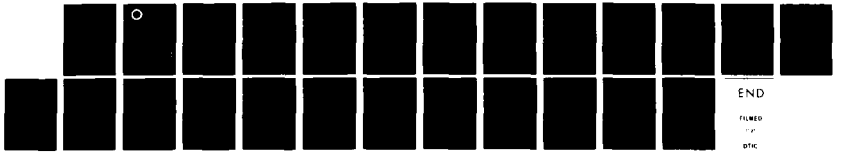
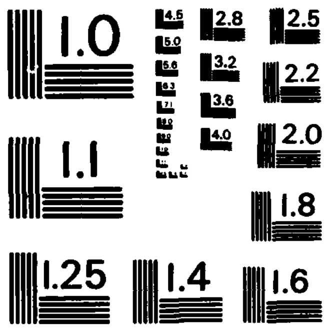
AD-A121 336 RANGE-SPACE METHODS FOR CONVEX QUADRATIC PROGRAMMING 1/1
(U) STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB
P E GILL ET AL. OCT 82 SOL-82-14 ARO-18424.8-MA
UNCLASSIFIED N00014-75-C-0267 F/G 12/1 NL





MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

ARO 18424.8-MA

AD A121336

Systems
Optimization
Laboratory

DTIC FILE COPY

Department of Operations Research
Stanford University
Stanford, CA 94305

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

RANGE-SPACE METHODS FOR CONVEX QUADRATIC PROGRAMMING

(12)

by

Philip E. Gill, Nicholas I.M. Gould,
Walter Murray, Michael A. Saunders,
Margaret H. Wright

TECHNICAL REPORT SOL 82-14

OCTOBER 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-7926009 and ECS-8012974; and Army Research Office Contract DAA29-79-C-0110. The work of Nicholas Gould was supported by the Science and Engineering Research Council of Great Britain.

Range-Space Methods for Convex Quadratic Programming

by

Philip E. Gill, Nicholas I. M. Gould, Walter Murray,
Michael A. Saunders and Margaret H. Wright

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

ABSTRACT

Range-space methods for convex quadratic programming improve in efficiency as the number of constraints active at the solution decreases. This paper describes in detail three alternative range-space methods, each based upon updating different matrix factorizations. It is shown that the choice of method depends upon the relative importance of storage needs, computational efficiency and numerical reliability.

The updating methods described are applicable to both primal and dual quadratic programming algorithms that use an active-set strategy.

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. The work of Nicholas Gould was supported by the Science and Engineering Research Council of Great Britain.

1. Introduction

This paper is concerned with methods for solving the *convex quadratic programming problem* — the minimization of a positive-definite quadratic form subject to a set of linear constraints on the variables. We shall assume the problem to be stated in the following form:

$$\text{QP} \qquad \begin{array}{ll} \underset{x \in \mathbb{R}^n}{\text{minimize}} & c^T x + \frac{1}{2} x^T H x \\ \text{subject to} & \ell \leq Ax \leq u, \end{array}$$

where H is a symmetric positive-definite matrix, and A is $m \times n$. (The methods to be described may be applied directly to *equality* constraints, which are omitted for simplicity. The adaptation of one of the methods to exploit the structure of *simple bound constraints* is described in Gill *et al.*, 1982.) The vector function $g(x) \equiv c + Hx$ will denote the gradient of the quadratic objective function.

Apart from the requirement of feasibility, the optimality conditions for problem QP involve only the constraints active (exactly satisfied) at the solution. Let x^* denote the (necessarily unique) solution of QP. Let A^* denote the matrix whose rows are the active constraints, and b^* the vector of corresponding components of ℓ or u, so that $A^* x^* = b^*$, with strict inequality in all other constraints. Let $\mathcal{L}$ denote the set of rows in A^* corresponding to active *lower bounds*, and $\mathcal{U}$ the set of rows in A^* corresponding to active *upper bounds*. There must exist a vector λ^* such that

$$A^{*T} \lambda^* = c + Hx^* = g(x^*), \tag{1.1}$$

where

$$\lambda_i^* \geq 0, \quad i \in \mathcal{L}; \quad \lambda_i^* \leq 0, \quad i \in \mathcal{U}.$$

Active-set methods are based on developing a *prediction* of the active set (the *working set*), which includes the constraints exactly satisfied at the current point (see, e.g., Gill and Murray, 1977). Let x_k denote the current iterate, and g_k its gradient; the t_k rows of the matrix A_k are defined as the coefficients of the constraints in the working set, and the vector b_k is composed of the corresponding components of ℓ and u (so that $A_k x_k = b_k$). The search direction p_k is chosen so that $x_k + p_k$ is the solution of a quadratic programming problem with the original objective function, subject to the *equality* constraints of the working set. With this definition, p_k is itself the solution of the quadratic program

$$\begin{array}{ll} \underset{p \in \mathbb{R}^n}{\text{minimize}} & g_k^T p + \frac{1}{2} p^T H p \\ \text{subject to} & A_k p = 0. \end{array} \tag{1.2}$$

The optimality conditions for (1.2) imply that there exists a vector of Lagrange multipliers (denoted by λ_k) such that

$$A_k^T \lambda_k = g_k + Hp_k. \tag{1.3}$$

If x_k is feasible and $A_k = A^*$, it follows from (1.1) and (1.3) that $x_k + p_k = x^*$ and $\lambda_k = \lambda^*$. Otherwise, the working set is changed by adding or deleting constraints until the correct active set is determined.

It is useful to classify active-set QP methods as either *range-space* or *null-space* methods. This terminology arises because the working set can be viewed as defining two complementary subspaces: the *range space* of vectors that can be expressed as linear combinations of the rows of A_k, and the *null space* of vectors orthogonal to the columns of A_k. Each iteration of a quadratic programming method involves computing p_k (and sometimes λ_k), and in many cases the work required in an iteration is directly proportional to the dimension of either the range space or the null space. For example, the methods of Murray (1971), Gill and Murray (1978), Bunch and Kaufman (1980) and Powell (1981) are null-space methods, and are most efficient when the number of constraints in the working set is close to n, since the dimension of the null space is then relatively small. (Some methods cannot be categorized as either range-space methods or null-space methods. See, for example, the methods proposed by Bartels, Golub and Saunders, 1970, Fletcher, 1971, and Goldfarb and Idnani, 1981.)

This paper is concerned with *range-space* methods, which increase in efficiency as the number of constraints in the working set *decreases*. In Sections 2, 3 and 4 we describe three alternative range-space methods. The differences in the methods arise because of the different emphasis placed on the relative importance of storage needs, computational efficiency and numerical reliability. In all cases we shall discuss primarily the details of how to compute p_k and λ_k, and not the various strategies for altering the working set. The techniques described may be applied in the implementation of both primal and dual quadratic programming algorithms that use an active-set strategy.

As general background for range-space methods, we observe that the constraints and optimality conditions for problem (1.3) imply that p_k and λ_k satisfy the $n + t_k$ linear equations

$$\begin{pmatrix} H & -A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} p_k \\ \lambda_k \end{pmatrix} = \begin{pmatrix} -g_k \\ 0 \end{pmatrix},$$

which may be solved in several different ways. The range-space approach takes advantage of the case where H is non-singular. If A_k has full rank, p_k and λ_k are defined by the following *range-space* equations:

$$A_k H^{-1} A_k^T \lambda_k = A_k H^{-1} g_k; \tag{1.4}$$

and

$$H p_k = A_k^T \lambda_k - g_k, \tag{1.5}$$

which underlie all the methods of this paper.

2. Method 1: a low-storage method

Probably the most straightforward range-space method is obtained by utilising the Cholesky factors of H and $A_k H^{-1} A_k^T$ to solve the range-space equations (1.4) and (1.5) directly. This method has been suggested with respect to general linearly constrained problems by Gill and Murray (1974, pp. 77–78) and Goldfarb (1976), and with respect to quadratic programming by Dax (1981).

We shall denote the required factorizations as

$$H = R^T R \quad \text{and} \quad A_k H^{-1} A_k^T = U_k^T U_k, \tag{2.1}$$

where R and U_k are upper-triangular matrices of dimension $n \times n$ and $t_k \times t_k$ respectively.

Note that R needs to be computed only once, and may be overwritten on H. In this section, all computations involving H are actually performed using R, but are represented in terms of H for simplicity. The matrix $A_k H^{-1} A_k^T$ changes in a special way as constraints enter and leave the working set. The associated update procedures for U_k will be summarized in this section; complete details are given in the Appendix (see also Dax, 1981).

The fundamental relationship connecting the updated Cholesky factors is the following. Consider the positive-definite symmetric matrix M with Cholesky factors $U^T U$, and the related positive-definite matrix $\bar{M}$ defined by

$$\bar{M} = \begin{pmatrix} M & m \\ m^T & \mu \end{pmatrix}, \tag{2.2}$$

where m is a column vector and μ a scalar. The Cholesky factor $\bar{U}$ of $\bar{M}$ is given by

$$\bar{U} = \begin{pmatrix} U & u \\ 0 & \rho \end{pmatrix}, \quad \text{where} \quad U^T u = m \quad \text{and} \quad \rho^2 = \mu - u^T u \tag{2.3}$$

(see Wilkinson, 1965, pp. 229–230 and Section A.2.1 of the Appendix). (If $\bar{M}$ is positive definite, $\mu - u^T u > 0$.)

For the remainder of this section we shall drop the subscript k, so that, for example, A denotes the matrix of constraints in the working set. Barred quantities will be associated with a single change in the working set. We shall show that the vector λ can be updated after every change in the working set with very little work beyond that required to update the Cholesky factor U. To demonstrate this, we require the following lemma, which we state without proof (see, for example, Cottle, 1974).

Lemma 1. *Let M ($M = U^T U$) be a positive-definite symmetric matrix and b a vector, and let x denote the solution of $Mx = b$. Consider the augmented matrix $\bar{M}$ ($\bar{M} = \bar{U}^T \bar{U}$) defined by (2.2) and (2.3), and the vector $\bar{b}$ defined by*

$$\bar{b} = \begin{pmatrix} b \\ \beta \end{pmatrix}, \tag{2.4}$$

where β is a scalar. If $\bar{M}$ is non-singular, then the solution of the system $\bar{M}\bar{x} = \bar{b}$ is given by

$$\bar{x} = \begin{pmatrix} x - \xi v \\ \xi \end{pmatrix},$$

where

$$U^T u = m, \quad Uv = u, \quad \rho^2 = \mu - u^T u \quad \text{and} \quad \xi = \frac{\beta - m^T x}{\rho^2}. \tag{2.5}$$

∎

Adding a constraint to the working set. When a constraint is *added* to the working set, a new row (say, a^T) becomes the *last* row of $\bar{A}$. The matrix $\bar{A} H^{-1} \bar{A}^T$ then has the form (2.2), with

$$M = AH^{-1}A^T, \quad m = AH^{-1}a, \quad \text{and} \quad \mu = a^T H^{-1} a, \tag{2.6}$$

so that its Cholesky factor $\bar{U}$ is given by (2.3).

Given the updated Cholesky factor $\bar{U}$ following a constraint addition, it is easy to obtain the updated vector of Lagrange multipliers, as shown in the following theorem.

Theorem 1. *Let λ and p denote the solutions to the range-space equations (1.4) and (1.5) at the point x. Let $\bar{x}$ ($\bar{x} = x + \alpha p$) be a point at which the row a^T is added to A. Assume that the updated Cholesky factor $\bar{U}$ of $\bar{A}H^{-1}\bar{A}^T$ has been computed, so that u and ρ of (2.3) are available. Then the Lagrange multiplier vector that solves (1.4) at $\bar{x}$ is given by the vector $\bar{\lambda}$, whose elements are defined by*

$$\bar{\lambda}_j = \lambda_j - \xi v_j, \quad j = 1, 2, \ldots, t, \quad \text{and} \quad \bar{\lambda}_{t+1} = \xi, \tag{2.7}$$

where v and ξ are defined by

$$Uv = u \quad \text{and} \quad \xi = \frac{(\alpha - 1)a^T p}{\rho^2}.$$

Proof. By construction of p and definition of $\bar{g}$,

$$AH^{-1}\bar{g} = AH^{-1}(g + \alpha Hp) = AH^{-1}g + \alpha Ap = AH^{-1}g. \tag{2.8}$$

Hence, Lemma 1 can be applied to the equations for λ and $\bar{\lambda}$, and the form (2.7) follows directly.

Since β in (2.4) is given by $a^T H^{-1}\bar{g}$, it follows from (2.5) that the scalar ξ is given by

$$\xi = \frac{a^T H^{-1}\bar{g} - m^T\lambda}{\rho^2}. \tag{2.9}$$

Substituting the expression $g + \alpha Hp$ for $\bar{g}$ and the definition of m from (2.6), the numerator of (2.9) becomes

$$\begin{aligned} a^T H^{-1}\bar{g} - m^T\lambda &= a^T H^{-1}(g + \alpha Hp) - a^T H^{-1}A^T\lambda \\ &= a^T H^{-1}(\alpha Hp - A^T\lambda + g). \end{aligned}$$

Using (1.5), this expression further simplifies to $(\alpha - 1)a^T p$. Hence, $\xi = (\alpha - 1)a^T p/\rho^2$. ∎

Theorem 1 shows that $\frac{1}{2}t^2$ multiplications are required to update λ when a constraint is added to the working set.

Deleting a constraint from the working set. First, suppose that $\bar{A}$ is given by A with its *last* row (denoted by a^T) deleted. In this case, the relationship between $AH^{-1}A^T$ and $\bar{A}H^{-1}\bar{A}^T$ is the *reverse* of (2.2), and hence the new Cholesky factor is simply a submatrix of the old, i.e.

$$U = \begin{pmatrix} \bar{U} & u \\ 0 & \rho \end{pmatrix}. \tag{2.10}$$

The following theorem shows that the updated multiplier vector can be computed with little effort in this case.

Theorem 2. *Let λ and p denote the solutions to the range-space equations at the point x. Suppose that the constraint corresponding to the last row of A is deleted from the working set at the point $z = x + p$. If $\bar{A}$ denotes the matrix of constraints in the working set after the row has been deleted, $\bar{U}$ denotes the Cholesky factor of $\bar{A}H^{-1}\bar{A}^T$, and u denotes the vector of the first $t-1$ elements in the last column of U, the vector of multipliers $\bar{\lambda}$ computed at z with the new working set is given by*

$$\bar{\lambda}_j = \lambda_j + \lambda_t v_j, \quad j = 1, 2, \ldots, t-1,$$

where v satisfies $\bar{U}v = u$.

Proof. Our assumption about the position of the deleted row means that

$$A^T H^{-1} A = \begin{pmatrix} \bar{A}H^{-1}\bar{A}^T & \bar{A}H^{-1}a \\ a^T H^{-1}\bar{A}^T & a^T H^{-1} a \end{pmatrix}. \tag{2.11}$$

Since λ solves (1.4) and (2.8) holds, we may apply Lemma 1 with $\bar{A}H^{-1}\bar{A}^T$ as M and $AH^{-1}A^T$ as $\bar{M}$. This gives

$$\lambda = \begin{pmatrix} \bar{\lambda} - \lambda_t v \\ \lambda_t \end{pmatrix},$$

where

$$\bar{A}H^{-1}\bar{A}^T v = \bar{A}H^{-1}a.$$

Because $U^T U = AH^{-1}A^T$, (2.10) and (2.11) imply that $\bar{U}^T u = \bar{A}H^{-1}a$; since $\bar{U}^T\bar{U} = \bar{A}H^{-1}\bar{A}^T$, it follows that v may be obtained by solving the triangular system $\bar{U}v = u$. ∎

Theorem 2 implies that the vector $\bar{\lambda}$ may be obtained in $\frac{1}{2}t^2$ multiplications.

In general, a row may be deleted from any position within A, so that Theorem 2 cannot be applied directly. In this case, the rows of A are permuted to move the deleted row to the last position and a sequence of plane rotations is applied to restore the permuted form of U to upper-triangular form (see Section A.1 of the Appendix). In this case, the relationship between U and $\bar{U}$ is of the form

$$QU\Pi = \begin{pmatrix} \bar{U} & u \\ 0 & \beta \end{pmatrix},$$

where Q is a sequence of plane rotations and Π is a permutation matrix. The number of multiplications required to perform these rotations when the j-th constraint is deleted is $3(t-j)^2$ (we assume the three-multiplication form of the plane rotations; see Gill *et al.*, 1974). Thus, the essential features of Theorem 2 remain unchanged, the only difference being the need to reorder the elements of λ before updating.

Discussion of the low-storage method. Two advantages of this method are the relative simplicity of the associated updates and the low storage requirements. Since R may be overwritten on the space required to store H, the storage space required for this method (beyond that needed to represent the problem) is of the order of $\frac{1}{2}t_{\max}^2$ locations, where $t_{\max}$ is the maximum number of constraints in the working set. In the case where there are always few constraints in the working set, the method requires very little extra storage. The disadvantage of the method lies in the possible magnification of any ill-conditioning in A and H during the implicit formation of the product $AH^{-1}A^T$.

The following table summarises the number of multiplications needed to perform each computation in the low-storage method.

Update λ	Compute p	Add constraint	Delete i-th constraint
$\frac{1}{2}t^2$	n^2+nt	$n^2+nt+\frac{1}{2}t^2$	$3(t-i)^2$

3. Method 2: a better conditioned method

The range-space equations (1.4) and (1.5) are not the best representation of p_k and λ_k for purposes of computation, since formation of the matrix $A_k H^{-1} A_k^T$ may exacerbate the conditioning of the original problem. This difficulty may be lessened by rewriting the equations in terms of the LY factorization of A_k, i.e.,

$$A_k = L_k Y_k^T, \tag{3.1}$$

where the t columns of Y_k are orthonormal n-vectors, and L_k is a $t \times t$ lower-triangular matrix. (The columns of Y_k form an orthonormal basis for the range space of the rows of A_k.)

Substituting (3.1) into (1.4) and (1.5), we obtain

$$L_k^T \lambda_k = \theta_k \tag{3.2}$$

$$H p_k = Y_k \theta_k - g_k, \tag{3.3}$$

where θ_k satisfies

$$Y_k^T H^{-1} Y_k \theta_k = Y_k^T H^{-1} g_k. \tag{3.4}$$

Equations (3.2) and (3.3) are theoretically equivalent to (1.4) and (1.5); however, they are likely to have superior numerical properties, since the orthogonality of the columns of Y_k means that the condition number of $Y_k^T H^{-1} Y_k$ is no greater than that of H. The use of the factorization (3.1) to derive a better conditioned projection matrix was suggested by Gill and Murray (1974, pp. 78–79) in the context of general linearly constrained problems; see also Goldfarb (1976).

Computation of λ_k and p_k from (3.2) and (3.3) may be implemented by storing the Cholesky factors of H (which need be computed only once), and updating the LY factorization of A_k and the Cholesky factors of $Y_k^T H^{-1} Y_k$ as the working set changes. A detailed description of the update procedures is given in Section A.2 of the Appendix. For simplicity, we shall drop the subscript k in the rest of this section; barred quantities will denote those resulting from a single change in the working set. Let U denote the Cholesky factor of $Y^T H^{-1} Y$, i.e.,

$$Y^T H^{-1} Y = U^T U.$$

Adding a constraint to the working set. When a constraint is added to the working set, a new row (say, a^T) is added to A. The effect of this change on the LY factorization is that a new column is added to Y, so that $\bar{Y}$ is given by

$$\bar{Y} = (\, Y \quad z \,).$$

In this case, the matrix $\bar{Y}^T H^{-1} \bar{Y}$ is of the augmented form $\bar{M}$ (2.2), with

$$M = Y^T H^{-1} Y, \quad m = Y^T H^{-1} z \quad \text{and} \quad \mu = z^T H^{-1} z.$$

Thus, exactly the same update procedures described in Theorem 2 can be applied to compute the updated Cholesky factorization of $\bar{Y}^T H^{-1} \bar{Y}$.

We now show that the vector θ of (3.4) can be updated rather than recomputed, using a technique similar to that given in Section 2.1 for updating λ.

Theorem 3. *Let p and θ denote the solutions to the equations (3.3) and (3.4) at the point x. Let $\bar{x}$ ($\bar{x} = x + \alpha p$) be a point at which the row a^T is added to A. Assume that the LY factorisation of $\bar{A}$ and the Cholesky factor $\bar{U}$ of $\bar{Y}^T H^{-1} \bar{Y}$ have been computed. Let u and ρ denote the corresponding portions of $\bar{U}$, as in (2.3), and let z denote the last column of $\bar{Y}$. Then the vector $\bar{\theta}$ that solves (3.4) at $\bar{x}$ is given by*

$$\bar{\theta}_j = \theta_j - \xi v_j, \quad j = 1, 2, \ldots, t, \quad \text{and} \quad \bar{\theta}_{t+1} = \xi,$$

where v and ξ are defined by

$$Uv = u \quad \text{and} \quad \xi = \frac{(\alpha - 1) z^T p}{\rho^2}.$$

Proof. Since $Y^T p = 0$, the proof is essentially the same as for Theorem 1. ∎

Theorem 3 shows that $\frac{1}{2}t^2$ additional multiplications are needed to update θ after a constraint is added to the working set.

Deleting a constraint from the working set. When a constraint is deleted from the working set, the LY factorization of A may be updated as described in Section A.2.2 of the Appendix, using a sequence of plane rotations that restore the lower-triangular form of L. The basic relationship between Y and $\bar{Y}$ in this case is

$$YP = (\bar{Y} \quad z), \tag{3.5}$$

where the orthonormal matrix P is the product of suitably chosen plane rotations. We assume that the Cholesky factor $\bar{U}$ of $\bar{Y} H^{-1} \bar{Y}^T$ is available.

In this situation, the vector $\bar{\theta}$ can be computed with only a small amount of additional work beyond that required to update L, Y and U, as indicated in the following theorem.

Theorem 4. *Let θ and p denote the solutions of (3.3) and (3.4) at the point x, and suppose that a constraint is deleted from the working set at the point $\bar{x} = x + p$. Assume that the updated matrix $\bar{Y}$ in the LY factorisation of $\bar{A}$, the updated Cholesky factor $\bar{U}$ of $\bar{Y}^T H^{-1} \bar{Y}$ and the related vector $\bar{u}$ have been computed. Then the solution $\bar{\theta}$ of (3.4) at $\bar{x}$ is given by*

$$\bar{\theta}_j = \omega_j + \omega_t v_j, \quad j = 1, 2, \ldots, t-1,$$

with

$$\omega = P^T \theta \quad \text{and} \quad \bar{U} v = \bar{u},$$

where P is defined in (3.5).

Proof. Multiplying (3.4) by P^T and inserting $\theta = P\omega$, we obtain

$$P^T Y^T H^{-1} Y P \omega = \begin{pmatrix} \bar{Y}^T H^{-1} \bar{Y} & \bar{Y}^T H^{-1} z \\ z^T H^{-1} \bar{Y} & z^T H^{-1} z \end{pmatrix} \omega = \begin{pmatrix} \bar{Y}^T H^{-1} g \\ z^T H^{-1} g \end{pmatrix}. \tag{3.6}$$

Since $Y^T H^{-1} g = Y^T H^{-1} g$, the form of (3.6) shows that Lemma 1 can be invoked with $\bar{Y}^T H^{-1} \bar{Y}$ and $Y^T H^{-1} Y$ taking the roles of M and $\bar{M}$ respectively. It follows that

$$\omega = \begin{pmatrix} \bar{\theta} - \omega_t v \\ \omega_t \end{pmatrix},$$

where v satisfies the single triangular system $\bar{U} v = \bar{u}$. ∎

Discussion of the better conditioned method. The numerical advantage of Method 2 compared to the low-storage method results from avoiding the use of λ_k in computing the search direction (see (1.5)). If $A_k H^{-1} A_k^T$ is ill-conditioned, the computed value of λ_k may contain significant errors, which may then propagate to p_k. The accuracy of θ_k in Method 2 depends on the conditioning of $Y_k^T H^{-1} Y_k$, which is no worse than that of H itself. A disadvantage of Method 2 is the additional storage required for the $n \times t_k$ matrix Y_k and the t_k-dimensional triangular factor of $Y_k^T H^{-1} Y_k$. Furthermore, the updates required at each iteration are more costly than in Method 1 or 3 (see Section 5).

The following table indicates the number of multiplications required to perform each computation during an iteration of Method 2.

Update θ	Compute p	Compute λ	Add constraint	Delete i-th constraint
$\frac{1}{2}t^2$	n^2+nt	$\frac{1}{2}t^2$	$n^2+3nt+\frac{1}{2}t^2$	$3n(t-i)^2+\frac{9}{2}(t-i)^2$

4. Method 3: the weighted Gram-Schmidt method

Following Bartels, Golub and Saunders (1970), we note that the range-space equations may be solved using the factorizations

$$H = R^T R \quad \text{and} \quad A_k R^{-1} = (\, L_k \quad 0\,) Q_k^T, \tag{4.1}$$

where R is the $n \times n$ Cholesky factor of H, L_k is a $t_k \times t_k$ lower-triangular matrix, and Q_k is an $n \times n$ orthogonal matrix. Note that $A_k H^{-1} A_k^T = L_k L_k^T$ and therefore the transpose of L_k is identical to the triangular matrix U_k (2.1) of Method 1.

The factorizations (4.1) provide a solution to the range-space equations (1.4) and (1.5) in the form:

$$L_k^T \lambda_k = Y_k^T R^{-T} g_k, \tag{4.2}$$

$$R p_k = -Z_k Z_k^T R^{-T} g_k, \tag{4.3}$$

where Y_k and Z_k are the $n \times t_k$ and $n \times (n - t_k)$ sections of the matrix Q_k, i.e.

$$Q_k = (\, Y_k \quad Z_k\,).$$

A variant of (4.1)–(4.3) has been used by Goldfarb and Idnani (1981), who recur the matrix $Q_k^T R^{-T}$.

We now propose an alternative method based on (4.1)–(4.3), in which we take advantage of the identity $Z_k Z_k^T \equiv I - Y_k Y_k^T$ in order to avoid storing Z_k. In place of the full orthogonal factorization in (4.1), we utilize the *weighted Gram-Schmidt factorization*

$$A_k R^{-1} = L_k Y_k^T, \tag{4.4}$$

which can be updated as described by Daniel *et al.* (1976) (see Section A.3 of the Appendix).

With this approach, λ_k could be recurred using the results of Theorems 1 and 2 (this follows directly from the equivalence of U_k and the transpose of L_k). However, we shall show that a more efficient method may be obtained by recurring three vectors u_k, v_k and w_k, which are defined by the relations

$$R^T u_k = g_k, \quad v_k = Y_k^T u_k \quad \text{and} \quad w_k = Y_k v_k - u_k. \tag{4.5}$$

(Note that $Y_k^T w_k = 0$.) Substitution into (4.2) and (4.3) allows λ_k and p_k to be defined from

$$L_k^T \lambda_k = v_k \tag{4.6}$$

and

$$R p_k = w_k. \tag{4.7}$$

In practice, initial vectors u_0, v_0 and w_0 are defined from (4.5) in terms of an initial point x_0 and the weighted Gram-Schmidt factors of a corresponding initial working set A_0. (The point x_0 must satisfy $A_0 x_0 = b_0$.) Thereafter, the vectors u_k, v_k and w_k can be updated at negligible cost, as we show below. The principal computational effort per iteration lies in updating the factorization (4.4) as the working set changes, and in computing p_k (and λ_k when needed) from (4.6) and (4.7).

As in the last two sections, we shall drop the subscript k and use barred quantities to denote a single change in the working set.

Adding a constraint to the working set. When a constraint is *added* to the working set, a new row (say, a^T) becomes the *last* row of $\bar{A}$. The relationship between the old and new matrices Y and $\bar{Y}$ of (4.4) is given by

$$\bar{Y} = (Y \quad z) \tag{4.8}$$

(see Section A.3.1 of the Appendix). The following theorem describes how the quantities u, v and p may be updated following a constraint addition.

Theorem 5. *Let p denote the vector that satisfies the range-space equation (1.5) at the point x. Let u, v and w satisfy (4.5) at the point x. Let $\bar{x}$ ($\bar{x} = x + \alpha p$) be a point at which the row a^T is added to A. Assume that the updated factors $\bar{L}$ and $\bar{Y}$ of $\bar{A} = \bar{L}\bar{Y}^T R$ have been computed, and that z, the new last column of $\bar{Y}$, is available. The vectors u, v and w are updated as follows:*

(i) $$\bar{u} = u + \alpha w; \tag{4.9}$$

(ii) $$\bar{v} = \begin{pmatrix} v \\ \nu \end{pmatrix}, \quad \text{where} \quad \nu = z^T \bar{u};$$

(iii) $$\bar{w} = (1 - \alpha) w + \nu z.$$

Proof. Using (4.1), (4.7) and the quadratic nature of the objective function, we have

$$\begin{aligned} R^T \bar{u} &= \bar{g} \\ &= g + \alpha H p \\ &= R^T u + \alpha R^T R p \\ &= R^T u + \alpha R^T w, \end{aligned}$$

and (4.9) follows immediately.

To prove *(ii)*, we use the definition of $\bar{v}$, (4.8) and (4.9) to give

$$\bar{v} = \bar{Y}^T\bar{a} = \begin{pmatrix} Y^T \\ z^T \end{pmatrix}\bar{a} = \begin{pmatrix} Y^Tu + \alpha Y^Tw \\ z^T\bar{a} \end{pmatrix}.$$

Since $Y^Tw = 0$ and $v = Y^Tu$, this proves *(ii)*. Similarly,

$$\bar{w} = \bar{Y}\bar{a} - \bar{v} = (Y \quad z)\begin{pmatrix} v \\ \nu \end{pmatrix} - (u + \alpha w),$$

which reduces to *(iii)* after substituting $Yv - u = w$. ∎

Note that in a dual QP algorithm that retains *dual feasibility*, the steplength $\alpha = 1$ will usually be taken when a constraint is added to the working set; cases *(i)* and *(iii)* then simplify. If further constraints are added at the same point, Theorem 5 remains true with $\alpha = 0$, $p = 0$, and $w = 0$.

Deleting a constraint from the working set. When a constraint is deleted from the working set, the relationship between the old and new matrices Y and $\bar{Y}$ of (4.4) is given by

$$YP = (\bar{Y} \quad z), \tag{4.10}$$

where the orthonormal matrix P is the product of suitably chosen plane rotations (see Section A.3.2 of the Appendix). The following theorem indicates how the quantities u, v and w may be updated in this case.

Theorem 6. *Let u, v and w satisfy (4.5) at the point x, and suppose that a constraint is deleted from the working set at the point $\bar{x} = x + \alpha p$, where p satisfies (4.7). Assume that the updated factors $\bar{L}$ and $\bar{Y}$ of $\bar{A} = \bar{L}\bar{Y}^TR$ have been computed, so that the relationship between the old and new orthogonal factors is given by (4.10). Then*

(i) $$\bar{a} = u + \alpha w; \tag{4.11}$$

(ii) $$\begin{pmatrix} \bar{v} \\ \nu \end{pmatrix} = P^Tv, \quad \text{where} \quad \nu = z^T\bar{a}; \tag{4.12}$$

(iii) $$\bar{w} = (1 - \alpha)w - \nu z. \tag{4.13}$$

Proof. Result *(i)* follows as in Theorem 5. To prove *(ii)*, note that, since $\bar{v} = \bar{Y}^T\bar{a}$, we may write

$$\begin{pmatrix} \bar{v} \\ \nu \end{pmatrix} = \begin{pmatrix} \bar{Y}^T \\ z^T \end{pmatrix}\bar{a},$$

where $\nu = z^T\bar{a}$. Using (4.10) and (4.11) gives

$$\begin{pmatrix} \bar{v} \\ \nu \end{pmatrix} = P^TY^T(u + \alpha w).$$

Since $Y^Tw = 0$ and $Y^Tu = v$, this gives the desired result.

Finally, to prove *(iii)*, we use the definition of $\bar{w}$ to write the identity

$$\bar{w} = \bar{Y}\bar{v} - \bar{u} = (\bar{Y} \quad z)\begin{pmatrix} v \\ \nu \end{pmatrix} - \bar{u} - \nu z.$$

Using (4.10), (4.11) and (4.12) gives

$$\bar{w} = YPP^T v - u - \alpha w - \nu z.$$

Since P is orthonormal and $Yv - u = w$, this expression simplifies to (4.13), as required. ∎

Note that a *primal* QP algorithm will usually delete a constraint only when $\alpha = 1$, in which case *(i)* and *(iii)* simplify. If more than one constraint is deleted at the same point, the theorem remains true with $\alpha = 0$, $p = 0$ and $w = 0$.

Discussion of the weighted Gram-Schmidt method. The major storage required by the weighted Gram-Schmidt (WGS) method beyond that for the low-storage method is for the $n \times t$ orthonormal matrix Y_k. The WGS method has a storage advantage compared to Method 2, since it requires storage of only one t-dimensional triangular matrix. A numerical advantage of the WGS method (as of Method 2) is that λ_k is not used explicitly to compute p_k. However, the conditioning of $A_k R^{-1}$ may be worse than that of A_k itself, and hence there may be more difficulties with the factorisation (4.4) than with (3.1).

The following table gives the number of multiplications needed to perform each computation in the weighted Gram-Schmidt method.

Compute p	Compute λ	Add constraint	Delete i-th constraint
$\frac{1}{2}n^2$	$\frac{1}{2}t^2$	$\frac{1}{2}n^2+2nt$	$3n(t-i)+3(t-i)^2$

5. Conclusions

Table 1 summarises the major work (measured by multiplications) associated with Methods 1, 2 and 3. In any active-set method, the quantities $\hat{A}x$ and $\hat{A}p$ must be computed, where $\hat{A}$ contains the constraints not already in the working set. In practice, the vector $\hat{A}p$ is computed, and $\hat{A}x$ is updated using $\hat{A}\bar{x} = \hat{A}x + \alpha\hat{A}p$. The work needed to perform this operation ($n(m-t)$ multiplications) is not included in the table, since it is the same for all methods.

The table gives the multiplication counts for two types of iteration of a typical feasible-point active-set QP method: (i) compute p and add a constraint; (ii) compute p, compute λ and delete a constraint. As indicated by the summary tables for each method, the amount of work required to delete a constraint depends on the index of the constraint to be deleted. The figure in the table corresponds to the case when the deleted constraint is the middle row of the working set (i.e., has index $\frac{1}{2}t$). The "average" figure is the average work between iterations of type (i) and (ii). The "average" value is given in a general form, and for three specific values of t ($t = 0$, $\frac{1}{2}n$ and n), in order to indicate the relative efficiencies of the three methods.

Table 2 summarises the factorisations used in each of the three methods, and the associated storage requirements. The entries that define storage give the number of floating-point words

Table 1

Summary of Work Required for Range-Space QP Methods

Cost to	Method 1 (Low Storage)	Method 2 (Better Conditioned)	Method 3 (WGS)
Compute p, add constraint	$2n^2+2nt+t^2$	$2n^2+4nt+t^2$	n^2+2nt
Compute p and λ, delete constraint	$n^2+nt+\frac{5}{4}t^2$	$n^2+\frac{5}{2}nt+\frac{17}{8}t^2$	$\frac{1}{2}n^2+\frac{3}{2}nt+\frac{5}{4}t^2$
Average	$\frac{3}{4}n^2+\frac{3}{2}nt+\frac{9}{8}t^2$	$\frac{3}{2}n^2+\frac{13}{4}nt+\frac{25}{16}t^2$	$\frac{5}{4}n^2+\frac{7}{4}nt+\frac{5}{8}t^2$
Average ($t=0$)	$1.5n^2$	$1.5n^2$	$\frac{3}{4}n^2$
Average ($t=\frac{1}{2}n$)	$2.5n^2$	$3.5n^2$	$1.6n^2$
Average ($t=n$)	$4.1n^2$	$6.3n^2$	$3.1n^2$

Table 2

Summary of Storage Required for Range-Space QP Methods

	Method 1 (Low Storage)	Method 2 (Better Conditioned)	Method 3 (WGS)
Storage for A	mn	Same	Same
Storage for H (R^TR)	$\frac{1}{2}n^2$	Same	Same
Additional factorizations	$A^TH^{-1}A=U^TU$	$A=LY^T$ $Y^TH^{-1}Y=U^TU$	$AR^{-1}=LY^T$
Additional storage	$\frac{1}{2}t_{max}^2$	$t_{max}^2+nt_{max}$	$\frac{1}{2}t_{max}^2+nt_{max}$

required for a dense problem. The entry "Same" means that the storage required is the same for all methods.

If only one of the three methods were to be chosen for general use, we would favor the choice of the weighted Gram-Schmidt method (Method 3). Its storage requirements are relatively modest, and it tends to encounter relatively few difficulties with numerical stability. Although the amount of storage required is more than that for Method 1, in our view the superior numerical properties justify the additional storage cost. Method 2 is the most numerically stable, but requires significant additional storage and work.

The methods described in this paper should be effective in solving QP problems with few constraints whenever the size or structure of the Hessian is such that its Cholesky factors can be computed and stored. Thus, these methods may be applied directly to large problems in which the Hessian is suitably sparse and structured (for example, when the Hessian is diagonal or banded).

Sparsity in the *constraints* will not tend to be helpful in Methods 1 and 3, since sparsity in A does not carry over to the factorizations involving A. For Method 2, it might be possible to compute a compact representation of Y if A is sparse.

If the available storage is sufficient for a $t \times t$ matrix but not an $n \times t$ matrix, Method 1 is the only possible choice. If an $n \times t$ matrix may be stored, any of the methods may be used. Unless numerical stability is of critical importance, Method 2 would generally not be considered. The choice between Methods 1 and 3 depends on the relative ease with which the system of equations $Hx = b$ can be solved given a factorization of H. Methods 1 and 3 are similar in efficiency if this process requires between nt and $2nt$ operations; Method 1 is preferred if the number of operations is less than nt, and Method 3 if the number is greater than $2nt$.

References

Bartels, R. H., Golub, G. H. and Saunders, M. A. (1970). "Numerical techniques in mathematical programming", in *Nonlinear Programming* (J. B. Rosen, O. L. Mangasarian and K. Ritter, eds.), pp. 123–176, Academic Press, London and New York.

Bunch, J. R. and Kaufman, L. C. (1980). A computational method for the indefinite quadratic programming problem, *Linear Algebra and its Applics.* **34**, pp. 341–370.

Cottle, R. W. (1974). Manifestations of the Schur complement, *Linear Algebra and its Applics.* **8**, pp. 189–211.

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

Daniel, J. W., Gragg, W. B., Kaufman, L. C. and Stewart, G. W. (1976). Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, *Mathematics of Computation* **30**, pp. 772–795.

Dax, A. (1981). An active set algorithm for convex quadratic programmming, manuscript, Hydrological Service, Jerusalem, Israel.

Fletcher, R. (1971). A general quadratic programming algorithm, *J. Inst. Maths. Applics.* **7**, pp. 76–91.

Gill, P. E., Golub, G. H., Murray, W. and Saunders, M. A. (1974). Methods for modifying matrix factorisations, *Mathematics of Computation* **28**, pp. 505–535.

Gill, P. E., Gould, N. I. M., Murray, W., Saunders, M. A., and Wright, M. H. (1982). A range-space method for quadratic programming problems with bounds and general constraints, Report SOL 82-15, Department of Operations Research, Stanford University, Stanford, California.

Gill, P. E. and Murray, W. (1974). "Quasi-Newton methods for linearly constrained optimization", in *Numerical Methods for Constrained Optimisation* (P. E. Gill and W. Murray, eds.), pp. 67–92, Academic Press, London and New York.

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. and Murray, W. (1978). Numerically stable methods for quadratic programming, *Math. Prog.* 14, pp. 349–372.

Goldfarb, D. (1976). Matrix factorizations in optimization of nonlinear functions subject to linear constraints, *Math. Prog.* 10, pp. 1–31.

Goldfarb, D. and Idnani, A. (1981). A numerically stable dual method for solving strictly convex quadratic programs, Technical Report CS 81-102, Department of Computer Sciences, The City University of New York, New York.

Murray, W. (1971). An algorithm for finding a local minimum of an indefinite quadratic program, Report NAC 1, National Physical Laboratory, England.

Powell, M. J. D. (1981). "An upper-triangular matrix method for quadratic programming", in *Nonlinear Programming 4* (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds.), pp. 1–24, Academic Press, London and New York.

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

Appendix — Methods for modifying matrix factorizations

In this Appendix we shall outline techniques for updating the matrix factorisations associated with a $t \times n$ matrix A and the $n \times t$ orthonormal matrix Y that forms a basis for the range space of the rows of A. The discussion is roughly divided into two parts (although much of the basic theory is common to both parts). In the first part, we consider how to update the factorizations

$$A = LY^T \quad \text{and} \quad Y^TMY = U^TU, \tag{A1}$$

where M is an $n \times n$ positive-definite symmetric matrix and U is a $t \times t$ upper-triangular matrix, when row changes are made to the matrix A.

In the second part we are concerned with updates following a row change in A of the factorization

$$A = LY^TR,$$

where H is an $n \times n$ symmetric positive-definite matrix with Cholesky factor R (so that $H = R^TR$), and L is a $t \times t$ lower-triangular matrix.

Some of the procedures described here are simple extensions of those described by Gill *et al.* (1974) and Daniel *et al.* (1976). Some methods that have been described previously are repeated here for completeness.

The discussion of the modifications 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). Throughout the Appendix we shall assume the three-multiplication form of a plane rotation; see Gill *et al.* (1974).

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.

A.1. A basic tool — updating after a column permutation

One of the basic operations in updating factorisations is the need to restore a matrix to triangular form after one of its rows or columns is moved to the first or last position. We shall illustrate this operation in the case where it is required to move an arbitrary column of an upper-triangular matrix R to the last position. There is no loss of generality if we consider moving the *first* column to the last position (this is the most expensive case to consider).

A.1.1. Method 1: using a column interchange. Suppose that Π is a permutation matrix that interchanges the first and last columns of R. We shall use a 5×5 example to illustrate the

column interchange. The permuted matrix has the form

$$R\Pi = \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ \times & & \times & \times & \\ \times & & & \times & \\ \times & & & & \end{pmatrix}.$$

We transform $R\Pi$ to upper-triangular form in two stages. First, we apply a sequence of plane rotations on the left so that the "vertical spike" of nonzero sub-diagonal elements is transformed into a "horizontal spike". The horizontal spike is eliminated with another sequence of plane rotations applied on the left.

To eliminate the vertical spike, a backward sweep of plane rotations is applied to the rows of $R\Pi$ in the planes $(n, n-1), (n, n-2), \ldots, (n, 2)$. We illustrate the effect of the transformation on the 5×5 example.

$$\begin{matrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ \times & & \times & \times & \\ \times & & \times & & \\ \times & & & & \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ \times & & \times & \times & \\ 0 & & \mathbf{m} & & \\ \mathbf{m} & & \mathbf{f} & & \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ 0 & & \mathbf{m} & \mathbf{m} & \\ \cdot & & \times & & \\ \mathbf{m} & & \mathbf{f} & \mathbf{m} & \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \times & \times \\ 0 & \mathbf{m} & \mathbf{m} & \mathbf{m} & \\ \cdot & & \times & \times & \\ \cdot & & & \times & \\ \mathbf{m} & \mathbf{f} & \mathbf{m} & \mathbf{m} & \end{matrix}$$

The resulting matrix is restored to upper-triangular form by a forward sweep of (row) rotations in the planes $(1, n), (2, n), \ldots, (n-1, n)$. In the 5×5 case we have

$$\begin{matrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \\ & & \times & \times & \\ & & & \times & \\ \times & \times & \times & \times & \end{matrix} \rightarrow \begin{matrix} \mathbf{m} & \mathbf{m} & \mathbf{m} & \mathbf{m} & \mathbf{m} \\ & \times & \times & \times & \\ & & \times & \times & \\ & & & \times & \\ 0 & \mathbf{m} & \mathbf{m} & \mathbf{m} & \mathbf{f} \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \times & \times \\ \mathbf{m} & \mathbf{m} & \mathbf{m} & \mathbf{f} & \\ & \times & \times & \times & \\ & & \times & & \\ \cdot & 0 & \mathbf{m} & \mathbf{m} & \mathbf{m} \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ \mathbf{m} & \mathbf{m} & \mathbf{f} & & \\ \times & & & & \\ \cdot & \cdot & 0 & \mathbf{m} & \mathbf{m} \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ \times & \times & \times & & \\ \mathbf{m} & \mathbf{f} & & & \\ \cdot & \cdot & \cdot & 0 & \mathbf{m} \end{matrix}$$

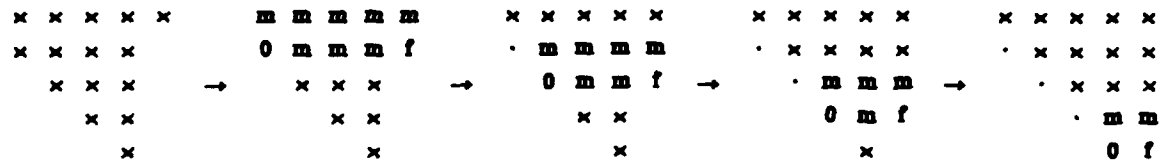
If both sweeps of plane rotations are denoted by a single orthonormal matrix P, we have

$$\bar{R} = PR\Pi. \tag{A2}$$

A.1.2. Method 2: using column shifts. There are other permutation matrices Π that will move the first column of R to the last position. A more common method is to move the first column to the last position and shift all the remaining columns one position to the left. In this case, the permuted matrix has the upper-Hessenberg form

$$R\Pi = \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ & \times & \times & \times & \\ & & \times & \times & \\ & & & \times & \end{pmatrix}.$$

This matrix is restored to upper-triangular form by a forward sweep of row rotations applied on the left to eliminate the subdiagonal elements, as shown in the following diagram.



The resulting transformation is of the form (A2) except that P comprises a single sweep of rotations instead of a double sweep.

A.1.3. Discussion. Clearly, the updating techniques of the last two sections are intimately related. (For example, in Section A.1.2 the transformation of the upper-Hessenberg matrix to triangular form could have been effected by first applying the permutation that moves the first row to the last position and shifts the remaining rows up by one position. The intermediate matrix is then triangular, with a "horizontal spike" in the last row. The spike can be eliminated using the method of Section A.1.1.) We have considered these transformations separately because each of the two methods will be most suitable under some circumstances. For example, the rotations used to transform R may also be applied to another matrix that itself must be maintained in some specific form (this is the case for the methods described in Section A.3). In another case, the triangular matrix may need to be stored as a one-dimensional array for maximum efficiency in some high-level computer languages. In this case, Method 1 would be most suitable because no new elements are created outside the triangular structure (except for the spikes, which can be held in a work vector).

If there is no overriding reason to use one method or the other, the decision may be influenced by other factors. For example, although Method 1 requires two sweeps of (row) rotations, some of the elements in the vertical and horizontal spikes could be zero, in which case the corresponding rotations may be skipped. In contrast, although Method 2 appears to require just one row sweep, the operations involved in shifting the columns one position to the left should be regarded as being equivalent to a sweep of "cheap" column rotations. Also, assuming R to be nonsingular, the rotations in the subsequent row sweep can never be skipped (although some of them could be simple interchanges).

In the last two sections we have shown that the restoration of a triangular matrix to triangular form after a row or column permutation implicitly or explicitly involves the application of plane rotations to one or more of the following three types of matrix:

$$\begin{pmatrix} R \\ v^T \end{pmatrix} \quad \text{"horizontal spike"}$$

$$(v \;\; R) \quad \text{"vertical spike"} \tag{A3}$$

$$(R \;\; v) \quad \text{"upper-Hessenberg"} \tag{A4}$$

The structure typified by these three standard cases will occur repeatedly in later sections.

A.2. Updating the factors of A and Y^TMY

A.2.1. Adding a new row to A. When a new row is added to A, the row dimension of L and the column dimension of Y in (A1) both increase by one. Without loss of generality we shall assume

that A is a $t \times n$ matrix, and that the new row is added in the last position, i.e.,

$$\bar{A} = \begin{pmatrix} A \\ a^T \end{pmatrix}.$$

The computation of the new column of Y is equivalent to a single step of the row-wise computation of the orthogonal factorization of $\bar{A}$ by the modified Gram-Schmidt algorithm. The new orthogonal factor $\bar{Y}$ is defined as

$$\bar{Y} = (Y \quad z), \tag{A5}$$

where z is given by a multiple of the vector a orthogonalised with respect to the orthonormal set of columns of Y, i.e. $z = \tau(I - YY^T)a$, where τ is a normalising factor. For more details of the computation of z, including the use of reorthogonalization, see Daniel *et al.* (1976).

Given z, the new row of L is found from the identity

$$\bar{A} = \begin{pmatrix} A \\ a^T \end{pmatrix} = \begin{pmatrix} L & 0 \\ l^T & \gamma \end{pmatrix} \begin{pmatrix} Y^T \\ z^T \end{pmatrix}, \tag{A6}$$

where the vector l and scalar γ are to be determined. The last row of this identity gives

$$a = Yl + \gamma z. \tag{A7}$$

Premultiplication of this equation by Y^T gives $l = Y^T a$ because of the orthogonality of the columns of $(Y \quad z)$. Similarly, premultiplication of (A7) by z^T gives $\gamma = z^T a$.

The relationship (A5) implies that

$$\bar{Y}^T M \bar{Y} = \begin{pmatrix} Y^T MY & Y^T Mz \\ z^T MY & z^T Mz \end{pmatrix}.$$

The Cholesky factor $\bar{U}$ of $\bar{Y}^T M \bar{Y}$ is given by

$$\bar{U} = \begin{pmatrix} U & u \\ 0 & \rho \end{pmatrix}, \quad \text{where} \quad U^T u = Y^T Mz \quad \text{and} \quad \rho^2 = z^T Mz - u^T u$$

(see Wilkinson, 1965, pp. 229–230). If $\bar{Y}^T M \bar{Y}$ is positive definite, $z^T Mz - u^T u > 0$.

The work necessary to update the factors L, Y and U when adding a new row to a $t \times n$ matrix A is dominated by the $\frac{1}{2}t^2 + n^2 + nt$ multiplications necessary to form u and ρ, and the $2(k+1)nt$ multiplications necessary to form z and the new row of L, where k ($k \geq 0$) is the number of reorthogonalization steps required. For well conditioned problems, reorthogonalization is usually not required.

A.2.2. Deleting a row from A. Suppose now that a row (say, the i-th) is *deleted* from a $t \times n$ matrix A, so that the row dimension of A, the column dimension of Y and the dimension of L in (A1) are *decreased* by one. The method given in the last section for *adding* a row to a matrix implies that if the *last* row of A is deleted, the new factors are identical to the old except that the last column of Y and row and column of L are omitted. Consequently, in the general case, the only nontrivial work necessary to perform a column deletion is that required to update the

factors after the deleted column is moved to the last position. We shall show that this updating can be achieved using the methods of Section A.1.

From (A1), we have

$$\begin{pmatrix} \bar{A} \\ a^T \end{pmatrix} = SY^T, \tag{A8}$$

where S is a $t \times t$ matrix that is identical to L except that the i-th row is moved to the last position and all the remaining rows have been shifted upwards. For example, in the case $t = 6$ and $i = 3$, we have

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

The matrix comprising the last $t-i+1$ columns of S is the transpose of a matrix of the form (A4) and it may be restored to lower-triangular form using a forward sweep of column rotations (say, P) as in Method 2 of Section A.1.2. The effect of these rotations is to give the factorisation

$$\begin{pmatrix} \bar{A} \\ a^T \end{pmatrix} = \tilde{L}P^TY^T,$$

where $\tilde{L}$ is a $t \times t$ lower-triangular matrix. Let $\tilde{L}$ and the product YP be partitioned so that

$$\tilde{L} = \begin{pmatrix} \bar{L} & 0 \\ l^T & \gamma \end{pmatrix}, \quad \text{and} \quad YP = (\bar{Y} \quad z), \tag{A9}$$

where $\bar{L}$ is a $(t-1) \times (t-1)$ lower-triangular matrix, $\bar{Y}$ is an $n \times (t-1)$ orthogonal matrix, and z is an n-vector. A comparison of (A9) with (A6) shows that $\bar{A} = \bar{L}\bar{Y}^T$ as required.

The plane rotations applied to Y also transform the Cholesky factor U of Y^TMY. The order of the rotations in P has been chosen so that each transformation introduces a single new subdiagonal element in the upper-triangular matrix U, as shown in the following sequence of diagrams.

$$\begin{matrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \end{matrix} \rightarrow \begin{matrix} \times & \times & \mathbf{m} & \mathbf{m} & \times & \times \\ & \times & \mathbf{m} & \mathbf{m} & \times & \times \\ & & \mathbf{m} & \mathbf{m} & \times & \times \\ & & \mathbf{f} & \mathbf{m} & \times & \times \\ & & & & \times & \times \\ & & & & & \times \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \mathbf{m} & \mathbf{m} & \times \\ & \times & \times & \mathbf{m} & \mathbf{m} & \times \\ & & \times & \mathbf{m} & \mathbf{m} & \times \\ & & \times & \mathbf{m} & \mathbf{m} & \times \\ & & & \mathbf{f} & \mathbf{m} & \times \\ & & & & & \times \end{matrix} \rightarrow \begin{matrix} \times & \times & \times & \times & \mathbf{m} & \mathbf{m} \\ & \times & \times & \times & \mathbf{m} & \mathbf{m} \\ & & \times & \times & \mathbf{m} & \mathbf{m} \\ & & \times & \times & \mathbf{m} & \mathbf{m} \\ & & & \times & \mathbf{m} & \mathbf{m} \\ & & & & \mathbf{f} & \mathbf{m} \end{matrix}$$

The transformed matrix UP is now in upper-Hessenberg form similar to (A4). As described in Section A.1.2, this matrix may be restored to upper-triangular form by a (partial) sweep of row rotations (say, the matrix Q), which is applied on the *left* to eliminate the subdiagonal elements.

The relationship between U and its transformed version is

$$QUP = \begin{pmatrix} \bar{U} & a \\ 0 & \beta \end{pmatrix},$$

where $\bar{U}$ is the upper-triangular factor of $\bar{Y}^TM\bar{Y}$.

The work associated with deleting a row from A comprises $\frac{3}{2}(t-i)^2$ multiplications to restore S, $3n(t-i)$ multiplications to compute YP, and $3(t-i)^2$ multiplications for the two sweeps of rotations applied to U.

A.3. Updating the factorization $A = LY^TR$

A.3.1. Adding a row to A. When a new row is added to A, the row dimension of L and the column dimension of Y both increase by one. Suppose that A is a $t \times n$ matrix. If the row is added to the last position, the updating of Y and L is identical to that described in Section A.2.1. In this case the new column z is given by a multiple of the vector $R^{-T}a$ orthogonalized with respect to the orthonormal set of columns of Y, i.e. $z = \tau(I - YY^T)q$, where q satisfies $R^Tq = a$ and τ is a normalizing factor.

The number of multiplications associated with adding a new row to A includes the following (where only the highest-order term is given): $\frac{1}{2}n^2$ to form q; and $2(k+1)nt$ to form z and the new row of L, where k is the number of reorthogonalization steps required.

A.3.2. Deleting a row from A. If the i-th row, say a^T, is deleted from a $t \times n$ matrix A, the column dimension of Y and the dimension of L are *decreased* by one. (As in Section A.3.1, the dimension of R is unaltered.) The method used to update the factors is identical to that given in Section A.2.2, except that it is not necessary to select P so that the structure of the factor U is maintained. In particular, the factorization

$$\begin{pmatrix} \bar{A} \\ a^T \end{pmatrix} = SY^TR,$$

analogous to (A8) may be obtained by interchanging the i-th and t-th rows of A. In this case, the matrix S has a form similar to the transpose of (A3) and may be reduced to lower-triangular form by Method 2 of Section A.1.1.

The work associated with deleting the i-th row from A comprises $3(t-i)^2$ multiplications to triangularize S and $3n(t-i)$ multiplications to compute YP.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SOL 82-14	2. GOVT ACCESSION NO. AD-A121336	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Range-Space Methods for Convex Quadratic Programming		5. TYPE OF REPORT & PERIOD COVERED Technical Report 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) P.E. Gill, N.I.M. Gould, W. Murray, M.A. Saunders, M.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 October 1982 13. NUMBER OF PAGES 20
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)

quadratic programming
range-space methods

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

Range-space methods for convex quadratic programming improve in efficiency as the number of constraints active at the solution decreases. This paper describes in detail three alternative range-space methods, each based upon updating different matrix factorizations. It is shown that the choice of method depends upon the relative importance of storage needs, computational efficiency and numerical reliability.

The updating methods described are applicable to both primal and dual quadratic programming algorithms that use an active-set strategy.

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

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)