Mathematical Programming 30 (1984) 176–195
North-Holland

# A WEIGHTED GRAM-SCHMIDT METHOD FOR CONVEX QUADRATIC PROGRAMMING*

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, CA 94305, USA*

Received 16 September 1983
Revised manuscript received 12 January 1984

Range-space methods for convex quadratic programming improve in efficiency as the number of constraints active at the solution decreases. In this paper we describe a range-space method based upon updating a weighted Gram–Schmidt factorization of the constraints in the active set. The updating methods described are applicable to both primal and dual quadratic programming algorithms that use an active-set strategy.

Many quadratic programming problems include simple bounds on all the variables as well as general linear constraints. A feature of the proposed method is that it is able to exploit the structure of simple bound constraints. This allows the method to retain efficiency when the number of *general* constraints active at the solution is small. Furthermore, the efficiency of the method improves as the number of active bound constraints increases.

*Key words:* Convex Quadratic Programming, Range-Space Methods, Active-Set Methods, Updated Orthogonal Factorizations, Bound Constraints.

## 1. Introduction

The problem of concern in this paper is the convex quadratic programming (QP) problem with a mixture of bounds and general constraints:

$$\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad c^{\mathrm{T}}x+\tfrac{1}{2}x^{\mathrm{T}}Hx$$

$$\text{subject to} \quad l\leq\begin{Bmatrix} x \\ \mathscr{A}x \end{Bmatrix}\leq u,$$

where $c$ is a constant $n$-vector and $H$ is a constant $n\times n$ symmetric positive-definite matrix. The matrix $\mathscr{A}$ is $m\times n$, where $m$ may be zero. The constraints involving $\mathscr{A}$ will be called the *general* constraints; the remaining constraints will be called *simple bounds* or just bounds.

* 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.

** Present address: Department of Combinatorics and Optimization, University of Waterloo, Ontario, Canada.

Apart from the requirement of feasibility, the optimality conditions for QP involve only the constraints *active* (exactly satisfied) at the solution. 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, Murray and Wright, 1981). Let $x$ denote the current iterate, and $g(x)$ its gradient vector ($g(x) = c + Hx$); the $t$ rows of the matrix $C$ are defined as the coefficients of the constraints in the working set, and the vector $b$ is composed of the corresponding components of $l$ and $u$ (so that $Cx = b$). Note that $x$ satisfies the constraints in the working set *exactly*. The search direction $p$ is chosen so that $x + p$ is the solution of a quadratic programming subproblem with the original objective function, subject to the *equality* constraints of the working set. Let $\lambda$ denote the Lagrange multiplier vector of the subproblem. With this definition, $p$ and $\lambda$ are the solution of the system

$$\begin{pmatrix} H & -C^{\mathrm{T}} \\ C & 0 \end{pmatrix} \begin{pmatrix} p \\ \lambda \end{pmatrix} = \begin{pmatrix} -g \\ 0 \end{pmatrix}. \tag{1.1}$$

Having solved (1.1) at a given iteration, it is necessary at the next iteration to solve a neighboring system in which $C$, $x$ and $g$ are replaced by related quantities $\bar{C}$, $\bar{x}$ and $\bar{g}$. Usually, $\bar{C}$ is just $C$ with a single row either added or deleted, $\bar{x} = x + \alpha p$ for a nonnegative scalar $\alpha$, and $\bar{g} = g(\bar{x})$.

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 $C$, and the *null space* of vectors orthogonal to the rows of $C$. 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. By contrast, the methods of Dax (1981) and Gill et al. (1982) are range-space methods, and are most efficient when there are few constraints. (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, 1983.)

The method described in this paper is a *range-space* method. A feature of the method is that it is able to exploit the structure of simple bound constraints. This is important for many practical problems in which all but a few of the constraints are bounds, and many bounds are active at the solution. The method will retain the efficiency of a range-space approach when the number of *general* constraints active at the solution is small, as well as the advantages of a null-space method when the number of active bound constraints is *large*.

We shall discuss primarily details of how to compute $p$ and $\lambda$, and not the various strategies for altering the working set. The techniques described may be applied in the implementation of primal, dual and primal–dual quadratic programming algorithms that use an active-set strategy.

## 2. The weighted Gram–Schmidt (WGS) method

If $C$ and $H$ have full rank, $p$ and $\lambda$ satisfy the following *range-space equations*:

$$CH^{-1}C^{\mathrm{T}}\lambda = CH^{-1}g; \tag{2.1}$$

and

$$Hp = C^{\mathrm{T}}\lambda - g. \tag{2.2}$$

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

$$H = R^{\mathrm{T}}R \quad \text{and} \quad C = (L\ 0)Q^{\mathrm{T}}R, \tag{2.3}$$

where $R$ is the $n \times n$ Cholesky factor of $H$, $L$ is a $t \times t$ lower-triangular matrix, and $Q$ is an $n \times n$ orthogonal matrix.

The factorizations (2.3) provide a solution to the range-space equations (2.1) and (2.2) in the form:

$$L^{\mathrm{T}}\lambda = Y^{\mathrm{T}}R^{-\mathrm{T}}g, \tag{2.4}$$

$$Rp = -ZZ^{\mathrm{T}}R^{-\mathrm{T}}g, \tag{2.5}$$

where $Y$ and $Z$ are the $n \times t$ and $n \times (n-t)$ sections of the matrix $Q$, i.e.

$$Q = (Y\ Z).$$

A variant of (2.3)–(2.5) has been used by Goldfarb and Idnani (1983), who recur the matrix $Q^{\mathrm{T}}R^{-\mathrm{T}}$.

We now propose a method that uses equations similar to (2.3)–(2.5), in which we take advantage of the identity $ZZ^{\mathrm{T}} \equiv I - YY^{\mathrm{T}}$ in order to avoid storing $Z$. In place of the orthogonal factorization in (2.3), we utilize the *weighted Gram–Schmidt* (WGS) *factorization*

$$C = LY^{\mathrm{T}}R, \tag{2.6}$$

where $L$ is easily invertible but not necessarily lower triangular.

Given $R$, $L$ and $Y$, we define the three auxiliary vectors $u$, $v$ and $w$ by

$$R^{\mathrm{T}}u = g, \quad v = Y^{\mathrm{T}}u \quad \text{and} \quad w = Yv - u. \tag{2.7}$$

(Note that $Y^{\mathrm{T}}w = 0$.) Substitution into (2.4) and (2.5) allows $\lambda$ and $p$ to be defined from

$$L^{\mathrm{T}}\lambda = v \tag{2.8}$$

and

$$Rp = w. \tag{2.9}$$

At each iteration of an active-set method, a constraint is added to or deleted from the working set after a move of the form $\bar{x} = x + \alpha p$. (Note that if $p = 0$, more

than one constraint may be added or deleted at the same point.) These changes lead to *updates* of the factorizations $H = R^T R$, $C = LY^T R$. In practice, initial values for the vectors $u$, $v$ and $w$ are defined from (2.7) in terms of an initial feasible point and initial working set. Thereafter, the vectors $u$, $v$ and $w$ can be updated at negligible cost, as we show below. The principal computational effort per iteration lies in updating the factorization (2.6) as the working set changes, and in computing $p$ (and $\lambda$ when needed) from (2.8) and (2.9).

### 2.1. *Special form of the working set*

At a typical iteration of an active-set method applied to problem QP, the working set 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 may be considered as *free to vary* (and will be called simply 'free variables'). We use the suffices 'F' ('fixed') and 'V' ('varying') to denote items associated with the two types of variable.

We denote by $C$ the matrix whose $t$ rows are constraints in the current working set, and assume that $C$ contains $n_F$ bounds and $m_L$ general constraints (where 'L' denotes 'linear'), so that $t = n_F + m_L$. Let $A$ denote the matrix whose rows are the $m_L$ general constraints in the working set, and let $n_V$ denote the number of free variables ($n_V = n - n_F$). (If bounds are not treated separately, $n_F = 0$, $n_V = n$, and $m_L = t$.)

We assume that the variables are ordered so that the last $n_F$ variables are fixed, with all other relevant vectors and matrices ordered accordingly. In practice, the order of the variables is indicated by lists of indices, so there is no loss of generality in making this assumption. However, we shall see that this assumption has important implications for the update procedures.

The Hessian matrix $H$ is partitioned as

$$H = \begin{pmatrix} H_V & K \\ K^T & H_F \end{pmatrix}, \tag{2.10}$$

where $H_V$ and $H_F$ are $n_V \times n_V$ and $n_F \times n_F$ symmetric matrices. Similarly, the upper-triangular matrix $R$ (the Cholesky factor of the Hessian) may be partitioned as

$$R = \begin{pmatrix} R_V & S \\ 0 & R_F \end{pmatrix}, \tag{2.11}$$

where $R_V$ and $R_F$ are $n_V \times n_V$ and $n_F \times n_F$ upper-triangular matrices. (Note that $R_V$ is the Cholesky factor of $H_V$.)

The ordering of the variables assumed above means that the matrix of constraints in the working set can be written as

$$C = \begin{pmatrix} 0 & I_F \\ A & \end{pmatrix} = \begin{pmatrix} 0 & I_F \\ A_V & A_F \end{pmatrix}, \tag{2.12}$$

where $A_V$ is an $m_L \times n_V$ matrix, and $I_F$ denotes the $n_F$-dimensional identity matrix.

Assume that the Gram–Schmidt factorization of $A_V R_V^{-1}$ is known, i.e.,

$$A_V = L_V Y_V^T R_V, \tag{2.13}$$

where $L_V$ is an $m_L \times m_L$ lower-triangular matrix, and $Y_V$ is an $n_V \times m_L$ orthonormal matrix whose columns form a basis for the row space of $A_V R_V^{-1}$ (see Daniel et al., 1976, and Gill et al., 1982). The matrix $C$ (2.12) then has the factorization

$$C = LY^T R = \begin{pmatrix} 0 & N \\ L_V & M \end{pmatrix} \begin{pmatrix} Y_V^T & 0 \\ 0 & I_F \end{pmatrix} R, \tag{2.14}$$

where $N$ and $M$ are matrices of order $n_F \times n_F$ and $m_L \times n_F$ respectively. (We shall show that it is unnecessary to store the matrices $N$ and $M$.) Note that the matrix $L$ in (2.14) is *not* lower-triangular, but that the columns of the $n \times t$ orthonormal matrix

$$Y = \begin{pmatrix} Y_V & 0 \\ 0 & I_F \end{pmatrix}, \tag{2.15}$$

form a basis for the row space of $CR^{-1}$.

In the following, we show how (2.12) and (2.14) may be used to simplify the calculation of $p$ and $\lambda$ using the auxiliary vectors (2.7). The amount of work required for each computation will be given as the highest-order terms in the expression for the number of multiplications.

### 2.2. *Calculation of the search direction*

Let $g$ and $u$ be partitioned as

$$g = \begin{pmatrix} g_V \\ g_F \end{pmatrix} \quad \text{and} \quad u = \begin{pmatrix} u_V \\ u_F \end{pmatrix}. \tag{2.16}$$

The form (2.11) of $R$ implies that $g_V = R_V^T u_V$. The special form of $Y$ in (2.15) implies that the $t$-vector $v$ is given by

$$v = Y^T u = \begin{pmatrix} Y_V^T & 0 \\ 0 & I_F \end{pmatrix} \begin{pmatrix} u_V \\ u_F \end{pmatrix} = \begin{pmatrix} Y_V^T u_V \\ u_F \end{pmatrix} \equiv \begin{pmatrix} v_L \\ u_F \end{pmatrix} \begin{matrix} \}m_L \\ \}n_F \end{matrix}.$$

Similarly, the vector $w$ can be written as

$$w = Yv - u = \begin{pmatrix} Y_V & 0 \\ 0 & I_F \end{pmatrix} \begin{pmatrix} v_L \\ u_F \end{pmatrix} - \begin{pmatrix} u_V \\ u_F \end{pmatrix} = \begin{pmatrix} Y_V v_L - u_V \\ 0 \end{pmatrix} = \begin{pmatrix} w_V \\ 0 \end{pmatrix}. \tag{2.17}$$

(Note that $Y_V^T w_V = 0$.) The $n_F$ zero elements of the right-hand side of (2.17) and the upper-triangular form of $R$ imply that the search direction has the form $p = (p_V^T \ 0)^T$. This confirms that the elements of $p$ corresponding to the fixed variables

are zero. The vector $p_V$ is given by the solution of

$$R_V p_V = w_V,$$

and may be computed in order $\frac{1}{2}n_V^2$ multiplications.

### 2.3. Calculation of the Lagrange multipliers

When bounds are treated separately, the constraints in the working set are naturally partitioned into bound constraints and general constraints. Let $\lambda$ be partitioned into an $n_F$-vector $\lambda_F$ (the multipliers corresponding to the bound constraints) and an $m_L$-vector $\lambda_L$ (the multipliers corresponding to the general constraints). Substitution of (2.12) and the partitioned form of $\lambda$ into (1.1) gives, after rearrangement,

$$A_V^T \lambda_L = g_V + H_V p_V \tag{2.18}$$

and

$$\lambda_F = \bar{g}_F - A_F^T \lambda_L, \tag{2.19}$$

where $\bar{g}_F$ is the gradient with respect to the fixed variables at the point $x+p$.

We simplify (2.18) by using the factorization (2.13) of $A_V$ and the relations $H_V = R_V^T R_V$, $R_V^T u_V = g_V$, $R_V p_V = w_V$, and $Y_V v_L - u_V = w_V$ to obtain

$$Y_V L_V^T \lambda_L = u_V + w_V = Y_V v_L.$$

Hence, $\lambda_L$ is the solution of

$$L_V^T \lambda_L = v_L, \tag{2.20}$$

and may be computed in order $\frac{1}{2}m_L^2$ multiplications.

A significant difference that arises when bounds are treated separately is the need to compute multipliers specifically for the bound constraints (otherwise, $\lambda_L$ includes the multipliers for all the constraints). Therefore, we must consider how to compute $\lambda_F$ efficiently. Calculation of $\lambda_F$ from (2.19) requires $n_V m_L$ multiplications to form $A_F^T \lambda_L$, plus the work needed to obtain $\bar{g}_F$.

### 2.4. Storage options

Calculation of $\bar{g}_F$ from scratch involves a term of order $nn_F$, and hence would be very expensive when $n_F$ is large. Fortunately, this expense can be avoided using one of two storage options (the details are given in Section 5). With the first storage arrangement, the entire matrix $R$ (2.11) will be stored (recall that $R_V$ is a partition of $R$), and $R$ will be overwritten on $H$. When $R$ is available, $g_F$ may be *updated* using $n_F n_V$ multiplications. With the second storage option, the original matrix $H$ is stored in addition to $R_V$. In this case, an auxiliary vector is recurred so that $g_F$ may be computed when necessary, again at a cost of $n_F n_V$ multiplications.

In the next two sections, we describe the update procedures associated with performing an iteration of the WGS method. With either storage option, only the vectors $u_V$, $v_L$ and $w_V$ need be recurred. Barred quantities will denote those associated with the new working set. We have assumed the three-multiplication form of a plane rotation (see Gill et al., 1974).

## 3. Changes in the status of general constraints

### 3.1. Adding a general constraint to the working set

When a *general* constraint is added to the working set at the point $\bar{x} = x + \alpha p$, a new row is added to $A_V$. Thus, the row dimension of $A_V$, the column dimension of $Y_V$, and the dimension of $L_V$ in (2.13) will increase by one. In practice, the ordering of the general constraints is indicated by a list of indices, and the index of the new constraint is placed at the end of the list. Hence, we may assume without loss of generality that the row $a^T$ is added in the *last* position. In this case, $\bar{Y}_V$ is given by

$$\bar{Y}_V = (Y_V \;\; z). \tag{3.1}$$

The new column $z$ is a multiple of the vector $R_V^{-T}a$ orthogonalized with respect to the orthonormal set of columns of $Y_V$, i.e. $z = \tau(I - Y_V Y_V^T)q$, where $q$ satisfies $R_V^T q = a$ and $\tau$ is a normalizing factor. The matrix $\bar{L}_V$ is obtained by adding a new row to $L_V$. For complete details of the updating algorithm, including the use of reorthogonalization to ensure sufficient accuracy in $z$, the reader is referred to Daniel et al. (1976) and Gill et al. (1982).

The following theorem describes how the quantities $u_V$, $v_L$ and $w_V$ may be updated following the addition of a general constraint.

**Theorem 1.** *Let $p$ denote the vector that satisfies the range-space equation (2.2) at the point $x$. Let $\bar{x}$ ($\bar{x} = x + \alpha p$) be a point at which the row $a^T$ is added to $A_V$. Assume that the updated factors $\bar{L}_V$ and $\bar{Y}_V$ of $\bar{A}_V = \bar{L}_V \bar{Y}_V^T R_V$ have been computed, and that $z$, the new last column of $\bar{Y}_V$, is available. The vectors $u_V$, $v_L$ and $w_V$ are updated as follows:*

(i) $$\bar{u}_V = u_V + \alpha w_V; \tag{3.2}$$

(ii) $$\bar{v}_L = \begin{pmatrix} v_L \\ \nu \end{pmatrix}, \quad \text{where } \nu = (1-\alpha) z^T u_V; \tag{3.3}$$

(iii) $$\bar{w}_V = (1-\alpha) w_V + \nu z. \tag{3.4}$$

**Proof.** Using the relations, $\bar{g} = g + \alpha H p$, $R_V^T R_V = H_V$ and $R_V p_V = w_V$, we have

$$R_V^T \bar{u}_V = \bar{g}_V = g_V + \alpha H_V p_V = R_V^T u_V + \alpha R_V^T R_V p_V = R_V^T u_V + \alpha R_V^T w_V,$$

and (i) follows immediately.

To prove (ii), we use the definition of $\bar{v}_V$, (3.1) and (3.2) to give

$$\bar{v}_L = \bar{Y}_V^T \bar{u}_V = \begin{pmatrix} Y_V^T \\ z^T \end{pmatrix} \bar{u}_V = \begin{pmatrix} Y_V^T u_V + \alpha Y_V^T w_V \\ z^T u_V + \alpha z^T w_V \end{pmatrix}.$$

Since $Y_V^T w_V = 0$, $z^T w_V = -z^T u_V$ and $v_L = Y_V^T u_V$, this proves (ii). Similarly,

$$\bar{w}_V = \bar{Y}_V \bar{v}_L - \bar{u}_V = (Y_V \;\; z) \begin{pmatrix} v_L \\ \nu \end{pmatrix} - (u_V + \alpha w_V),$$

which reduces to (iii) after substituting $Y_V v_L - u_V = w_V$. □

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)–(iii) then simplify. If further constraints are added at the same point, Theorem 1 remains true with $\alpha = 0$.

The number of multiplications required to update $L_V$ and $Y_V$ following addition of a general constraint is of order $\frac{1}{2}n_V^2 + 2(k+1)n_V m_L$, where $k$ is the number of reorthogonalization steps (for well conditioned problems, $k$ is usually zero). The updates of $u_V$, $v_L$ and $w_V$ require negligible work.

### 3.2. *Deleting a general constraint from the working set*

When a general constraint is *deleted* from the working set at the point $x + \alpha p$, the row dimension of $A_V$, the column dimension of $Y_V$, and the dimension of $L_V$ are all decreased by one. In this case, the relationship between $Y_V$ and $\bar{Y}_V$ is given by

$$Y_V P_V = (\bar{Y}_V \;\; \bar{z}), \tag{3.5}$$

where $P_V$ is an orthonormal matrix representing a sequence of plane rotations (see Gill et al., 1982, for further details).

The following theorem indicates how the quantities $u_V$, $v_L$ and $w_V$ may be updated in this case.

**Theorem 2.** *Suppose that a constraint is deleted from the working set at the point* $\bar{x} = x + \alpha p$, *where* $p$ *satisfies* (2.2). *Assume that the updated factors* $\bar{L}_V$ *and* $\bar{Y}_V$ *of* $\bar{A}_V = \bar{L}_V \bar{Y}_V^T R_V$ *have been computed, so that the relationship between the old and new orthogonal factors is given by* (3.5). *Then*

(i) $$\bar{u}_V = u_V + \alpha w_V; \tag{3.6}$$

(ii) $$\begin{pmatrix} \bar{v}_L \\ \nu \end{pmatrix} = P_V^T v_L; \tag{3.7}$$

(iii) $$\bar{w}_V = (1-\alpha) w_V - \nu \bar{z}. \tag{3.8}$$

**Proof.** Result (i) follows as in Theorem 1. To prove (ii), note that, since $\bar{v}_L = \bar{Y}_V^T \bar{u}_V$,

we may write

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

where $\nu = \bar{z}^T \bar{u}_V$. Using (3.5) and (3.6) gives

$$\begin{pmatrix} \bar{v}_L \\ \nu \end{pmatrix} = P_V^T Y_V^T (u_V + \alpha w_V).$$

Since $Y_V^T w_V = 0$ and $Y_V^T u_V = v_L$, this gives the desired result.

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

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

Using (3.5), (3.6) and (3.7) gives

$$\bar{w}_V = Y_V P_V P_V^T v_L - u_V - \alpha w_V - \nu \bar{z}.$$

Since $P_V$ is orthonormal and $Y_V v_L - u_V = w_V$, this expression simplifies to (iii), 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 $\alpha = 0$, $p_V = 0$ and $w_V = 0$.

The number of multiplications required to update $L_V$ and $Y_V$ after deleting the $i$th constraint is of order $\frac{3}{2}(m_L - i)^2 + 3n_V(m_L - i)$; the updates of $u_V$, $v_L$ and $w_V$ require negligible extra work.

## 4. Changes in the status of bound constraints

When the status of a bound constraint changes, in general the variables must be *reordered* to maintain the convention given in Section 2.1. This leads to several differences from the update procedures given in Section 3, since reordering the variables alters the Hessian $H$ (and hence the Cholesky factor $R_V$).

### *4.1. Adding a bound constraint to the working set*

When a *bound* constraint is added to the working set, a previously free variable becomes fixed on its bound. Thus, the column dimension of $A_V$, the dimension of $R_V$, and the row dimension of $Y_V$ in (2.13) are decreased by one. The dimension of $L_V$ is unaltered.

In order to clarify the explanation of the update procedures, we shall first assume that the *last* free variable (variable $n_V$) is to be fixed. This corresponds to deleting the last column (say, $a$) of $A_V$, so that $A_V = (\bar{A}_V \ a)$. In this case, $\bar{R}_V$ is simply a

submatrix of $R_V$, i.e.

$$\begin{pmatrix} \bar{R}_V & r \\ 0 & \rho \end{pmatrix} = R_V. \tag{4.1}$$

The weighted Gram–Schmidt factorization (2.13) of $\bar{A}_V$ is computed as follows. First, note that for any vector $z$, it holds that

$$A_V = (\bar{A}_V \;\; a) = (L_V \;\; 0)(Y_V \;\; z)^T R_V. \tag{4.2}$$

We shall choose a special unit vector $z$ that is the result of orthogonalizing the $n_V$-th coordinate vector $e_V$ with respect to the columns of $Y_V$, i.e. $z = \tau(I - Y_V Y_V^T)e_V$ for some normalization factor $\tau$. (Note that $z$ is orthogonal to all the columns of $Y_V$.) Daniel et al. (1976) show that, if $Y_V$ is partitioned as

$$Y_V = \begin{pmatrix} \tilde{Y} \\ \tilde{y}^T \end{pmatrix},$$

then $z$ is of the form

$$z = \begin{pmatrix} -\tau \tilde{Y}\tilde{y} \\ 1/\tau \end{pmatrix}, \tag{4.3}$$

where $\tau = (1 - \tilde{y}^T\tilde{y})^{-1/2}$.

The crucial property of the vector (4.3) is that a sequence of plane rotations that transforms the last *column* of $(Y_V \;\; z)$ to $e_V$ will simultaneously produce $e_V^T$ as its last *row*. Hence, if $P$ denotes an $(m_L+1)$-dimensional orthonormal matrix that represents an appropriate sequence of plane rotations, we have

$$(Y_V \;\; z)P = \begin{pmatrix} \bar{Y}_V & 0 \\ 0 & 1 \end{pmatrix}. \tag{4.4}$$

Substituting (4.1) and (4.4) into (4.2), and using the orthogonality of $P$ gives

$$(\bar{A}_V \;\; a) = (L_V \;\; 0)PP^T(Y_V \;\; z)^T \begin{pmatrix} \bar{R}_V & r \\ 0 & \rho \end{pmatrix} = (L_V \;\; 0)P \begin{pmatrix} \bar{Y}_V^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{R}_V & r \\ 0 & \rho \end{pmatrix}.$$

The rotations represented by $P$ take linear combinations of the columns of $(Y_V \;\; z)$ in the order $(m_L, m_L+1), (m_L-1, m_L+1), \ldots, (1, m_L+1)$. Thus, $P$ does not alter the lower-triangular structure of the first $m_L$ columns of $L_V$, and we have

$$(L_V \;\; 0)P = (\bar{L}_V \;\; v),$$

where $\bar{L}_V$ is lower-triangular ($v$ is a reconstituted version of the column of $A_V R_V^{-1}$ that is being deleted). Clearly

$$\bar{A}_V = \bar{L}_V \bar{Y}_V^T \bar{R}_V,$$

as required.

Now we turn to the general case in which the $j$-th variable ($j \leq n_V$) is to be fixed on a bound. Because of the ordering convention defined in Section 2, the variables must be reordered so that the first $n_V - 1$ variables will be free during the next iteration. This is accomplished formally through a permutation matrix that reorders the variables so that variable $j$ is in position $n_V$ (note that the last $n_F$ variables are not reordered, and hence the permutation matrix affects only components 1 through $n_V$).

Let $\Pi$ denote a suitable $n_V$-dimensional permutation matrix, such that the reordered Hessian with respect to the first $n_V$ variables is $\Pi^T H_V \Pi$. The Cholesky factor $\hat{R}_V$ of $\Pi^T H_V \Pi$ is given by

$$\hat{R}_V = Q R_V \Pi,$$

where the $n_V \times n_V$ orthonormal matrix $Q$ represents a sequence of plane rotations that make $R_V \Pi$ upper triangular. To verify that $\hat{R}_V$ is indeed the Cholesky factor, observe that

$$\Pi^T H_V \Pi = \Pi^T R_V^T R_V \Pi = \Pi^T R_V^T Q^T Q R_V \Pi = \hat{R}_V^T \hat{R}_V.$$

The Hessian $\bar{H}_V$ with respect to the new (smaller by one) set of free variables is $\Pi^T H_V \Pi$ with its last row and column deleted, i.e.

$$\Pi^T H_V \Pi = \begin{pmatrix} \bar{H}_V & h^T \\ h & \eta \end{pmatrix}.$$

This implies that the $(n_V - 1)$-dimensional matrix $\bar{R}_V$ satisfies

$$\begin{pmatrix} \bar{R}_V & r \\ 0 & \rho \end{pmatrix} = \hat{R}_V = Q R_V \Pi. \tag{4.5}$$

The number of multiplications required to compute $\bar{R}_V$ is of order $\frac{3}{2}(n_V - j)^2$. (When all of $R$ (2.11) is stored, a further $3n_F(n_V - j)$ multiplications are required to apply the plane rotations in $Q$ to the rows of $S$.)

When $\bar{R}_V$ is defined by (4.5), $\bar{Y}_V$ and $\bar{L}_V$ may be obtained by a generalization of the procedure described at the beginning of the section. The major difference in the results is that the relationship between $Y_V$ and $\bar{Y}_V$ changes from (4.4) to

$$Q(Y_V \;\; z)P = \begin{pmatrix} \bar{Y}_V & 0 \\ 0 & 1 \end{pmatrix}. \tag{4.6}$$

The number of multiplications required to update $L_V$ and $Y_V$ when the $j$-th variable is fixed on a bound is of order $4n_V m_L + 3m_L(n_V - j) + \frac{3}{2}m_L^2$.

The following theorem indicates how to update the vectors $u_V$, $v_L$ and $w_V$ following addition of a bound to the working set.

**Theorem 3.** *Let the j-th free variable be fixed on a bound at the point $x + \alpha p$. Assume that the updated factors $\bar{R}_V$, $\bar{L}_V$, and $\bar{Y}_V$ of $\bar{A}_V = \bar{L}_V \bar{Y}_V^T \bar{R}_V$ have been computed, and that z, the vector defined in* (4.3), *is available. The quantities $u_V$, $v_L$ and $w_V$ are*

*updated as follows:*

$$\text{(i)} \qquad \begin{pmatrix} \bar{u}_V \\ \omega \end{pmatrix} = Q(u_V + \alpha w_V); \tag{4.7}$$

$$\text{(ii)} \qquad \begin{pmatrix} \bar{v}_L \\ \omega \end{pmatrix} = P^T \begin{pmatrix} v_L \\ \nu \end{pmatrix}, \text{ with } \nu = (1-\alpha) z^T u_V; \tag{4.8}$$

$$\text{(iii)} \qquad \begin{pmatrix} \bar{w}_V \\ 0 \end{pmatrix} = Q((1-\alpha) w_V + \nu z). \tag{4.9}$$

**Proof.** To prove (i), observe that $\tilde{g}_V$, the (reordered) gradient with respect to the *old* set of free variables at the new point is given by

$$\tilde{g}_V = \begin{pmatrix} \bar{g}_V \\ \gamma \end{pmatrix} = \Pi^T (g_V + \alpha H_V p_V),$$

where $\gamma$ is the component of the gradient with respect to the variable to be fixed on a bound (the value of $\gamma$ is not needed to perform the updates). Using the relations $R_V^T u_V = g_V$, $H_V = R_V^T R_V$, and $R_V p_V = w_V$, we obtain

$$\begin{pmatrix} \bar{g}_V \\ \gamma \end{pmatrix} = \Pi^T R_V^T (u_V + \alpha w_V). \tag{4.10}$$

Since $\bar{R}_V^T \bar{u}_V = \bar{g}_V$, we may write

$$\begin{pmatrix} \bar{R}_V^T & 0 \\ r^T & \rho \end{pmatrix} \begin{pmatrix} \bar{u}_V \\ \omega \end{pmatrix} = \begin{pmatrix} \bar{g}_V \\ \gamma \end{pmatrix}.$$

Substituting from (4.5) and (4.10), we have

$$\Pi^T R_V^T Q^T \begin{pmatrix} \bar{u}_V \\ \omega \end{pmatrix} = \Pi^T R_V^T (u_V + \alpha w_V)$$

and therefore

$$Q^T \begin{pmatrix} \bar{u}_V \\ \omega \end{pmatrix} = u_V + \alpha w_V. \tag{4.11}$$

Since $Q$ is orthonormal, (i) follows from (4.11).

In order to prove (ii), we begin with the definition $\bar{v}_L = \bar{Y}_V^T \bar{u}_V$, and note that

$$\begin{pmatrix} \bar{v}_L \\ \omega \end{pmatrix} = \begin{pmatrix} \bar{Y}_V^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{u}_V \\ \omega \end{pmatrix}.$$

Substituting from (4.6) and (4.11) gives

$$\begin{pmatrix} \bar{v}_L \\ \omega \end{pmatrix} = P^T \begin{pmatrix} Y_V^T \\ z^T \end{pmatrix} Q^T \begin{pmatrix} \bar{u}_V \\ \omega \end{pmatrix} = P^T \begin{pmatrix} Y_V^T \\ z^T \end{pmatrix} (u_V + \alpha w_V) = P^T \begin{pmatrix} v_L \\ \nu \end{pmatrix},$$

where the last expression was obtained using the relations $Y_V^T u_V = v_L$, $Y_V^T w_V = 0$, $w_V = Y_V v_L - u_V$ and $Y_V^T z = 0$.

Finally, since $\bar{w}_V = \bar{Y}_V \bar{v}_L - \bar{u}_V$, it also holds that

$$\begin{pmatrix} \bar{w}_V \\ 0 \end{pmatrix} = \begin{pmatrix} \bar{Y}_V & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{v}_L \\ \omega \end{pmatrix} - \begin{pmatrix} \bar{u}_V \\ \omega \end{pmatrix}$$

$$= Q(Y_V \;\; z) \begin{pmatrix} v_L \\ \nu \end{pmatrix} - Q(u_V + \alpha w_V) = Q(Y_V v_L + \nu z - u_V - \alpha w_V),$$

using (4.6), (4.7) and (4.8). Since $Y_V v_L - u_V = w_V$, we have the desired result (4.9). □

Comparison of (3.2)–(3.4) with (4.7)–(4.9) shows that the updates to $u_V$, $v_L$ and $w_V$ associated with adding a bound to the working set are very similar to those needed when adding a general constraint; the difference is that further plane rotations must be applied to certain vectors. This means that the updates can be implemented with very little additional programming complexity.

### 4.2. *Deleting a bound constraint from the working set*

When a bound constraint is *deleted* from the working set, a previously fixed variable becomes free. In this case, the column dimension of $A_V$, the row and column dimensions of $R_V$ in (2.13), and the row dimension of $Y_V$ are increased by one; and the dimension of $L_V$ remains unaltered.

In order to maintain the convention defined in Section 2, the new free variable will become variable $n_V + 1$. Thus, the Hessian $\bar{H}_V$ with respect to the new set of free variables and its Cholesky factor $\bar{R}_V$ will be such that

$$\bar{H}_V = \begin{pmatrix} H_V & h \\ h^T & \eta \end{pmatrix} \quad \text{and} \quad \bar{R}_V = \begin{pmatrix} R_V & r \\ 0 & \rho \end{pmatrix}. \tag{4.12}$$

When $R$ is available, $r$ is obtained from the update of $S$ and $R_F$. Assuming that variable $n_V + j$ is released from its bound, this update requires of order $\frac{3}{2}j(j-1)$ multiplications. When $H$ is available, $r$ is computed (after reordering) from one further step of the column-wise Cholesky factorization of $H_V$, which requires of order $\frac{1}{2}n_V^2$ multiplications.

Deletion of a bound from the working set as described above adds a new column at the end of $A_V$; let $a$ denote the new column of $\bar{A}_V$. From (2.13), the augmented matrix $\bar{A}_V$ may be written as

$$\bar{A}_V = (A_V \;\; a) = (L_V \;\; v) \begin{pmatrix} Y_V^T & 0 \\ 0 & 1 \end{pmatrix} \bar{R}_V,$$

with $v = \bar{A}_V q$. The vector $q$ is the solution of the triangular system $\bar{R}_V q = e_{V+1}$, where $e_{V+1}$ denotes the $(n_V + 1)$-th coordinate vector.

The matrix $(L_V \;\; v)$ is lower-triangular except for the 'vertical spike' $v$, and may be reduced to lower-triangular form by a sequence of plane rotations in the planes $(1, m_L + 1)$, $(2, m_L + 1), \ldots, (m_L, m_L + 1)$. If we let $P$ denote the $(m_L + 1)$-dimens-

ional orthonormal matrix of plane rotations, we may write

$$\bar{A}_V = (L_V \;\; v)PP^T\begin{pmatrix} Y_V^T & 0 \\ 0 & 1 \end{pmatrix}\bar{R}_V = (\bar{L}_V \;\; 0)P^T\begin{pmatrix} Y_V^T & 0 \\ 0 & 1 \end{pmatrix}\bar{R}_V.$$

The effect of the application of $P$ on the columns of the augmented orthogonal factor is to fill in the zero elements of the last row and column. Thus

$$\begin{pmatrix} Y_V & 0 \\ 0 & 1 \end{pmatrix}P = (\bar{Y}_V \;\; \bar{z}). \tag{4.13}$$

The matrix $\bar{Y}_V$ is the orthogonal factor associated with $\bar{A}_V$ and $\bar{R}_V$, and the vector $\bar{z}$ lies in the null space of $\bar{A}_V\bar{R}_V^{-1}$. The updates of $\bar{L}_V$ and $\bar{Y}_V$ following deletion of a bound from the working set require of order $\frac{1}{2}n_V^2 + 4n_V m_L + \frac{3}{2}m_L^2$ multiplications.

Following the changes described above, the updates to the vectors $v_V$, $v_L$ and $w_V$ are given in the following theorem.

**Theorem 4.** *Assume that a bound is to be deleted from the working set at the point* $\bar{x} = x + \alpha p$, *and that the updated factors* $\bar{R}_V$, $\bar{L}_V$ *and* $\bar{Y}_V$ *of* $\bar{A}_V = \bar{L}_V\bar{Y}_V^T\bar{R}_V$ *have been computed. Then*

(i) $$\bar{u}_V = \begin{pmatrix} u_V + \alpha w_V \\ \mu \end{pmatrix}, \tag{4.14}$$

*where* $\mu$ *can be calculated from* $u_V$, $g_F$ *and the elements of* $\bar{R}_V$;

(ii) $$\begin{pmatrix} \bar{v}_L \\ \nu \end{pmatrix} = P^T\begin{pmatrix} v_L \\ \mu \end{pmatrix}; \tag{4.15}$$

(iii) $$\bar{w}_V = \begin{pmatrix} (1-\alpha)w_V \\ 0 \end{pmatrix} - \nu\bar{z}.$$

**Proof.** When a bound has been deleted from the working set, we have

$$\bar{g}_V = \begin{pmatrix} \tilde{g}_V \\ \gamma \end{pmatrix},$$

where $\tilde{g}_V$ is the new gradient vector with respect to the old (smaller) set of free variables, and $\gamma$ is the component of the gradient with respect to the newly freed variable. With the first storage option, $\gamma$ will be available after the update of $g_F$ (see Section 5.2); with the second storage option, $\gamma$ may be computed directly from $H$.

It follows from the definition of the quadratic function, the form (2.10) of $H$ and the form of $p$ that

$$\tilde{g}_V = g_V + \alpha H_V p_V.$$

Using the relations $g_V = R_V^T u_V$, $H_V = R_V^T R_V$, and $R_V p_V = w_V$, this expression

becomes

$$\tilde{g}_V = R_V^T(u_V + \alpha w_V). \tag{4.16}$$

Let us partition the $(n_V+1)$-vector $\bar{u}_V$ as

$$\bar{u}_V = \begin{pmatrix} \tilde{u}_V \\ \mu \end{pmatrix},$$

where $\tilde{u}_V$ is an $n_V$-vector. Then, from the expression (4.12) for $\bar{R}_V$, we have

$$\begin{pmatrix} R_V^T & 0 \\ r^T & \rho \end{pmatrix}\begin{pmatrix} \tilde{u}_V \\ \mu \end{pmatrix} = \begin{pmatrix} \tilde{g}_V \\ \gamma \end{pmatrix}. \tag{4.17}$$

The expression (4.16) and the first $n_V$ equations in (4.17) give

$$R_V^T\tilde{u}_V = \tilde{g}_V = R_V^T(u_V + \alpha w_V),$$

from which it follows that $\tilde{u}_V = u_V + \alpha w_V$. The scalar $\mu$ may then be obtained from the last equation in (4.17), i.e.

$$\mu = \frac{\gamma - r^T\tilde{u}_V}{\rho}.$$

This proves (i).

To prove (ii), we use the definition $\bar{v}_L = \bar{Y}_V^T\bar{u}_V$ to write

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

where $\nu = \bar{z}^T\bar{u}_V$. Substituting from (4.13) and (4.14) gives

$$\begin{pmatrix} \bar{v}_L \\ \nu \end{pmatrix} = P^T\begin{pmatrix} Y_V^T & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} u_V + \alpha w_V \\ \mu \end{pmatrix}.$$

The desired result then follows from the relations $Y_V^T u_V = v_L$ and $Y_V^T w_V = 0$.

Finally, by definition, $\bar{w}_V = \bar{Y}_V\bar{v}_L - \bar{u}_V$, and hence

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

Substituting from (4.13), (4.14) and (4.15) and using the orthogonality of $P$, we obtain

$$\begin{aligned} \bar{w}_V &= \begin{pmatrix} Y_V & 0 \\ 0 & 1 \end{pmatrix}PP^T\begin{pmatrix} v_L \\ \eta \end{pmatrix} - \begin{pmatrix} u_V + \alpha w_V \\ \nu \end{pmatrix} - \nu\bar{z} = \begin{pmatrix} Y_V v_L - u_V - \alpha w_V \\ 0 \end{pmatrix} - \nu\bar{z} \\ &= \begin{pmatrix} (1-\alpha)w_V \\ 0 \end{pmatrix} - \nu\bar{z}, \end{aligned}$$

which is the desired result. □

One of the most important implications of Theorems 1–4 is that the only difference between a change in working set involving a *bound* constraint and a change involving a *general* constraint is that some of the relevant vectors must be transformed by an additional sweep of rotations.

## 5. Computing the multipliers corresponding to the fixed variables

In order to delete a bound constraint from the working set, the vector $\lambda_F$ must be computed. As noted in Section 2.3, $\lambda_F$ is defined by (2.19), which involves the two terms $\bar{g}_F$ and $A_F^T\lambda_L$. The second term is obtained by solving (2.20) for $\lambda_L$; forming $A_F^T\lambda_L$ then requires $n_F m_L$ multiplications. We now discuss how to obtain $\bar{g}_F$ with the two available storage options.

### 5.1. Change in the gradient after a change in x

The change in the gradient may be viewed as two separate parts, corresponding respectively to the move from $x$ to $\bar{x}$ and to the change in the working set. The first change is independent of the type of constraint to be added or deleted; the second change is just a reordering.

From the definition of the quadratic function, the gradient $\tilde{g}$ at the point $\bar{x} = x + \alpha p$ is given by

$$\tilde{g} = g + \alpha Hp. \tag{5.1}$$

With the first storage option, the Cholesky factor $R$ (2.11) of $H$ is available. We may therefore substitute (2.11) and the definition $Rp = w$ from (2.1) into (5.1), giving

$$\tilde{g} = g + \alpha R^T w. \tag{5.2}$$

The forms (2.11) of $R$ and (2.17) of $w$ imply that

$$R^T w = \begin{pmatrix} R_V^T w_V \\ S^T w_V \end{pmatrix}. \tag{5.3}$$

Thus, using (5.2), (5.3) and the partitioned form (2.16) of $g$, we have

$$\tilde{g}_V = g_V + \alpha R_V^T w_V \tag{5.4}$$

and

$$\tilde{g}_F = g_F + \alpha S^T w_V. \tag{5.5}$$

Equation (5.5) shows that $\tilde{g}_F$ may be *updated* with $n_F n_V$ multiplications when all of $R$ is stored.

With the second storage option, all of $H$ is available. In this case, it follows from (2.10) and the partitioned form (2.16) of $g$ that $\tilde{g}_F$ may be written as

$$\begin{aligned}\tilde{g}_F &= H_F\bar{x}_F + K^T\bar{x}_V + c_F \\ &= \bar{m} + K^T\bar{x}_V, \end{aligned} \tag{5.6}$$

where $\bar{m}$ denotes the vector $H_F\bar{x}_F + c_F$. Thus, for the second storage option, the vector $m$ ($m = H_Fx_F + c_F$) is recurred, and $\tilde{g}_F$ may be computed *when necessary* from (5.6), at a cost of $n_Fn_V$ multiplications.

### 5.2. Change in status of a general constraint

When a general constraint changes status, the ordering of the variables is not altered, and hence $\bar{g}_F = \tilde{g}_F$. Thus, the update (5.5) may be used to obtain $\bar{g}_F$ with the first storage option. The change in status of a general constraint does not alter $m$.

### 5.3. Change in status of a bound constraint

Following the change in status of a bound constraint, the variables are reordered as described in Sections 4.1 and 4.2. The reordering is expressed formally through a suitable permutation matrix $\Pi$. The gradient is also reordered using $\Pi$; thus, at $x + \alpha p$ we have

$$\bar{g} = \Pi^T\tilde{g} = \Pi^T(g + \alpha Hp).$$

*Storage Option 1.* The update to $\bar{g}_F$ depends on whether a bound constraint is added or deleted. When the $j$-th bound variable is *added* to the working set, a scalar $\gamma$ (the component of the gradient with respect to the newly fixed variable) is added at the front of $\tilde{g}_F$ to give

$$\bar{g}_F = \begin{pmatrix} \gamma \\ \tilde{g}_F \end{pmatrix}.$$

The value of $\gamma$ is one of the components of $\tilde{g}_V$; since $g_V$ is not updated, $\gamma$ may be computed using $R$ as follows. The relationship $g_V = R_V^Tu_V$ and (5.4) imply that

$$\tilde{g}_V = R_V^T(u_V + \alpha w_V). \tag{5.7}$$

Since $\gamma$ is the $j$-th component of $\tilde{g}_V$, it can be computed using (5.7) by multiplying the $j$-th row of $R_V^T$ by the vector $u_V + \alpha w_V$ before any updates are performed.

When a bound is *deleted* from the working set, (5.5) gives the updated gradient with respect to the old set of fixed variables. The reordering in this case simply removes the component of $\tilde{g}_F$ corresponding to the variable to be freed. This value, $\gamma$, is then used to update $u_V$ (see Theorem 4), and the remaining $n_F - 1$ elements of $\tilde{g}_F$ form $\bar{g}_F$.

In either case, updating $g_F$ involves negligible work beyond that required to obtain $\tilde{g}_F$ from (5.5).

*Storage Option 2.* With the second storage option, $R_V$ and $H$ are available. In this case, $g_F$ is computed from (5.6), where the vector $m$ is updated ($m = H_F x_F + c_F$).

When a bound is *added* to the working set, a new component is added at the front of $x_F$, and we have

$$\bar{x}_F = \begin{pmatrix} \xi \\ x_F \end{pmatrix}, \quad \bar{c}_F = \begin{pmatrix} \zeta \\ c_F \end{pmatrix} \quad \text{and} \quad \bar{H}_F = \begin{pmatrix} \eta & h^T \\ h & H_F \end{pmatrix}.$$

Thus, $\bar{m}$ may be written as

$$\bar{m} = \bar{H}_F \bar{x}_F + \bar{c}_F = \begin{pmatrix} \eta & h^T \\ h & H_F \end{pmatrix} \begin{pmatrix} \xi \\ x_F \end{pmatrix} + \begin{pmatrix} \zeta \\ c_F \end{pmatrix} = \begin{pmatrix} \eta\xi + h^T x_F + \zeta \\ m + \xi h \end{pmatrix}. \tag{5.8}$$

The formula (5.8) gives the update for $m$.

When *deleting* a bound, the reordering of the last $n_F$ components of $x$ is defined by a permutation matrix $\Pi$ such that

$$\Pi^T x_F = \begin{pmatrix} \xi \\ \bar{x}_F \end{pmatrix} \quad \text{and} \quad \Pi^T c_F = \begin{pmatrix} \zeta \\ \bar{c}_F \end{pmatrix}.$$

The Hessian $\bar{H}_F$ with respect to the reduced set of free variables is given by

$$\Pi^T H_F \Pi = \begin{pmatrix} \eta & h^T \\ h & \bar{H}_F \end{pmatrix}.$$

Applying the permutation to $m$ gives

$$\Pi^T m = \Pi^T (H_F x_F + c_F) = \Pi^T H_F \Pi \Pi^T x_F + \Pi^T c_F = \begin{pmatrix} \eta & h^T \\ h & \bar{H}_F \end{pmatrix} \begin{pmatrix} \xi \\ \bar{x}_F \end{pmatrix} + \begin{pmatrix} \zeta \\ \bar{c}_F \end{pmatrix}$$

$$= \xi \begin{pmatrix} \eta \\ h \end{pmatrix} + \begin{pmatrix} h^T \bar{x}_F + \zeta \\ \bar{H}_F \bar{x}_F + \bar{c}_F \end{pmatrix}.$$

Thus, we may update $\bar{m}$ from

$$\begin{pmatrix} \mu \\ \bar{m} \end{pmatrix} = \Pi^T m - \xi \begin{pmatrix} \eta \\ h \end{pmatrix}.$$

## 6. Summary and discussion

In Table 1, we summarize the number of multiplications required to perform the calculations associated with an iteration of the WGS method, for both storage options. The word 'Same' in the column for the second storage option means that the procedure requires the same number of multiplications as with the first storage option; the entry '—' in a column means that the given procedure is not executed with that storage option.

Table 1
Summary of calculations in the WGS method

| Computation | | Storage option 1 | Storage option 2 |
|---|---|---|---|
| Compute $p_V$ | | $\frac{1}{2}n_V^2$ | Same |
| Compute $\lambda_L$ | | $\frac{1}{2}m_L^2$ | Same |
| Update $g_F$ | | $n_F n_V$ | — |
| Compute $\lambda_F$ | | $n_F m_L$ | $n_F(n_V + m_L)$ |
| Update $L_V$ and $Y_V$ | Add general | $\frac{1}{2}n_V^2 + 2n_V m_L$ | Same |
| | Add bound $j$ | $4n_V m_L + 3m_L(n_V - j) + \frac{3}{2}m_L^2$ | Same |
| | Delete $i$-th general | $3n_V(m_L - i) + \frac{3}{2}(m_L - i)^2$ | Same |
| | Delete bound | $\frac{1}{2}n_V^2 + 4n_V m_L + \frac{3}{2}m_L^2$ | Same |
| Update $R$ | Add bound $j$ | $\frac{3}{2}(n_V - j)^2 + 3n_F(n_V - j)$ | — |
| | Delete bound $n_V + j$ | $\frac{3}{2}j(j-1)$ | — |
| Update $R_V$ | Add bound $j$ | — | $\frac{3}{2}(n_V - j)^2$ |
| | Delete bound | — | $\frac{1}{2}n_V^2$ |

Note that substantial savings in work are achieved by taking advantage of bounds as $n_F$ increases ($n_V$ decreases). (We have assumed that no reorthogonalization is required when adding a general constraint to the working set.)

Because of the extra work needed to compute the multipliers for bound constraints, it is recommended for primal methods that bound constraints be considered for deletion only when no general constraint is suitable for deletion. With this strategy, the multipliers for the bound constraints need not be computed until they are required. (However, $g_F$ must be updated at every iteration with the first storage option.)

With the first storage option, the major storage requirements for dense problems are $\frac{1}{2}n^2$ elements for the Cholesky factor $R$ (since it is assumed that the Cholesky factors of $H$ are stored in place of $H$ itself, this storage is necessary to store the definition of the problem), and $\frac{1}{2}\bar{m}_L^2 + \bar{m}_L \bar{n}_V$ elements for the matrices $L_V$ and $Y_V$, where $\bar{m}_L$ denotes the maximum number of general constraints in the working set, and $\bar{n}_V$ denotes the maximum number of free variables at any iteration.

The first storage option is particularly useful when the Cholesky factors of $H$ are available rather than $H$ itself. For instance, the QP problem may be a subproblem within a nonlinear programming algorithm that performs quasi-Newton updates to the Hessian $H$ of the Lagrangian function (see, e.g., Schittkowski, 1982). The updates are often expressed in terms of the Cholesky factor $R$ of the Hessian. In order to begin the method, the variables must be ordered as described in Section 2. In general, this could involve several modifications to $R$ to reflect the reordering of its columns. However, the expectation would be that the set of free variables at the solution of one subproblem will eventually be the same as for the next.

In the dense case, the storage requirements for the second option include an additional $\frac{1}{2}\bar{n}_V^2$ locations to store $R_V$. (We assume that storing $H$ is equivalent to storing $R$.) The second option would have an advantage in storage for problems in

which $H$ is sparse, but substantial fill-in occurs when computing the Cholesky factor $R$. In this case, the storage required for $H$ and $R_V$ might be significantly less than that required to store all of $R$.

## Acknowledgements

The authors would like to thank the referee and Associate Editor for a number of helpful suggestions.

## References

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

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

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

A. Dax, "An active set algorithm for convex quadratic programming", Technical Report, Hydrological Service (Jerusalem, Israel, 1981).

R. Fletcher, "A general quadratic programming algorithm", *Journal of the Institute of Mathematics and its Applications* 7 (1971) 76–91.

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

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

P.E. Gill, N.I.M. Gould, W. Murray, M.A. Saunders and M.H. Wright, "Range-space methods for convex quadratic programming", Technical Report SOL 82-14, Department of Operations Research, Stanford University (Stanford, CA, 1982).

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

D. Goldfarb, "Matrix factorizations in optimization of nonlinear functions subject to linear constraints", *Mathematical Programming* 10 (1976) 1–31.

D. Goldfarb and A. Idnani, "A numerically stable dual method for solving strictly convex quadratic programs", *Mathematical Programming* 27 (1983) 1–33.

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

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

K. Schittkowski, "On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function", Report SOL 82-4, Department of Operations Research, Stanford University (Stanford, CA, 1982).