AD-A115 854 STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB F/G 12/2
PROCEDURES FOR OPTIMIZATION PROBLEMS WITH A MIXTURE OF BOUNDS A--ETC(U)
MAY 82 P E GILL, W MURRAY, M A SAUNDERS N00014-75-C-0267
UNCLASSIFIED SOL-TR-82-6 ARO-18424.6-MA NL

END
DATE FILMED
7 82
DTIC

ARO 18424.6-MA

(12)

Systems
Optimization
Laboratory

AD A115854

DTIC FILE COPY

Department of Operations Research
Stanford University
Stanford, CA 94305

82

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints

(12)

by

Philip E. Gill, Walter Murray,
Michael A. Saunders and Margaret H. Wright

TECHNICAL REPORT SOL 82-6

May 1982

Research and reproduction of this report were partially supported by the Department of Energy Contract AM03-76SF00326, PA# DE-AT03-76ER72018; Office of Naval Research Contract N00014-75-C-0267; National Science Foundation Grants MCS-8119774, ECS-8012974; and Army Research Office Contract DAA29-79-C-0110.

Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints

by

Philip E. Gill, Walter Murray,
Michael A. Saunders and Margaret H. Wright

Systems Optimization Laboratory
Department of Operations Research
Stanford University
Stanford, California 94305

May 1982

ABSTRACT

When describing active-set methods for linearly constrained optimization, it is often convenient to treat all constraints in a uniform manner. However, in many problems the linear constraints include simple bounds on the variables as well as general constraints. Special treatment of bound constraints in the implementation of an active-set method yields significant advantages in computational effort and storage requirements. In this paper, we describe how to perform the constraint-related steps of an active-set method when the constraint matrix is dense and bounds are treated separately. These steps involve updates to the TQ factorization of the working set of constraints and the Cholesky factorization of the projected Hessian (or Hessian approximation).

This research was supported by the U.S. Department of Energy Contract DE-AC03-76SF00326, PA No. DE-AT03-76ER72018; National Science Foundation Grants MCS-7926009 and ECS-8012974; the Office of Naval Research Contract N00014-75-C-0267; and the U.S. Army Research Office Contract DAAG29-79-C-0110.

1. Introduction

Constrained optimization problems often include a set of *linear inequality constraints*, which may be written in several different forms. We consider the following three:

$$\begin{aligned} &LC1 && Ax \geq b; \\ &LC2 && Ax = b, \qquad \ell \leq x \leq u; \\ &LC3 && \ell \leq \begin{pmatrix} I \\ A \end{pmatrix} x \leq u. \end{aligned}$$

For convenience we shall always assume that A is a matrix with m rows and n columns. The dimensions of other quantities follow in each case. The constraints involving A are called *general constraints*, and inequalities of the form $\ell \leq x \leq u$ are called *simple bounds* or just *bounds*. If necessary, some of the components of ℓ or u may be taken as $-\infty$ or ∞. (Note that general *equality* constraints may be represented in *LC1* by extending the relations to include equality, and in *LC3* by specifying the same value for the corresponding elements of ℓ and u.)

The forms *LC1* – *LC3* are *equivalent*, in the sense that any set of linear inequality constraints may be expressed in each of the forms, given suitable definition of A, b, ℓ, u and x. The primary feature of interest in *LC2* is that general inequality constraints are converted to general *equality* constraints by adding slack variables.

The most popular methods for treating linear inequality constraints are called *active-set methods* (see Section 2). An essential characteristic of these methods is that they maintain a prediction of the set of constraints that are active at the solution (this prediction will be called the *working set*). The working set is *updated* by adding and deleting constraints as the iterations proceed. In presenting the formal description of an active-set method, it is often convenient to treat all inequality constraints uniformly, since the strategies that determine changes in the working set are usually unaffected by whether or not a constraint is a general constraint or a simple bound.

In any *implementation* of an active-set method, changes in the working set involve updates to a certain matrix C associated with the working set (and often to other matrices as well). The data structures chosen for an implementation will inevitably be more efficient for one constraint form than another. If the implementation is based on form *LC1*, changes in the working set lead to changes in the *rows* of C, while with form *LC2*, changes in the working set lead to changes in the *columns* of C. With form *LC3*, the changes involve both the rows and the columns of C.

Corresponding changes must also be made to some *factorization* of C. For reasons of simplicity, past implementors have dealt almost exclusively with forms *LC1* and *LC2*. Some examples in the literature follow.

LC1, dense A: Rosen (1960) and many others for nonlinear programs (see Gill and Murray, 1974, for further references); Stoer (1971) for constrained linear least squares.

LC1, sparse A: Buckley (1975) for nonlinear programs.

LC2, dense A: Mifflin (1979) for constrained linear least squares; Bartels (1980) for linear programs.

LC2, sparse A: Commercial mathematical programming systems for linear and integer programs; Murtagh and Saunders (1978, 1982) for nonlinear programs.

The disadvantage of implementations based on *LC1* or *LC2* arises when the "natural" statement of the constraints corresponds most closely to *LC3* (i.e., when there is a significant number of bounds in *LC1* or general inequalities in *LC2*). In such cases, a large proportion of the rows or

columns of C will be those of the identity matrix:

$$C = \begin{pmatrix} 0 & I \\ C_1 & C_2 \end{pmatrix} \quad \text{or} \quad C = \begin{pmatrix} C_1 & 0 \\ C_2 & I \end{pmatrix}$$

respectively. Maintaining a factorization of the whole of C therefore involves more than the ideal amount of storage and work. (Certain economies do arise automatically if C is treated as a sparse matrix, but much of the objection remains.)

Implementations based on *LC3* effectively take advantage of the above structure in C. Few authors have previously considered the associated complications. Zoutendijk (1970) and Powell (1975) have considered how changes in the working set may be performed when C is square (with varying dimension) and its inverse is updated in product form. Gill and Murray (1973) discussed the nature of the updates required in a non-simplex linear programming method based on an orthogonal factorization of C.

In this paper we discuss the implementation of an active-set method suited to constraints of the form *LC3*, with A treated as a dense matrix. We describe how to update the TQ factors of the matrix C and the Cholesky factors of the accompanying projected Hessian (or approximate Hessian). The procedures have been implemented in computer software for linear and quadratic programs and for linearly constrained optimization, as described in Gill *et al.* (1982a,b). The principal advantages in dealing with *LC3* are as follows.

1. The matrix to be factorized has dimension $m_L \times n_{FR}$, where $m_L \leq \min\{m, n\}$ and $n_{FR} \leq n$. Further, m_L and n_{FR} are often much smaller than these bounds.
2. When finite differences are used to approximate derivatives, special treatment of bounds may lead to significant economies in function evaluations.
3. Certain methods for semi-definite and indefinite quadratic programming may construct a temporary set of simple bounds in order to begin optimization. For such methods, the ability to handle bounds efficiently is crucial even if the original problem does not contain bounds.

The first advantage is best illustrated by the case of linear programming. Standard implementations of the primal simplex method (Dantzig, 1963) apply to constraints in the form *LC2*. These are most efficient when $m \ll n$, since the matrix to be factorized is always $m \times m$. If most of the n variables in *LC2* are slack variables, the standard device for avoiding gross inefficiency is to solve the dual problem. In contrast, if the form *LC3* is assumed when implementing the simplex method (the most famous of all active-set methods!), then *maximum efficiency is obtained regardless of the ratio of m to n*. This advantage is all the more important for nonlinear problems, where the device of solving the dual is not necessarily applicable or efficient.

The techniques given here may be applied to active-set methods for general optimization problems, whenever linear and nonlinear constraints are treated separately — particularly in methods that solve a sequence of quadratic programming subproblems (e.g., Murray, 1969; Biggs, 1972; Han, 1976; Powell, 1977; Murray and Wright, 1982) or linearly constrained subproblems (e.g., Rosen and Kreuser, 1972; Robinson, 1972; Murtagh and Saunders, 1982).

2. Overview of an active-set method

Apart from the requirement of feasibility, the optimality conditions for a constrained problem involve only the constraints that are *active* (hold with equality) at the solution. Active-set methods are based on an attempt to identify the constraints that are active at the solution, and to treat these as *equality* constraints in the subproblems that define the iterates. The temporary equalities

are used to reduce the dimensionality of the minimization. In a typical active-set method, the direction of search is computed by solving a (usually simplified) subproblem in which a subset of the problem constraints are treated as *equalities*. The subset of the problem constraints used to compute the search direction will be called the *working set*.

Before giving a detailed description of the special treatment of bounds, we consider some of the main steps in an active-set method. Our concern is with the k-th iteration, and the associated iterate x_k. We denote by C_k the matrix whose rows are the constraints in the current working set, by t_k the number of constraints in the working set (the number of rows of C_k), and by g_k the gradient of the function to be minimized, evaluated at x_k. The matrix Z_k is defined as a matrix whose columns span the null space of C_k (i.e., $C_k Z_k = 0$); *this paper is primarily concerned with active-set methods in which Z_k is stored explicitly.*

The major operations associated with the current working set are:

(i) formation of the projected gradient $Z_k^T g_k$;
(ii) solution of the linear system

$$Z_k^T H_k Z_k p_z = -Z_k^T g_k \tag{1}$$

for the $(n - t_k)$-dimensional vector p_z;
(iii) calculation of the search direction $p_k = Z_k p_z$;
(iv) calculation of a Lagrange multiplier estimate λ_k by solving

$$\min_{\lambda} \|C_k^T \lambda_k - v_k\|_2^2 \tag{2}$$

for some vector v_k.

(These quantities may be computed in other mathematically equivalent forms; see Gill, Murray and Wright (1981) for a discussion of alternatives.)

The matrix H_k in (1) usually represents second-derivative information about the objective function, but is not necessarily stored explicitly. For example, H_k may be the exact Hessian of the objective function in a quadratic program, or a factorized representation of the Hessian in a linear least-squares problem. In some methods, H_k will be a quasi-Newton approximation of the Hessian matrix, or $Z_k^T H_k Z_k$ itself will be approximated. For simplicity, we shall always refer to H_k as the "Hessian", and to $Z_k^T H_k Z_k$ as the "projected Hessian".

3. Representation of the working set and associated factorizations

Our concern in this section is with the factorizations used in an active-set algorithm, and the effect of the separate treatment of bounds. We shall assume that rank $(C_k) = t_k$, i.e. that the rows of C_k are linearly independent. (In practice, this condition can be enforced by suitable choice of the working set; see Gill *et al.*, 1982a.) For simplicity of notation, we temporarily drop the subscript k associated with the current iteration.

At a typical iteration, the working set of t constraints will include a mixture of general constraints and bounds. If the working set contains any simple bounds, those variables will be *fixed* on the corresponding bounds during the given iteration; all other variables will be regarded as *free*. We use the suffices "FX" and "FR" to denote items associated with the two types of variable. Suppose that C contains n_{FX} bounds and m_L general constraints (so that $t = n_{FX} + m_L$). Let A denote the matrix whose rows are the m_L general constraints in the working set, and let n_{FR} denote the number of free variables ($n_{FR} = n - n_{FX}$). If bounds are *not* treated separately, $n_{FX} = 0$, $n_{FR} = n$, and $m_L = t$.

In the implementation of an active-set method, the indices of the free variables and of the general constraints in the working set may be stored in lists (and relevant vectors ordered accordingly). Hence, we shall assume without loss of generality that the *last* n_{FX} variables are fixed. The matrix of constraints in the working set can then be written as

$$C = \begin{pmatrix} 0 & I_{FX} \\ A & \end{pmatrix} = \begin{pmatrix} 0 & I_{FX} \\ A_{FR} & A_{FX} \end{pmatrix}, \tag{3}$$

where A_{FR} is an $m_L \times n_{FR}$ matrix, and I_{FX} denotes an n_{FX}-dimensional identity matrix.

The first matrix that must be available in order to perform the calculations (i) through (iv) is the matrix Z, whose columns form a basis for the set of vectors orthogonal to the rows of C. The special form of (3) means that Z also has a special form, which involves *only the columns of A corresponding to free variables.* Let n_z denote $n - t$, the number of columns of Z. An $n \times n_z$ matrix Z whose columns are orthogonal to the rows of C in (3) is given by

$$Z = \begin{pmatrix} Z_{FR} \\ 0 \end{pmatrix}, \tag{4}$$

where Z_{FR} is an $n_{FR} \times n_z$ matrix whose columns form a basis for the subspace of A_{FR} (i.e., $A_{FR}Z_{FR} = 0$). (If m_L is zero, Z_{FR} is the n_{FR}-dimensional identity matrix.)

We shall obtain the matrix Z_{FR} in (4) from a variant of the usual orthogonal factorization which we shall call the *TQ factorization*. (The reasons for using the TQ factorization will be discussed in Section 5.3, when we consider procedures for updating the matrix factorizations following a constraint deletion.) The TQ factorization of A_{FR} is defined by

$$A_{FR}Q = (\,0 \quad T\,), \tag{5}$$

where Q is an $n_{FR} \times n_{FR}$ orthonormal matrix, and T is an $m_L \times m_L$ "reverse" triangular matrix such that $T_{ij} = 0$ for $i + j \leq m_L$. We illustrate the form of T with a 4×4 example:

$$T = \begin{pmatrix} & & & \times \\ & & \times & \times \\ & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix}.$$

(Clearly, T is simply a lower-triangular matrix with its columns in reverse order.) From (5) it follows that the *first* n_z columns of Q can be taken as the columns of the matrix Z_{FR}. We denote the remaining columns of Q by Y_{FR} (the columns of Y_{FR} form an orthogonal basis for the subspace of vectors spanned by the rows of A_{FR}).

The TQ factorization for the full matrix C (3) has the form

$$C \begin{pmatrix} Q & 0 \\ 0 & I_{FX} \end{pmatrix} = \begin{pmatrix} 0 & I_{FX} \\ A_{FR} & A_{FX} \end{pmatrix} \begin{pmatrix} Z_{FR} & Y_{FR} & 0 \\ 0 & 0 & I_{FX} \end{pmatrix} = \begin{pmatrix} 0 & 0 & I_{FX} \\ 0 & T & A_{FX} \end{pmatrix}. \tag{6}$$

We emphasize that the usefulness of the TQ factorization does not depend on separate treatment of bounds, since the TQ factorization of the full matrix C may also be computed and updated using the procedures to be described in an implementation based on the form *LC1*.

4. Calculation of the search direction and Lagrange multipliers

The calculations in Section 2 simplify when the working set and its factorization have the special forms (3), (4) and (6). From (4) it follows that the search direction has the form $p = (p_{FR}^T \quad 0)^T$. Further, $Z^T g = Z_{FR}^T g_{FR}$ and $Z^T H Z = Z_{FR}^T H_{FR} Z_{FR}$. Consequently, the computation of p_{FR} involves three steps: forming the vector $Z_{FR}^T g_{FR}$; solving the linear system

$$Z_{FR}^T H_{FR} Z_{FR} \hat{p}_z = -Z_{FR}^T g_{FR} \tag{7}$$

for the vector $\hat{p}_z$; and forming the vector $p_{FR} = Z_{FR} \hat{p}_z$. (In certain contexts, such as quadratic programming and linear least-squares, the known form of the objective function allows substantial savings in solving (7).) The work involved is reduced as n_{FX} increases; therefore, if the working set contains any bound constraints, less work is required if bound constraints are treated separately.

In the active-set algorithms of interest, the matrix in (7) is assumed to be positive definite, and thus (7) is solved using the *Cholesky factorization* of $Z_{FR}^T H_{FR} Z_{FR}$ (see, e.g., Wilkinson, 1965; Stewart, 1973):

$$Z_{FR}^T H_{FR} Z_{FR} = R^T R, \tag{8}$$

where R is upper triangular.

Simplifications also arise in solving equation (2) for the Lagrange multiplier estimates. Let λ be partitioned into an m_L-vector λ_L (the multiplier estimates corresponding to the general linear constraints) and an n_{FX}-vector λ_{FX} (the multiplier estimates corresponding to the active bound constraints). From (6), the equations defining the multipliers are

$$C^T \lambda = \begin{pmatrix} 0 & A_{FR}^T \\ I_{FX} & A_{FX}^T \end{pmatrix} \begin{pmatrix} \lambda_{FX} \\ \lambda_L \end{pmatrix} = \begin{pmatrix} A_{FR}^T \lambda_L \\ \lambda_{FX} + A_{FX}^T \lambda_L \end{pmatrix} \approx \begin{pmatrix} v_{FR} \\ v_{FX} \end{pmatrix}, \tag{9}$$

where $\approx$ means "equal in the least-squares sense".

The vector λ_L is the least-squares solution of the first n_{FR} equations of (9) (which are compatible if $Z_{FR}^T v_{FR} = 0$):

$$A_{FR}^T \lambda_L \approx v_{FR}.$$

It follows from (6) that λ_L may be obtained by forming $Y_{FR}^T v_{FR}$ and solving the $m_L \times m_L$ non-singular reverse-triangular system

$$T^T \lambda_L = Y_{FR}^T v_{FR}.$$

The multiplier estimates associated with the bound constraints may then be computed directly by substituting in the remaining equations of (9), i.e.

$$\lambda_{FX} = v_{FX} - A_{FX}^T \lambda_L.$$

Note that if m_L is zero, λ_{FX} is given simply by v_{FX}.

The number of multiplications required to solve (9) in the manner given above is nm_L to form $Y_{FR}^T v_{FR}$ and $A_{FX}^T \lambda_L$, and $\frac{1}{2} m_L^2$ to solve the reverse-triangular system. The saving in work compared to treating all constraints uniformly is $\frac{1}{2} n_{FX}^2 + n_{FX}(n + m_L)$ multiplications.

5. Implementation and storage

When implementing an active-set method based on the TQ and Cholesky factorizations, the matrices to be stored include, from (6) and (8), the $n_{FR} \times n_{FR}$ matrix Q, the m_L-dimensional reverse triangular matrix T, and the n_z-dimensional upper triangular matrix R. The matrix Q is conceptually partitioned into two submatrices — Z_{FR}, the first n_z columns, and Y_{FR}, the last

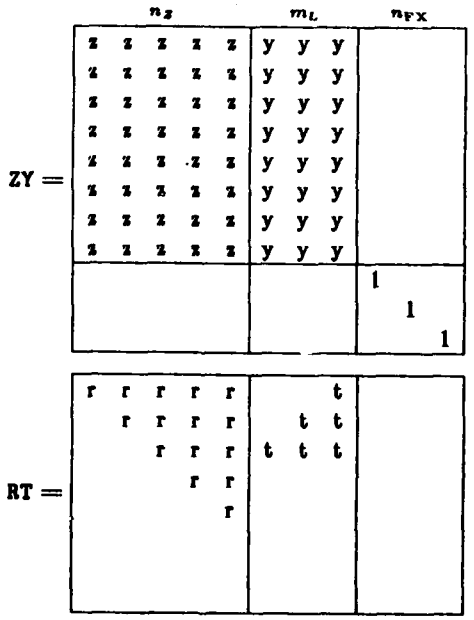
m_L columns, i.e.

$$Q = (Z_{FR} \quad Y_{FR}).$$

Changes in the working set will cause changes in these four matrices. From (6) we see that any transformations applied to the columns of Y_{FR} will also be applied to the columns of T; from (8) it follows that any transformations applied to the columns of Z_{FR} will also be applied to the columns of R. (The matrix R may also be changed in other ways — for example, by a low-rank modification in a quasi-Newton method. However, we consider only the effect of changes in the working set.)

In the implementation, the $n_{FR} \times n_{FR}$ matrix Q is stored explicitly, in the upper left corner of sufficiently large array ZY (in a general problem, n_{FR} may be as large as n). The dimensions of R and T are complementary (in the sense that $n_Z + m_L = n_{FR}$), and hence both matrices are stored in the upper left corner of a single array RT. The matrix R is stored in the *first* n_Z columns (corresponding to the columns of Z_{FR}), and the matrix T in the following m_L columns (corresponding to the columns of Y_{FR}). With this storage arrangement, rotations applied to columns of ZY can be applied to exactly the same columns of RT. We have chosen to store the triangular matrices R and T as two-dimensional arrays (rather than in a "packed" form), so that separate subroutines are not required to apply linear algebraic operations to triangular matrices. (In our implementation we use a set of linear algebra subroutines similar to the BLAS (Lawson et al., 1979) to perform these operations.)

The following diagram illustrates the parallel storage arrangements in ZY and RT for the case $n_Z = 5$ and $m_L = 3$. The elements of Z_{FR}, Y_{FR}, R and T are denoted by z, y, r, and t, respectively.



6. Changes in the working set

Unless the correct active set is known *a priori*, the working set must be modified during the execution of an active-set method, by adding and deleting constraints. Because of the simple

nature of these changes, it is possible to *update* the necessary matrix factorizations to correspond with the new working set. In the remainder of this section, we consider how to update the TQ factorization (6) and the Cholesky factorization (8) following a single change in the working set. If several constraints are to be added or deleted, the procedures are repeated as necessary.

The discussion of updates will assume a general familiarity with the properties of *plane rotations*. Sequences of plane rotations are used to introduce zeros into appropriate positions of a vector or matrix, and have exceptional properties of numerical stability (see, e.g., Wilkinson, 1965, pp. 47–48).

We shall illustrate each modification process using sequences of simple diagrams, following the conventions of Cox (1981) to show the effects of the plane rotations. Each diagram depicts the changes resulting from one plane rotation. The following symbols are used:

x	denotes a non-zero element that is not altered;
m	denotes a non-zero element that is *modified*;
f	denotes a previously zero element that is *filled in*;
0	denotes a previously non-zero element that is annihilated; and
·	(or blank) denotes a zero element that is unaltered.

In the algebraic representation of the updates, barred quantities will represent the "new" values.

6.1. Adding a general constraint. When a general constraint is added to the working set, its index can simply be placed at the end of the list of indices of general constraints in the working set. Therefore, without loss of generality we shall assume that the new constraint is added as the *last* row of A. The row dimension of A_{FR} and the dimension of T will thus *increase* by one, and the column dimension of Z_{FR} will *decrease* by one. (Note that the column dimension of A_{FR} is unchanged.) Let a^T denote the new row of A, partitioned into $(a_{FR}^T \quad a_{FX}^T)$. Let w^T denote the vector $a_{FR}^T Q$, and partition w^T as $(w_Z^T \quad w_Y^T)$, so that w_Z^T consists of the first n_Z components of w^T. From (5), it follows that

$$\bar{A}_{FR} Q = \begin{pmatrix} A_{FR} \\ a_{FR}^T \end{pmatrix} Q = \begin{pmatrix} 0 & T \\ a_{FR}^T Q \end{pmatrix} = \begin{pmatrix} 0 & T \\ w_Z^T & w_Y^T \end{pmatrix}.$$

We see that a new matrix $\bar{Q}$ can be obtained by applying a sequence of plane rotations on the *right* of Q to transform the vector w_Z^T to suitable form; the transformed matrix Q then becomes $\bar{Q}$. The sequence of rotations take linear combinations of the elements of w_Z^T to reduce it to a multiple (say, γ) of e_Z^T, where e_Z denotes the n_Z-th coordinate vector. The rotations are constructed to alter pairs of components in the order $(1,2), (2,3), \ldots, (n_Z - 1, n_Z)$, as indicated in the following diagrams, which depict the vector w_Z^T as it is reduced to γe_Z^T:

$$(\times\ \times\ \times\ \times\ \times) \rightarrow (0\ \mathrm{m}\ \times\ \times\ \times) \rightarrow (\cdot\ 0\ \mathrm{m}\ \times\ \times) \rightarrow (\cdot\ \cdot\ 0\ \mathrm{m}\ \times) \rightarrow (\cdot\ \cdot\ \cdot\ 0\ \mathrm{m}).$$

The effect of these transformations can be expressed algebraically as

$$\bar{A}_{FR} Q \begin{pmatrix} P & 0 \\ 0 & I \end{pmatrix} = \bar{A}_{FR} \bar{Q} = \begin{pmatrix} 0 & 0 & T \\ 0 & \gamma & w_Y^T \end{pmatrix} = \begin{pmatrix} 0 & \bar{T} \end{pmatrix}.$$

By construction, the rotations in P affect only the first n_Z columns of Q, so that the last m_L columns of $\bar{Q}$ are identical to those of Q, and the first n_Z columns of $\bar{Q}$ are linear combinations

of the first n_z columns of Q. Hence,

$$Z_{FR}P = (\bar{Z}_{FR} \quad y),$$

where y, the transformed last column of Z_{FR}, becomes the first column of $\bar{Y}_{FR}$.

The plane rotations applied to Z_{FR} also transform the Cholesky factor R of the projected Hessian. The chosen order of the rotations in P means that each successive rotation has the effect of introducing a subdiagonal element into the upper-triangular matrix R, as shown in the following sequence of diagrams. For clarity, we again show the vector w_z^T at the top as it is reduced to $(0 \ \gamma)^T$; the matrices underneath represent the transformed version of R.

```
x x x x x      0 m x x x      . 0 m x x      . . 0 m x      . . . 0 m
x x x x x      m m x x x      x m m x x      x x m m x      x x x m m
  x x x x  →     f m x x  →     x m m x  →     x x m m  →     x x x m m
    x x x        x x x          f m x          x m m          x x m m
      x x          x x            x x            f m            x m m
        x            x              x              x              f m
```

Since the last column of the matrix $Z_{FR}P$ is not part of $\bar{Z}_{FR}$, the last column of RP can be discarded. The remaining matrix is then restored to upper-triangular form by a forward sweep of *row* rotations (say, the matrix $\bar{P}$), which is applied on the *left* to eliminate the subdiagonal elements, as shown in the following diagrams.

```
x x x x      m m m m      x x x x      x x x x      x x x x
x x x x      0 m m m      . m m m      . x x x      . x x x
  x x x  →     x x x  →     0 m m  →     . m m  →     . x x
    x x          x x          x x          0 m          . m
      x            x            x            x            0
```

Let $\tilde{R}$ denote the matrix RP with its last column deleted; then we have

$$\bar{P}\tilde{R} = \begin{pmatrix} \bar{R} \\ 0 \end{pmatrix},$$

where $\bar{R}$ is upper-triangular. Note that the rotations in $\bar{P}$ affect $\bar{R}$, but not $\bar{Q}$ or $\bar{T}$.

The number of multiplications associated with adding a general constraint includes the following (where only the highest-order term is given): n_{FR}^2 to form $a_{FR}^T Q$; $3n_z^2$ for the two sweeps of rotations applied to R; and $3n_{FR}n_z$ to transform the appropriate columns of Q. (We assume the three-multiplication form of a plane rotation; see Gill *et al.*, 1974.)

6.2. Adding a bound. When a *bound* constraint is added to the working set, a previously free variable becomes fixed on its bound. Thus, the column dimension of A_{FR}, the column *and* row dimensions of Z_{FR} and the dimension of Q are decreased by one. The dimension of T is unaltered.

We assume that the new fixed variable corresponds to the *last* (n_{FR}-th) column of A_{FR}; in practice, the index of the variable at the end of the list of free variables is moved to the position of the newly fixed variable. Let e_{FR}^T denote the n_{FR}-th coordinate vector. Addition of the n_{FR}-th

variable to the working set causes the vector e_{FR}^T to be added as the first row of $\bar{C}$, i.e.

$$\bar{C} = \begin{pmatrix} e_{FR}^T \\ C \end{pmatrix}, \tag{10}$$

and, in effect, moves the last column of A_{FR} into A_{FX}. Let w^T denote the n_{FR}-th row of Q, and note that w has unit Euclidean length. As in Section 6.1, let w^T be partitioned as $(w_Z^T \quad w_Y^T)$. After adding the bound to the working set, it follows from (3) and (10) that

$$\bar{C}\begin{pmatrix} Q & 0 \\ 0 & I_{FX} \end{pmatrix} = \begin{pmatrix} e_{FR}^T \\ C \end{pmatrix}\begin{pmatrix} Q & 0 \\ 0 & I_{FX} \end{pmatrix} = \begin{pmatrix} w_Z^T & w_Y^T & 0 \\ 0 & 0 & I_{FX} \\ 0 & T & A_{FX} \end{pmatrix}. \tag{11}$$

In order to compute the updated TQ factorization, the first row of the matrix on the right-hand side of (11) must be reduced to the n_{FR}-th coordinate vector. This is achieved by a forward sweep of plane rotations (say, P) that alter columns 1 through n_{FR}, in the order $(1,2)$, ..., $(n_{FR}-1, n_{FR})$, such that the n_{FR}-th row and column of Q are transformed to the n_{FR}-th coordinate vector. The effect of the rotations in P on the matrix Q can be represented as

$$QP = (\, Z_{FR} \quad Y_{FR} \,)P = \begin{pmatrix} \bar{Q} & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \bar{Z}_{FR} & \bar{Y}_{FR} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The effect of the rotations in P on R and T is best understood by considering them in two groups. Firstly, the rotations that alter columns 1 through n_z of Q affect the columns of R exactly as described in Section 6.1, and a set of row rotations are then applied to restore the upper-triangular form of $\bar{R}$. Secondly, the rotations that alter columns n_z through n_{FR} of Q cause elements to be added above the reverse diagonal of T, as shown in the following diagrams. The vector at the top shows the order of the rotations, with T below.

```
x x x x x      0 m x x x      . 0 m x x      . . 0 m x      . . . 0 m
    x              x              x              x              f m
  x x      →     x x      →     x x      →     f m x      →     x m m
x x x            x x x          f m x          x m m x          x x m m
x x x x        f m x x x        x m m x x      x x m m x        x x x m m
```

The first m_L columns of the transformed T become $\bar{T}$, and the remaining column becomes the column of $\bar{A}$ corresponding to the new fixed variable.

Let I_{FX}^+ denote the $(n_{FX}+1)$-dimensional identity matrix. The final configuration is thus

$$\bar{C}\begin{pmatrix} Q & 0 \\ 0 & I_{FX} \end{pmatrix}\begin{pmatrix} P & 0 \\ 0 & I_{FX} \end{pmatrix} = \bar{C}\begin{pmatrix} \bar{Q} & 0 \\ 0 & I_{FX}^+ \end{pmatrix} = \begin{pmatrix} 0 & 0 & I_{FX}^+ \\ 0 & \bar{T} & \bar{A}_{FX} \end{pmatrix},$$

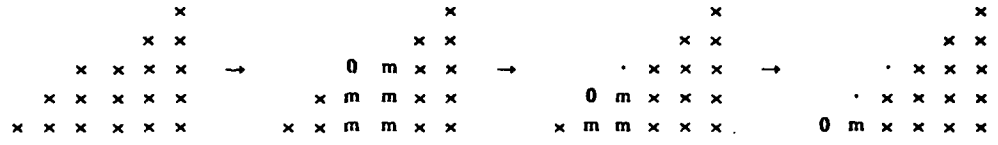
as desired.

The number of multiplications associated with adding a bound constraint includes all those needed to add a general constraint, with an additional $\frac{1}{2}m_L^2$ to modify T.

6.3. Deleting a general constraint. When a general constraint (say, the i-th) is deleted from the working set, the row dimension of A_{FR} and the dimension of T are decreased by one, and the column dimension of Z_{FR} is *increased* by one. (As in Section 6.1, the column dimension of A_{FR} is not altered.) From (5), we have

$$\bar{A}_{FR}Q = (\,0 \quad S\,),$$

where S is an $(m_L - 1) \times m_L$ matrix such that rows 1 through $i-1$ are in reverse triangular form, and the remaining rows have one extra element above the reverse diagonal. In order to reduce the last $m_L - 1$ columns of S to the desired reverse triangular form of $\bar{T}$, a backward sweep of plane rotations (say, P) is applied on the right, to take linear combinations of columns $(m_L - i + 1, m_L - i), \ldots, (2, 1)$. The effect of these rotations is shown in the following diagrams for the case $m_L = 6$ and $i = 3$:



This transformation may be expressed as

$$\bar{A}_{FR}Q\begin{pmatrix} I & 0 \\ 0 & P \end{pmatrix} = \bar{A}_{FR}\bar{Q} = (\,0 \quad \bar{T}\,).$$

The first n_z columns of Q are not affected by the rotations in P, and hence the first n_z columns of $\bar{Z}_{FR}$ are identical to the columns of Z_{FR}. The matrix $\bar{Z}_{FR}$ is given by

$$\bar{Z}_{FR} = (\,Z_{FR} \quad z\,), \tag{12}$$

where the new (last) column z of $\bar{Z}_{FR}$ is a linear combination of the relevant columns of Y_{FR}. (When $i = m_L$, no reduction at all is needed, and z is just the first column of Y_{FR}.)

Because of the form (12), the new projected Hessian matrix $\bar{Z}_{FR}^T H_{FR} \bar{Z}_{FR}$ is given by

$$\bar{Z}_{FR}^T H_{FR} \bar{Z}_{FR} = \bar{R}^T \bar{R} = \begin{pmatrix} R^T R & v \\ v^T & \theta \end{pmatrix}, \tag{13}$$

where $v = Z_{FR}^T H_{FR} z$ and $\theta = z^T H_{FR} z$. (Note that $\bar{H}_{FR} = H_{FR}$.) In cases when H_{FR} or $R^T R$ is a quasi-Newton approximation, the new row and column of (13) may need to be approximated.

Let r denote the solution of $R^T r = v$; if $\bar{Z}_{FR}^T H_{FR} \bar{Z}_{FR}$ is positive definite, the quantity $\theta - r^T r$ must be positive. In this case, only one further step of the row-wise Cholesky factorization is needed to compute the new Cholesky factor $\bar{R}$, which is given by

$$\bar{R} = \begin{pmatrix} R & r \\ 0 & \rho \end{pmatrix},$$

where $\rho^2 = \theta - r^T r$. If the matrix $\bar{Z}_{FR}^T H_{FR} \bar{Z}_{FR}$ is not positive definite, the Cholesky factorization may be undefined or ill-conditioned, and other techniques should be used to modify the factorization without excessive additional computation or loss of numerical stability (e.g., see Gill *et al.*, 1982a, for techniques applicable to quadratic programming).

The number of multiplications associated with deleting the i-th general constraint includes the following (where only the highest-order term is given): $\frac{3}{2}(m_L-i)^2$ to operate on T; $3n_{FR}(m_L-i)$ to transform Q; n_{FR}^2 to form $H_{FR}z$; $n_{FR}n_z$ to form $Z_{FR}^T H_{FR}z$; and $\frac{1}{2}n_z^2$ to compute the additional row of the Cholesky factor. It is clearly advantageous to delete constraints at the *end* of the list of general constraints in the working set; hence, the indices of general equality constraints are always placed at the beginning of the list.

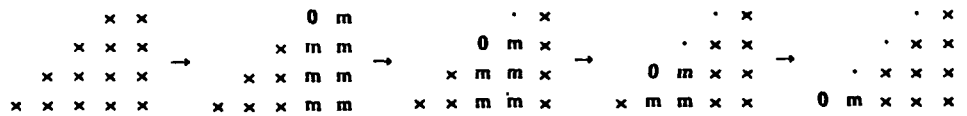
The justification for using the TQ factorization arises from this part of an active-set method. From a theoretical viewpoint, $\bar{Z}_{FR}$ would remain an orthogonal basis for the null space of $\bar{A}_{FR}$ regardless of the position in which the new column appeared. However, in order to update the Cholesky factors efficiently, the new column must appear *after* the columns of Z_{FR} (otherwise, (13) would not hold). The TQ factorization has an implementation advantage because the new column of $\bar{Z}_{FR}$ automatically appears in the correct position after deletion of a constraint from the working set. With other alternatives, the housekeeping associated with the update of R is more complicated. For example, in an implementation based on the LQ factorization, the new column might be moved to the end of Z_{FR}, or a list could be maintained of the locations of the columns of Z_{FR}; another alternative is to store the columns of Z_{FR} in reverse order (see Gill and Murray, 1977).

6.4. Deleting a bound. When a *bound* constraint is deleted from the working set, a previously fixed variable becomes free. In this case, the column dimension of A_{FR}, the column and row dimensions of Z_{FR} and the dimension of Q are increased by one; the dimension of T remains unaltered. In practice, the index of the newly freed variable is added at the end of the list of free variables. Hence, we assume without loss of generality that the $(n_{FR}+1)$-th variable is to be freed from its bound, so that $\bar{C}$ is defined by deleting row $n_{FR}+1$ of C. In effect, the boundary between A_{FR} and A_{FX} is "shifted" by one column; this corresponds to augmenting Q by a row and column of the identity, and reducing by one the dimension of the identity matrix associated with the fixed variables.

Let a denote the column of A corresponding to the newly freed variable, and let I_{FX}^- denote the $(n_{FX}-1)$-dimensional identity matrix. The result of deleting the bound constraint is then

$$\bar{C}\begin{pmatrix} Q & 0 \\ 0 & I_{FX} \end{pmatrix} = \begin{pmatrix} 0 & 0 & I_{FX}^- \\ A_{FR} & a & \bar{A}_{FX} \end{pmatrix} \begin{pmatrix} Q & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & I_{FX}^- \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & I_{FX}^- \\ 0 & T & a & \bar{A}_{FX} \end{pmatrix}.$$

To reduce the augmented matrix $(T \quad a)$ to the desired form $(0 \quad \bar{T})$, a backward sweep of column plane rotations (say, P) is applied in the order $(m_L+1, m_L), \ldots, (2,1)$, as shown in the following diagrams:



The rotations in P affect columns n_z+1 through $n_{FR}+1$ of the augmented Q. The first n_z columns of $\bar{Z}_{FR}$ are thus simply those of Z_{FR}, with a row of zeros added at the bottom. It follows

that $\bar{Z}_{FR}$ is given by

$$\bar{Z}_{FR} = \begin{pmatrix} Z_{FR} & \\ & z \\ 0 & \end{pmatrix}, \tag{14}$$

where the last element of z will be nonzero.

The effect of freeing the $(n_{FR} + 1)$-th variable is to augment H_{FR} by the row and column corresponding to the newly released variable. Since (14) is similar to (12), the Cholesky factors of the new projected Hessian can be obtained from the existing factors by performing one further step of the factorization as before (assuming that the updated projected Hessian is positive definite).

The number of multiplications associated with deleting a bound constraint includes the following, where only the highest-order term is given: $\frac{3}{2}m_L^2$ to operate on T; $3m_L n_{FR}$ to transform Q; n_{FR}^2, to form $H_{FR}z$; $n_{FR}n_z$ to form $Z_{FR}^T H_{FR} z$; and $\frac{1}{2}n_z^2$ to update R.

References

Bartels, R. H. (1980). A penalty linear programming method using reduced-gradient basis-exchange techniques, *Linear Algebra and its Applics.* **29**, pp. 17–32.

Biggs, M. C. (1972). "Constrained minimization using recursive equality quadratic programming", in *Numerical Methods for Non-Linear Optimization* (F. A. Lootsma, ed.), pp. 411–428, Academic Press, London and New York.

Buckley, A. G. (1975). An alternative implementation of Goldfarb's minimization algorithm, *Math. Prog.* **8**, pp. 207–231.

Cox, M. G. (1981). The least squares solution of overdetermined linear equations having band or augmented band structure, *IMA J. Numer. Anal.* **1**, pp. 3–22.

Dantzig, G. B. (1963). *Linear programming and extensions*, Princeton University Press, Princeton, New Jersey.

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

Gill, P. E. and Murray, W. (1973). A numerically stable form of the simplex algorithm, *Lin. Alg. Applics.* **7**, pp. 99–138.

Gill, P. E. and Murray, W. (1974). Newton-type methods for unconstrained and linearly constrained optimization, *Math. Prog.* **28**, pp. 311–350.

Gill, P. E., and Murray, W. (1977). "Linearly constrained problems, including linear and quadratic programming," in *The State of the Art in Numerical Analysis* (D. Jacobs, ed.), pp. 313–363, Academic Press, London and New York.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1982a). The design and implementation of a quadratic programming algorithm, Report SOL 82-8, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H. (1982b). Software for linearly constrained optimization, Report SOL 82-9, Department of Operations Research, Stanford University, California.

Gill, P. E., Murray, W. and Wright, M. H. (1981). *Practical Optimization*, Academic Press, London and New York.

Han, S.-P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming problems, *Math. Prog.* **11**, pp. 263–282.

Lawson, C. L., Hanson, R. J., Kincaid, D. R., and Krogh, F. T. (1979). Basic linear algebra subprograms for Fortran usage, *ACM Trans. Math. Software* **5**, pp. 308–325.

Mifflin, R. (1979). A stable method for solving certain constrained least-squares problems, *Math. Prog.* **16**, pp. 141–158.

Murray, W. (1969). "An algorithm for constrained minimization", in *Optimization* (R. Fletcher, ed.), pp. 247–258, Academic Press, London and New York.

Murray, W. and Wright, M. H. (1982). Computation of the search direction in constrained optimization algorithms, *Math. Prog. Study* **16**, pp. 62–83.

Murtagh, B. A. and Saunders, M. A. (1978). Large-scale linearly constrained optimization, *Math. Prog.* **14**, pp. 41–72.

Murtagh, B. A. and Saunders, M. A. (1982). A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints, *Math. Prog. Study* **16**, pp. 84–117.

Powell, M. J. D. (1977). A fast algorithm for nonlinearly constrained optimization calculations, Report DAMTP 77/NA 2, University of Cambridge, England.

Powell, S. (1975). A development of the product form algorithm for the simplex method using reduced transformation vectors, *Math. Prog. Study* **4**, pp. 93–107.

Robinson, S. M. (1972). A quadratically convergent algorithm for general nonlinear programming problems, *Math. Prog.* **3**, pp. 145–156.

Rosen, J. B. (1960). The gradient projection method for nonlinear programming, Part I — linear constraints, *SIAM J. Appl. Math.* **8**, pp. 181–217.

Rosen, J. B. and Kreuser, J. (1972). "A gradient projection algorithm for nonlinear constraints", in *Numerical Methods for Non-Linear Optimization* (F. A. Lootsma, ed.), pp. 297–300, Academic Press, London and New York.

Stewart, G. W. (1973). *Introduction to Matrix Computations*, Academic Press, London and New York.

Stoer, J. (1971). On the numerical solution of constrained least-squares problems, *SIAM J. Numer. Anal.* **8**, pp. 382–411.

Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem*, Oxford University Press.

Zoutendijk, G. (1970). "A product-form algorithm using contracted transformation vectors", in *Integer and Nonlinear Programming* (J. Abadie, ed.), pp. 511–523, North-Holland, Amsterdam.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 82-6	2. GOVT ACCESSION NO. AD-A115 854	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints.		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Philip E. Gill, Walter Murray Michael A. Saunders, Margaret H. Wright		8. CONTRACT OR GRANT NUMBER(s) N00014-75-C-0267 DAAG29-81-K-0156
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR-047-143
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		12. REPORT DATE May, 1982 13. NUMBER OF PAGES 13 pp.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Linear constraints; active-set methods; updating matrix factorizations; computer software; orthogonal factorization; TQ factorization; Cholesky factorization.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See other side.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

SOL 82-6: PROCEDURES FOR OPTIMIZATION PROBLEMS WITH A MIXTURE OF BOUNDS AND GENERAL LINEAR CONSTRAINTS (May 1982, 13 pp)

When describing active-set methods for linearly constrained optimization, it is often convenient to treat all constraints in a uniform manner. However, in many problems the linear constraints include simple bounds on the variables as well as general constraints. Special treatment of bound constraints in the implementation of an active-set method yields significant advantages in computational effort and storage requirements. In this paper, we describe how to perform the constraint-related steps of an active-set method when the constraint matrix is dense and bounds are treated separately. These steps involve updates to the TQ factorization of the working set of constraints and the Cholesky factorization of the projected Hessian (or Hessian approximation).


DATE
FILME
7-8