihilated.

As the snapshots of Fig. 1 indicate, one or two elements are initially annihilated in the vector y. When the annihilating transformations are applied to the columns and rows of the D matrix, unwanted off diagonal elements appear, which are then zeroed.

Let $\tilde{D}$ and $\tilde{D}'$ be the parts of D and D', respectively, which are now under consideration; let $\tilde{y}$ be the corresponding part of y. In case 1, $\tilde{D}$ has two 1×1 blocks; a fill-in occurs when the element of y$'$ is annihilated, and the new $\tilde{D}'$ may be a 2×2 block or it may reduce to two 1×1 blocks. In case 2, $\tilde{D}$ has a 2×2 block; $\tilde{D}'$ may still be 2×2 or it may reduce to two 1×1 blocks. In case 3, $\tilde{D}$ has a 1×1 block followed by a 2×2 block; the worst situation here would be that $\tilde{D}$ would fill in and become a full 3×3 block; this would be reduced to tridiagonal form by zeroing out the $(3,1)\equiv(1,3)$ element; then the tridiagonal form is reduced to block diagonal form. In case 4, $\tilde{D}$ has a 2×2 block followed by a 1×1 block; once again the worst situation would be for $\tilde{D}$ to fill in and become a full 3×3 block; we zero out the $(3,1)\equiv(1,3)$ element and reduce the tridiagonal matrix to block diagonal form. In case 5, $\tilde{D}$ has two 2×2 blocks; the $(3,2)\equiv(2,3)$ and the $(4,2)\equiv(2,4)$ elements fill in; the resulting matrix is reduced to tridiagonal and then to block diagonal form.

As in the previous method, applying the transformations to the Z matrix is the major part of the work; this requires $3n(n-t)$ multiplications. Determining which transformations should be performed requires at most $3(n-t)$ comparisons.

Case 1:

$$\begin{array}{lccccccccc}
D: & x & & & x & x & \text{might reduce to} & & x & \\
 & & x & \rightarrow & x & x & & \rightarrow & & x \\
y^T: & + & x & & 0 & x & & & 0 & x
\end{array}$$

Case 2:

$$\begin{array}{lccccccccc}
D: & x & x & & x & x & \text{might reduce to} & & x & \\
 & x & x & \rightarrow & x & x & & \rightarrow & & x \\
y^T: & + & x & & 0 & x & & & 0 & x
\end{array}$$

Case 3:

$$\begin{array}{lccccccccccccccccc}
D: & x & & & & x & & & \text{worse case:} & x & x & + & & x & x & & & \text{to block} \\
 & & x & x & \rightarrow & & x & x & \rightarrow & x & x & x & \rightarrow & x & x & x & \rightarrow & \text{diagonal} \\
 & & x & x & & & x & x & \text{No pivoting} & + & x & x & & & x & x & & \text{form} \\
y^T: & x & + & x & & + & 0 & x & & 0 & 0 & x & & 0 & 0 & x & &
\end{array}$$

Case 4:

$$\begin{array}{lccccccccccccc}
D: & x & x & & \text{worse case:} & x & x & + & & x & x & & & \text{to block} \\
 & x & x & & \rightarrow & x & x & x & \rightarrow & x & x & x & \rightarrow & \text{diagonal} \\
 & & & x & \text{pivoting} & + & x & x & & & x & x & & \text{form} \\
 & & & & \text{necessary} & & & & & & & & & \\
y^T: & 0 & + & x & & 0 & 0 & x & & 0 & 0 & x & &
\end{array}$$

Case 5:

$$\begin{array}{lcccccccccccccc}
D: & x & x & & & & x & x & & & \text{if permutation of 2} & x & x & & \\
 & x & x & & & & x & x & & & \rightarrow & x & x & x & + \\
 & & & x & x & \rightarrow & & & x & x & \text{and 4 is necessary,} & & x & x & x \\
 & & & x & x & & & & x & x & \text{also permute 1 and 3} & & + & x & x \\
y^T: & 0 & x & + & x & & 0 & + & 0 & x & & 0 & 0 & 0 & x
\end{array}$$

$$\begin{array}{lccccccccccc}
\text{Worse} & x & x & + & & & x & x & & & & \\
\text{case:} & x & x & x & & & x & x & x & & & \text{to block} \\
\rightarrow & + & x & x & x & \rightarrow & & x & x & x & \rightarrow & \text{diagonal form} \\
\text{pivoting} & & & x & x & & & & x & x & & \\
\text{necessary} & & & & & & & & & & & \\
y^T: & 0 & 0 & 0 & x & & 0 & 0 & 0 & x & &
\end{array}$$

FIG. 1.

5. MATRIX UPDATE TECHNIQUES WHEN A CONSTRAINT IS DEACTIVATED

Let $\hat{A}^T$ denote the current $t\times n$ matrix of active constraints. Let Z be an $n\times(n-t)$ matrix of rank $n-t$ such that

$$\hat{A}^T Z=0 \quad \text{and} \quad Z^T QZ=D, \quad \text{a block diagonal matrix.}$$

Assume the matrix A'^T is constructed by deleting the sth row from $\hat{A}^T$. In this section we shall outline the construction of D' and Z', the new D and Z matrices respectively.

To determine Z', find a vector $\mathbf{z}$ such that $A'^T\mathbf{z}=0$ and $\mathbf{z}$ is linearly independent of Z. Since $A'^T Z=0$,

$$A'^T[\,Z \,\vdots\, \mathbf{z}\,]=0.$$

The matrix Z' is $[\,Z \,\vdots\, \mathbf{z}\,]M$, where M is chosen such that

$$D'=Z'^T QZ'$$

is block diagonal.

The vector $\mathbf{z}$ can be formed in a number of ways; perhaps one of the easiest is the method suggested in [7]. If $U\hat{A}=R$ and $U'A'=R'$ where U and U' are orthogonal matrices and R and R' are upper triangular, then there exists a matrix V such that $VR=R'$. Then VU can be partitioned as

$$VU=\begin{bmatrix} U' \\ \cdots \\ \mathbf{z}^T \end{bmatrix}.$$

The matrix D' may be formed by taking explicit advantage of the diagonality of D. However, since part of the problem is very similar to the one addressed in Sec. 4, we suggest using the module designed for that section with y of (4.1) defined by $Z^T Q$. This module will give us a matrix S such that $\tilde{D}=S^T(\,Z \,\vdots\, \mathbf{z})^T Q(\,Z \,\vdots\, \mathbf{z})S$ has the form

$$\begin{bmatrix} x & & & \\ & x & & \\ & & \ddots & \\ & & & xx \\ & & & xx \end{bmatrix}.$$

The Bunch-Kaufman algorithm is then applied to the lower 2×2 of $\tilde{D}$ to form D'.

6. SIMPLE CONSTRAINTS

Often the constraints are very simple:

$$\pm x_i \geqslant b_i,$$

i.e., just lower and upper bounds on the variables. The simplicity of these constraints can be used to decrease the amount of computation.

The algorithms in this section apply the ideas of Murray [13] for simple constraints to our general factorization. The extension of the general algorithm to cover simple constraints is not different theoretically; however, it meant a nontrivial change to the data structures used in the program implementing the general algorithm. The importance of simple constraints was impressed upon the authors by the first three scientists who desired to use their program: the majority of their constraints were simple.

Let us assume that of the t active constraints, r are general, and $t-r$ are of the simple form. Certainly the rows and columns of $\hat{A}^T$ can be permuted into the form

$$\hat{A}^T = \begin{array}{cc} \left(\begin{array}{c:c} A_1^T & A_2^T \\ \hdashline E & 0 \end{array}\right) & \begin{array}{l} r \\ t-r \end{array}, \\ \begin{array}{cc} t-r & n-(t-r) \end{array} & \end{array} \tag{6.1}$$

where

$$e_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ \pm 1 & \text{if } i = j. \end{cases}$$

As we will show, the matrix A_2 plays much the same role as $\hat{A}$ played in the original algorithm. The smaller the ratio of r to t, the more efficient the algorithm will be.

In step 4 of the algorithm in Sec. 2, one solves $\hat{A}\mathbf{u} = \mathbf{y}$. Partitioning

$$\mathbf{u} \text{ as } \begin{pmatrix} \mathbf{u}_1 \\ \hdashline \mathbf{u}_2 \end{pmatrix} \begin{array}{l} r \\ n-r \end{array}$$

and

$$\mathbf{y} \text{ as } \begin{pmatrix} \mathbf{y}_1 \\ \hdashline \mathbf{y}_2 \end{pmatrix} \begin{array}{l} t \\ n-(t-r) \end{array},$$

we find from (6.1) that

$$A_2\mathbf{u}_1=\mathbf{y}_2$$

and

$$E\mathbf{u}_2=\mathbf{y}_1-A_1\mathbf{u}_1.$$

Thus it is not necessary to store a full decomposition of $\hat{A}$; it is only necessary to store a decomposition of A_2 which would facilitate the formation of $\mathbf{u}_1$. When a constraint is added or deleted from the active set, a row or column is added or deleted from A_2. If one uses a decomposition for A_2 similar to that given in (2.5) for $\hat{A}$, then the algorithms in [7] may be used for updating the decomposition when A_2 is changed.

In a similar vein, only that portion of the Z matrix which spans the null space of A_2 need be computed. Because of the structure of $\hat{A}$, the Z matrix may be written as

$$Z=\begin{bmatrix} 0 \\ \cdots \\ Z_2 \end{bmatrix}_{n-(t-r)} \tag{6.2}$$

where Z_2 has full column rank and $A_2^TZ_2=0$. Moreover requiring Z^TQZ to be block diagonal is tantamount to requiring that $Z_2^TQ_{22}Z_2$ be block diagonal where

$$Q=\begin{array}{cc} \begin{pmatrix} Q_{11} & \vdots & Q_{12} \\ \cdots & \cdot & \cdots \\ Q_{21} & \vdots & Q_{22} \end{pmatrix} & n-(t-r) \\ \qquad\qquad n-(t-r) & \end{array}$$

and the rows and columns of Q have first been permuted as the columns of $\hat{A}^T$.

To update the Z matrix when a general constraint is either added or deleted, one would apply the appropriate algorithms of Secs. 4 and 5 to Z_2.

When a simple constraint is added, the A_2 matrix loses a row and the Z_2 matrix loses a row and a column. Let us assume A_2 loses its qth row and let A_2' represent A_2 without its qth row.

Knowing that

$$A_2^TZ_2=0 \quad \text{and} \quad \begin{bmatrix} 0 & \vdots & Z_2^T \end{bmatrix} Q \begin{bmatrix} 0 \\ \cdots \\ Z_2 \end{bmatrix}=D,$$

a block diagonal matrix, we would like to construct an $[n-1-(t-r)]\times(n-t-1)$ matrix Z_2' such that

$$A_2'^T Z_2' = 0$$

and

$$\begin{bmatrix} 0 & \vdots & Z_2'^T \end{bmatrix} Q \begin{bmatrix} 0 \\ \cdots \\ Z_2' \end{bmatrix}$$

is block diagonal. Certainly a nonsingular matrix S can be found such that $y^T S = \beta e_{n-t}^T$ for some constant β where y^T is the qth row of the Z matrix. If $W = Z_2 S$, then W has the form

$$\begin{bmatrix} W_1 & x \\ & x \\ 0 & x \\ & x \\ W_2 & x \end{bmatrix} \qquad \longleftarrow q\text{th row} .$$

Let

$$X = \begin{bmatrix} W_1 \\ \cdots \\ W_2 \end{bmatrix}.$$

Then

$$0 = A_2^T Z_2 S \begin{bmatrix} I \\ \cdots \\ 0 \end{bmatrix} = A_2^T \begin{bmatrix} W_1 \\ \cdots \\ 0 \\ \cdots \\ W_2 \end{bmatrix} = A_2'^T X.$$

If S is constructed so that

$$\begin{bmatrix} 0_{t+1} & \vdots & X^T \end{bmatrix} G \begin{bmatrix} 0_{t+1} \\ \cdots \\ X \end{bmatrix}$$

is diagonal, then X satisfies the properties of Z_2'. This is exactly the problem addressed in Sec. 4, and one can certainly use either of the algorithms given there to form S.

When a simple constraint is deleted, the A_2 matrix gains a row and the Z_2 matrix gains a row and a column. A ntatural way to proceed is to append a column z to the matrix $\begin{bmatrix} Z_2 \\ 0 \end{bmatrix}$ whose last element is nonzero. This column should lie in the null space of the new A_2 matrix, A_2'. One then proceeds as in Sec. 5 to insure the block diagonality property.

The vector z may be generated in various ways; perhaps the method suggested in [7] is the most tractable. Assume one has the decomposition of (2.5) for A_2, i.e., one has a nonsingular upper triangular matrix R and an $r \times [r+(n-t)]$ matrix U of rank r such that

$$UA_2 = R.$$

Then

$$\begin{bmatrix} U & 0 \\ \cdots & \cdots \\ 0 & 1 \end{bmatrix} \begin{bmatrix} A_2 \\ \cdots \\ \mathbf{a}^T \end{bmatrix} = \begin{bmatrix} R \\ \cdots \\ \mathbf{a}^T \end{bmatrix} = \begin{bmatrix} x & x & x \\ & x & x \\ & & x \\ x & x & x \end{bmatrix}$$

and there exists a matrix S such that

$$S \begin{bmatrix} R \\ \cdots \\ \mathbf{a}^T \end{bmatrix} = \begin{bmatrix} R' \\ \cdots \\ 0 \end{bmatrix}$$

where R' is nonsingular and upper triangular. The vector $\mathbf{z}^T$ is given by

$$\mathbf{e}^T_{r+(n-t)+1} S \begin{bmatrix} U & 0 \\ \cdots & \cdots \\ 0 & 1 \end{bmatrix}.$$

Note that A_2 must have full rank if it is stipulated that $\hat{A}$ has full rank. Moreover the last element of z must be nonzero if $\hat{A}'$ has full column rank.

Our update algorithms in Secs. 4, 5, and 6 have two phases: the first phase is designed to insure that Z spans the current space, and the second phase to insure that Z^TQZ has block diagonal structure. Gill and Murray [9] have to update Z and L from the LDL^T decomposition of Z^TQZ. It is easy to show that if for certain parts of our algorithm, we used orthogonal transformations rather than stabilized elementary transformations, then our first phase would be equivalent to their algorithm for updating Z and our second phase would be equivalent to their algorithm for updating L.

7. INITIALIZATION OF Z AND D

Before one begins the algorithm of Sec. 2, two major obstacles must be overcome:

(1) the identification of a feasible point,
(2) the construction of the initial Z and D matrices.

Several sources ([5], [10], and [11]) contain algorithms for identifying points which satisfy a system of linear inequalities.

The approach one chooses for finding Z and D matrices, once a feasible point has been located, depends on the structure of the problem and the number of active constraints. A few ideas are given below.

Let $\hat{A}^T$ denote a set of t linearly independent active constraints at the initial point of which r are of the general form, and consider the orthogonal decomposition of A_2 in (6.1) given by

$$A_2 = H\begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{7.1}$$

where R is the upper triangular matrix in (2.5) and H is an $s \times s$ orthogonal matrix where $s = n-(t-r)$. If H is partitioned as

$$H = \begin{array}{c} [\, U^T \ \vdots \ V \,], \\ \quad r \quad\ s-r \end{array}$$

then V spans the null space of A, i.e., the same space that must be spanned by the Z_2 matrix of (6.2). Bunch and Parlett [4] show that there exists a matrix L, a product of permutations and elementary lower triangular matrices, such that

$$LV^TQ_{22}VL^T \equiv D \tag{7.2}$$

where D is a block diagonal matrix with blocks of order 1 and 2. Thus the matrix VL^T has the properties stipulated for the Z matrix, and the D in (7.2) for the matrix D of Sec. 2.

Any good implementation of these ideas will probably consider the fact that H can be represented as a sequence of Householder transformations and L as a sequence of interspersed elementary transformations and permutations. The ratio of r to s will dictate more explicitly how these transformations should be used.

Method A. If r is small, i.e., $r<s/2$, perhaps the best way to proceed is to apply the Householder transformations, which compose H, as congruent transformations to Q_{22} of (6.3). Then the Bunch-Kaufman algorithm can be applied to the lower $(s-r)\times(s-r)$ submatrix of the transformed Q_{22} matrix to form D. To find Z_2 one applies the transpose of all the transformations, in the reverse order in which they were computed, to $(0 \,\vdots\, I_{s-r})^T$. When r is 0, the algorithm is most efficient and requires $s^3/3+O(s^2)$ multiplications and additions to find U, R, Z_2, and D. As r increases, applying the transformations to Q_{22} takes its toll, and when r is s, $\frac{7}{3}s^3+O(s^2)$ multiplications and additions are required.

Method B. For values of $r>s/2$, it makes more sense not to apply the Householder transformations to Q but to apply them in reverse order to $\left[0 \,\vdots\, I_{s-r}\right]^T$ to form V explicitly. When the Bunch-Kaufman algorithm is applied to $V^TQ_{22}V$, each transformation in L is determined by portions of one or two columns of the matrix formed by applying all previous transformations as congruent transformations to $V^TQ_{22}V$. Since $M(V^TQ_{22}V)M^T=(MV^T)Q_{22}(VM^T)$, one can postmultiply V by the transpose of the transformations as they are determined, and if the (i,j)th element of the transformed matrix is needed by the Bunch-Kaufman algorithm, then it may be determined as $\mathrm{v}_i^TQ_{22}\mathrm{v}_j$. One can certainly take advantage of the fact that the algorithm looks at a column at a time. Moreover, by the time the D matrix is formed, the matrix which had contained V now contains Z_2. The operation count is highest when $r=0$, when $2s^3+O(s^2)$ multiplications and additions are needed, and is lowest when $r=s$, when $\frac{4}{3}s^3+O(s^2)$ multiplications and additions are needed.

In our experience people using the algorithm have been able to identify feasible points without using a feasible point subroutine. Because these initial points are obtained by sight, rarely are any general constraints active, and $r\ll s$. Thus in our implementation we have chosen Method A given above. If a feasible point were generated mechanically, it would probably lie at the intersection of general constraints, and the wasted effort would be about s^3 operations. On the other hand, if Method B were applied to a problem in which $r\ll s$, the wasted effort would probably be close to $2s^3$ operations.

8. A NUMERICAL EXAMPLE

The algorithm was implemented in a FORTRAN program and applied to several examples. The example given below and constructed by the authors is complicated enough to exercise most of the code, but simple enough to permit easy detection of trouble. Its Hessian matrix is indefinite, and the D matrix obtained by the application of the Bunch-Kaufman algorithm to it has two 2×2 blocks. The problem has at least two local minima, at both of which simple and general constraints are active. It is easy to choose initial feasible points for which the algorithm dictates the adding and dropping of simple and general constraints.

EXAMPLE. Find a local minimum of

$$f(x)=\tfrac{1}{2}\mathbf{x}^TQ\mathbf{x}+\mathbf{c}^T\mathbf{x}$$

where

$$\mathbf{x}^T=(x_1,\dots,x_8),$$

$$\mathbf{c}^T=(7,6,\dots,0),$$

and

$$q_{ij}=\begin{cases}|i-j| & \text{if } i\neq j,\\ 1.69 & \text{if } i=j,\end{cases}$$

subject to the 16 simple constraints

$$-i-(i-1)\times 0.1\leqslant x_i\leqslant i,\qquad i=1,2,\dots,8,$$

and the 7 general constraints

$$-x_i+x_{i+1}\geqslant -1-(i-1)\times 0.05,\qquad i=1,2,\dots,7.$$

The problem has a local minimum of -621.488 at

$$\mathbf{x}^T=(-1,\ -2,\ -3.05,\ -4.15,\ -5.3,6,7,8),$$

one of -642.6 at

$$\mathbf{x}^T=(-1,\ -2.1,\ -3.15,\ -4.25,\ -5.4,6,7,8),$$

and one of -131.774 at

$$\mathbf{x}^T=(1,2,1.880144,0.780144,\ -0.369856,$$
$$-1.569856,\ -2.819856,\ -4.119856).$$

Beginning at $x_i = -i$, with no constraints termed active, both Methods 1 and 2 of Sec. 3 eventually reached the first local minimum. With Method 1 two constraints were activated that were later deactivated, and with Method 2 four constraints were activated that were later deactivated and an indefinite constrained Hessian was encountered after constraints were activated. To give the reader an idea of the course of a typical example, we give a trace of Method 1 on this example in Table 1. Initially $f(x) \approx 1516$.

TABLE 1

COURSE OF THE ALGORITHM ON AN EXAMPLE

	Structure of D before iteration	Iteration activity	Function value after activity
(1)	2 2×2 blocks 1 negative 1×1 block 3 positive 1×1 blocks	activated general constraint 1	1516
(2)	1 2×2 block 1 negative 1×1 block 4 positive 1×1 blocks	activated general constraint 3	1510
(3)	1 2×2 block 1 negative 1×1 block 3 positive 1×1 blocks	activated general constraint 2	1509
(4)	1 2×2 block 1 negative 1×1 block 2 positive 1×1 blocks	activated general constraint 4	1500
(5)	1 2×2 block 1 negative 1×1 block 1 positive 1×1 block	activated general constraint 5	1490
(6)	1 negative 1×1 block 2 positive 1×1 blocks	activated lower bound on x_6	1489
(7)	2 positive 1×1 blocks	activated lower bound on x_1	717
(8)	1 positive 1×1 block	activated upper bound on x_8, deactivated lower bound on x_6	67
(9)	1 positive 1×1 block	activated upper bound on x_7, deactivated general constraint 5	−482
(10)	1 positive 1×1 block	added upper bound on x_6	−621

For the same initial point with the lower bound of x_1 considered active, Method 2 led to the second minimum and Method 1 led to the third local minimum listed above. None of the methods gave any early indication which local minimum would be reached.

9. REMARKS

In this section several decisions which were made during the construction of the algorithm are discussed.

At first we tried to use Aasen's [1] method for the reduction of a symmetric matrix to tridiagonal form as the basis for a quadratic programming algorithm. We soon realized that the connectivity of the elements in a tridiagonal matrix prevented the development of stable, efficient updating algorithms when constraints were activated and deactivated; fill-in occurred which could not easily be zeroed out. We have shown that stable, efficient updating algorithms can be developed using the block diagonal decomposition.

Most of the cost of the algorithm lies in the determination of the active set of the solution. For some problems we have probably lengthened the process by insisting that constraints be deactivated only after the current equality problem has been solved and that only one constraint be deactivated before trying to proceed further. On the other hand this process prevents constant zigzagging between active sets, i.e., it insures the termination of the algorithm.

There are no fundamental reasons which prevent one from using a looser criterion for dropping constraints; only the termination proof is in jeopardy. In particular, our algorithm has a sufficiently general design that it does not break down if a constraint is dropped when the projected Hessian is indefinite.

We directed our attention mainly to the matrix updating algorithms, and we realize that there are some more sophisticated ideas that could be incorporated in the basic exchange algorithm. For example, if a step hits two constraints simultaneously, one may wish to add them both to the active set if they are linearly independent of those already in that set. One may also wish to choose that constraint for which the cosine of the angle between the constraint normal and the negative search direction is smallest [8]. Similarly, when dropping constraints one may wish to use a different criterion when it is realized that dropping any of a number of constraints will decrease the function further.

Like other linear and nonlinear programming algorithms which use the active set strategy, our algorithm may pose an intermediate problem with an ill-conditioned constrained Hessian. In this situation Z^TQZ ceases to be block diagonal, and significant roundoff error is observed in the computed solution. We have not found examples in which the computed solution is significantly infeasible; however, we have found examples in which the true local minimum is not found. These problems could be eliminated if one had a strategy for dropping constraints based on the condition of the equality constrained problem which will be generated next. At any rate, when the problem has been purportedly solved, a robust implementation of the algorithm should perform the matrix multiplications to find Z^TQZ and compare it with the computed D matrix. If they are significantly different, the procedure should be invoked again with the initial active set specified by the solution just obtained.

Our algorithm is based on a decomposition of the projected Hessian, which exists even when that matrix is indefinite. Hence we can accept any initial feasible point, regardless of the eigenstructure of the projected Hessian. While this point will often satisfy the additional Gill-Murray criterion that the projected Hessian must have at most one negative eigenvalue, the user of their algorithm may not be aware of this fact and will then first solve a linear programming program to find an extreme point of the feasible region.

REFERENCES

1 J. O. Aasen, On the reduction of a symmetric matrix to tridiagonal form, *Nordisk Tidskr. Informationsbehandling* (*BIT*) 11:233–242 (1971).

2 J. R. Bunch, Analysis of the diagonal pivoting method, *SIAM J. Numer. Anal.* 8:656–680 (1971).

3 J. R. Bunch, and L. Kaufman, Some stable methods for calculating inertia and solving symmetric linear systems, *Math. Comp.* 31:163–179 (1977).

4 J. R. Bunch, and B. N. Parlett, Direct methods for solving symmetric indefinite systems of linear equations, *SIAM J. Numer. Anal.* 8:639–655 (1971).

5 R. Fletcher, The calculation of feasible points for linearly constrained optimization problems, U. K. Atomic Energy Authority Report R6354, 1971.

6 D. M. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill, 1972.

7 P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, Methods for modifying matrix factorizations, *Math. Comp.* 28:505–535 (1974).

8 P. E. Gill, and W. Murray, *Numerical Methods for Constrained Optimization*, Academic, 1974.

9 P. E. Gill, and W. Murray, Numerically stable methods for quadratic programming, *Math. Programming* 14:349–372 (1978).

10 C. L. Lawson, and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, N.J., 1974.

11 D. L. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, 1973.
12 L. Mirsky, *An Introduction to Linear Algebra*, Clarendon Press, Oxford, 1975.
13 W. Murray, An algorithm for finding a local minimum of an indefinite quadratic program, Nat. Phys. Lab. Rept. NAC1, 1971.
14 G. W. Stewart, *Introduction to Matrix Computations*, Academic, 1973.
15 R. Fletcher, and T. L. Freeman, A modified Newton method for minimization, *J. Optimization Theory Appl.* 3:357–372 (1977).
16 D. C. Sorensen, Updating the symmetic indefinite factorization with applications in a modified Newton's method, Argonne Nat. Lab. Report ANL-77-49, June 1977.

Received 11 March 1979; revised 20 September 1979