

Stable Reduced Hessian Updates for Indefinite Quadratic Programming ¹

Roger Fletcher

Department of Mathematics, University of Dundee, Dundee DD1 4HN, Scotland, UK.

Numerical Analysis Report NA/187, January 1999

Abstract

Stable techniques are considered for updating the reduced Hessian matrix that arises in a null-space active set method for Quadratic Programming when the Hessian matrix itself may be indefinite. A scheme for defining and updating the null-space basis matrix is described which is adequately stable and allows advantage to be taken of sparsity. A new canonical form for the reduced Hessian matrix is proposed that can be updated in a numerically stable way. Some consequences for the choice of minor iteration search direction are described.

Keywords Quadratic Programming, Active Set Method, Null-space method, Reduced Hessian matrix, Numerical stability.

1 Introduction

This paper documents recent progress in the development of a Quadratic Programming (QP) solver called `bqpd` that is based on a null-space active set method. In the first production version of the code, numerical instability could occasionally occur due to the way in which the null-space basis matrix Z was chosen, and the way in which the reduced Hessian matrix was updated. In a revised version of `bqpd`, some new ideas for handling these situations have been implemented, and are described here. Practical experience on many thousands of QP problems arising from the use of `bqpd` in an SQP method for nonlinear programming now shows no evidence of numerical instability in the reduced Hessian updates.

For the purposes of this paper, a simplified form of the QP problem will suffice, namely

$$\begin{aligned} & \text{minimize} && \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{c}^T\mathbf{x} && \mathbf{x} \in \mathbb{R}^n \\ & \text{subject to} && \mathbf{a}_i^T\mathbf{x} \geq b_i && i = 1, 2, \dots, m, \end{aligned} \tag{1.1}$$

in which the (symmetric) Hessian matrix G may be indefinite (in the sense of not being required to be positive definite). In such cases, a null-space primal active set method is

¹This paper is dedicated to Dr. William C. Davidon on the occasion of his 70th birthday

the preferred way to find a Kuhn–Tucker (KT) point, which will almost always be a local solution. (Guaranteeing to finding a global solution is a much more difficult task.) It is assumed that the reader is familiar with these concepts, which are described in more detail in, for example, Fletcher [1]. The description of a primal active set method in Section 10.3 of [1] is largely followed. An index set \mathcal{A} , referred to as the *active set*, is kept and may be regarded as the current estimate of the active constraints at the solution to (1.1). The current point $\mathbf{x}^{(k)}$ is always a feasible point at which the constraints in the active set are satisfied as equations. Each iteration of the method tries to locate the solution to an *equality problem* (EP) in which the constraints in \mathcal{A} are treated as equations, and the remaining constraints are disregarded. This is most conveniently done by shifting the origin to $\mathbf{x}^{(k)}$ and (where possible) solving the EP

$$\begin{aligned} & \text{minimize} && \frac{1}{2}\mathbf{s}^T G\mathbf{s} + \mathbf{s}^T \mathbf{g}^{(k)} && \mathbf{s} \in \mathbb{R}^n \\ & \text{subject to} && A^T \mathbf{s} = \mathbf{0}, \end{aligned} \tag{1.2}$$

in which $\mathbf{g}^{(k)} = G\mathbf{x}^{(k)} + \mathbf{c}$ is the gradient of the quadratic function at $\mathbf{x}^{(k)}$, and A is the matrix whose columns are the vectors \mathbf{a}_i , $i \in \mathcal{A}$. The next iterate $\mathbf{x}^{(k+1)}$ is then found by a line search for the best feasible point in the direction \mathbf{s} . If $\mathbf{x}^{(k)}$ is the feasible solution of the current equality problem (an FSEP), then a vector of multipliers $\boldsymbol{\lambda}$ is calculated, defined by $A\boldsymbol{\lambda} = \mathbf{g}^{(k)}$. If $\boldsymbol{\lambda} \geq \mathbf{0}$ then $\mathbf{x}^{(k)}$ solves (1.1). Otherwise some $\lambda_p < 0$ is selected, p is removed from \mathcal{A} , and the method continues as above.

The main linear algebra task in this method is the calculation of the solution and multipliers for the EP (1.2). In a null–space space method, a matrix Z is (implicitly) calculated, whose columns are a basis for the null–space of the matrix A^T . Then the feasible region $A^T \mathbf{s} = \mathbf{0}$ of (1.2) can be parametrized by $\mathbf{s} = Z\mathbf{u}$, and (1.2) reduces to the unconstrained minimization over \mathbf{u} of the quadratic function

$$\frac{1}{2}\mathbf{u}^T M\mathbf{u} + \mathbf{u}^T Z^T \mathbf{g}^{(k)} \tag{1.3}$$

in which

$$M = Z^T G Z \tag{1.4}$$

is referred to as the *reduced Hessian matrix*, and $Z^T \mathbf{g}^{(k)}$ as the *reduced gradient*. If M is positive definite then this may be achieved by solving the system

$$M\mathbf{u} = -Z^T \mathbf{g}^{(k)}. \tag{1.5}$$

This is most readily done by calculating Choleski factors (either $M = LL^T$ or $M = LDL^T$), which also enables the positive definiteness condition to be verified. In the context of an active set method, it is important to devise ways of updating both the representation of Z and the factors of M , in order to make the method as efficient as possible, subject to an adequate assurance of numerical stability.

A well known technique for defining Z is to border A with the columns of an arbitrary matrix V so that the matrix

$$B = [A \ V] \tag{1.6}$$

is nonsingular (it is a property of the active set method that A has full rank). Then the matrix Z defined by

$$B^{-T} = [Y \ Z] \tag{1.7}$$

is readily shown to provide a basis for the null-space of the equations $A^T \mathbf{s} = \mathbf{0}$. For reasons of both efficiency and numerical stability, B^{-T} is not held directly. Rather some suitable factors of B are kept, and are updated much as in linear programming. Products with Z or Z^T are effected by carrying out solves with the matrix B . This approach allows advantage to be taken of *sparsity* in the matrix A , which potentially allows much larger problems to be solved. In `bqpd`, both sparse and dense matrix algebra kernels are available, based on the use of *L-implicit-U factors*. See Fletcher [3] and [4] for further details. In the original version of `bqpd`, the columns of V were essentially arbitrary (they were columns of previously active constraints that had become inactive). In the revised version, columns of V are unit vectors, chosen with the aim of keeping B well-conditioned, as far as possible. This approach also induces more sparsity in the matrix B .

2 The Active Set Method

In this section the structure of the active set method is examined in more detail, particularly in regard to the implications for updating M . We may regard the method as a sequence of *major iterations*, each of which proceeds from a current FSEP to another FSEP. In absence of degeneracy, each major iteration reduces the value of the objective function, which provides a proof of termination. There is also the possibility if G is indefinite that a direction is generated along which the objective function can be reduced without bound, in which case the method also terminates.

Quite frequently the current FSEP is in fact a *vertex* of the feasible region. This is recognised by having n active constraints in \mathcal{A} , and the matrices V and hence Z are vacuous (that is, not present). Consider what happens if the QP iterations proceed from one vertex to another, much as in linear programming (except that the gradient $\mathbf{g}^{(k)}$ may change in quadratic programming). In this case the constraint p that is relaxed is replaced in \mathcal{A} by the new constraint q that becomes active in the line search. Thus all that needs to be done is to replace the column vector \mathbf{a}_p in B by the vector \mathbf{a}_q . This requires the corresponding update of the invertible representation of B . This is just the standard update of the simplex method for linear programming, and there are various efficient ways for doing this. The method used in `bqpd` is described in [3]. The code in

bqpd is organised in such a way that such simplex updates are used whenever vertex to vertex iterations take place, and no updates of M are necessary.

We now consider what happens when a vertex to vertex iteration does not occur. Let p be the constraint that is relaxed at the current FSEP $\mathbf{x}^{(k)}$ with active set \mathcal{A} . In this case a suitable search direction is the vector

$$\mathbf{s} = \mathbf{y}_p - ZM^{-1}Z^TG\mathbf{y}_p, \quad (2.1)$$

where \mathbf{y}_p denotes the column of B^{-T} corresponding to constraint p (see (1.7)), and Z and M refer to matrices that derive from \mathcal{A} . This formula can be deduced by relaxing constraint p in Equation (10.2.2) of [1], and using the definition of T in Equation (10.2.7). The formula reduces to $\mathbf{s} = \mathbf{y}_p$ when $\mathbf{x}^{(k)}$ is a vertex and V and Z are vacuous. A line search along \mathbf{s} is made. If a vertex to vertex iteration does not take place, then a new matrix Z appropriate to the active set \mathcal{A}/p is computed. This is referred to as *extending the null-space* since the number of columns of Z is increased by one. Likewise the number of rows and columns of M is increased by one, and it is necessary to update the factors of M . Details of these calculations are described in more detail in Section 3. If an unconstrained minimum of (1.3) is reached in the line search along \mathbf{s} then the resulting point is a new FSEP, and the major iteration is completed.

If however a new constraint q becomes active in the line search, then the new point is usually not an FSEP. The index q is added to \mathcal{A} and the new column \mathbf{a}_q is adjoined to A . A sequence of *minor iterations* now takes place. When M is positive definite, these use the search direction defined by $\mathbf{s} = Z\mathbf{u}$ where \mathbf{u} is obtained by solving (1.5). Until an FSEP is reached, each minor iteration adds a column to A and hence reduces the dimension of the null-space. Details of how Z and M may be updated are presented in Section 4. Eventually the sequence of minor iterations must terminate at a vertex FSEP, if not before, thus ending the current major iteration.

Some discussion of the inertia of the matrices M is important. We can only move to the solution of the EP (1.2) if M is positive definite, so that the current M is always positive definite at a (non-vertex) FSEP. However, if G is not positive definite, then it is possible that extending the null-space can introduce a single negative eigenvalue into M . On the other hand, reducing the null-space cannot introduce further negative eigenvalues, although it is possible that the single negative eigenvalue might persist (but not necessarily so). These assertions are readily proved using the structure of the updates described in Sections 3 and 4, and the use of Sylvester's Law of Inertia. How best to deal with this single negative eigenvalue is a major concern that is addressed in this paper. There are the issues both of what factors of M are most suitable, and of what to use in respect of a search direction when M is not positive definite. If $\mathbf{x}^{(k)}$ is an FSEP, then (2.1) can always be used to provide a search direction. However we make new proposals in Section 4 for calculating the search direction at the start of a minor iteration when M is not positive definite. These proposals are related to the new factorization scheme for M that we propose in Section 3.

A few remarks about degeneracy are appropriate since degeneracy is equally likely

to affect a QP problem as it is (and often does) for an LP problem. A *degeneracy block* refers to the situation in which a step of zero length is taken in the line search, due to a new constraint q immediately becoming active. (This constraint is active at the current point, but is not in the active set.) First we note that a degeneracy block is not a problem on minor iterations: it can just be ignored as it cannot give rise to cycling. The main difficulty occurs when a degeneracy block occurs in the line search leaving an FSEP along the direction \mathbf{s} given by (2.1). If the current FSEP is a vertex, then the degeneracy block can be resolved by any method that is valid for linear programming. The revised version of `bqp`d uses Wolfe's method [7] which works well. If the current FSEP is not a vertex, but $Z^T \mathbf{a}_q \neq \mathbf{0}$, then minor iterations can again be used without the possibility of cycling. In this case, if no progress is made, then the current FSEP is reduced to a vertex and is treated as above. The only awkward situation is when the current FSEP is not a vertex and $Z^T \mathbf{a}_q = \mathbf{0}$. In this case the active set obtained by replacing p by q is still an FSEP and minor iterations cannot be used. However the ideas expressed in Equations (5.9) onwards of Fletcher [2] can be used to remove the degeneracy block or to establish that the current point is optimal. In the revised version of `bqp`d, Wolfe's method is adapted to work in this situation.

3 Extending the Null-Space

In this section we introduce a new idea for representing the matrix M when it has a negative eigenvalue. We show how this representation of M can be updated when the null-space matrix Z is changed. We also describe the extra features needed to ensure that V is extended with a unit column in a stable way.

Extending the null-space only arises on leaving an FSEP, and when a vertex to vertex step does not occur. At an FSEP, the current matrix M is positive definite and we choose to represent it using Choleski factors

$$M = LL^T. \quad (3.1)$$

Changes to Z and M occur because we must remove the column vector \mathbf{a}_p from A . Hence it is necessary to augment V in (1.6) by one column to keep B nonsingular. A simple possibility is that the outgoing column \mathbf{a}_p derives from a simple bound constraint and so is a unit column that qualifies for inclusion in V . We examine this possibility first.

3.1 \mathbf{a}_p is a unit column

In this case we only need to transfer \mathbf{a}_p from the A partition to the V partition in (1.6). Correspondingly the column \mathbf{y}_p of Y in (1.7) is transferred to the Z partition. We assume that \mathbf{y}_p becomes the new rightmost column of Z . Then the new matrix M may be expressed as

$$M = [Z \ \mathbf{y}_p]^T G [Z \ \mathbf{y}_p] = \begin{bmatrix} LL^T & \mathbf{m} \\ \mathbf{m}^T & \mu \end{bmatrix}, \quad (3.2)$$

where $\mathbf{m} = Z^T G \mathbf{y}_p$ and $\mu = \mathbf{y}_p^T G \mathbf{y}_p$. Let \mathbf{l} denote the solution of $L\mathbf{l} = \mathbf{m}$ and let $\lambda = \mu - \mathbf{l}^T \mathbf{l}$. If $\lambda > 0$ then the new M matrix is positive definite and its Choleski factor is

$$\begin{bmatrix} L & \\ \mathbf{l}^T & \lambda^{1/2} \end{bmatrix}. \quad (3.3)$$

In this case there is nothing more to do. We note that \mathbf{m} and \mathbf{l} are available from the calculation of \mathbf{s} in (2.1) so that this extension of L is achieved at negligible cost.

On the other hand, if $\lambda \leq 0$, then Choleski factors no longer exist (or are singular). One way around the difficulty is to use LDL^T factors rather than LL^T factors. Then it is possible to extend L and D and λ becomes the additional diagonal element of D . However Gill and Murray [5] rightly point out that if M is nearly singular before the update, then substantial growth in $|\lambda|$ can occur. The associated large rounding error is then a source of numerical instability when $\lambda < 0$.

To handle this situation we introduce a new idea in which we first decompose the border as

$$\begin{bmatrix} 0 & \mathbf{m} \\ \mathbf{m}^T & \mu \end{bmatrix} = \mathbf{b}\mathbf{b}^T - \mathbf{a}\mathbf{a}^T \quad (3.4)$$

in a stable way, as described in the next paragraph. Then the new matrix M may be expressed as

$$M = \left\{ \begin{bmatrix} LL^T & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \mathbf{b}\mathbf{b}^T \right\} - \mathbf{a}\mathbf{a}^T. \quad (3.5)$$

Choleski factors of the term in curly brackets are readily computed. (Essentially the lower Hessenberg matrix

$$\left[\mathbf{b} \mid \begin{array}{c} L \\ \mathbf{0}^T \end{array} \right] \quad (3.6)$$

is reduced to a lower triangular matrix by a forward sequence of Givens' rotations from the right, in which each rotation is applied to adjacent columns of the lower Hessenberg matrix, as illustrated in the Appendix. See also Golub and Van Loan [6] for example.) Denoting the resulting matrix by L , then

$$M = LL^T - \mathbf{a}\mathbf{a}^T \quad (3.7)$$

forms the new representation of M when M is not positive definite.

The decomposition (3.4) requires \mathbf{a} and \mathbf{b} to be of the form

$$\mathbf{a} = \begin{pmatrix} k\mathbf{m} \\ \alpha \end{pmatrix} \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} k\mathbf{m} \\ \beta \end{pmatrix} \quad (3.8)$$

where k , α and β are to be determined. It follows from (3.4) that $\beta^2 - \alpha^2 = \mu$ and $k(\beta - \alpha) = 1$, and hence that $\beta + \alpha = k\mu$. Thus we have two equations and three parameters, so we are free to choose one parameter in an arbitrary way. If we fix k , then we can determine β from

$$\beta = \frac{1}{2} \left(k\mu + \frac{1}{k} \right). \quad (3.9)$$

Alternatively, if we fix β , then we can determine k from

$$k = 1/(\beta + (\beta^2 - \mu)^{1/2}). \quad (3.10)$$

In both cases $\alpha = k\mu - \beta$.

There are various ways in which we might fix the free parameter. The method which has been implemented has the property that the partitions of $\mathbf{b}\mathbf{b}^T$ arising from (3.8) are of the same magnitude as the partitions on the right hand side of (3.2), when this matrix is badly scaled. This is the choice

$$\beta = \max(\tau, (\mathbf{m}^T \mathbf{m} / l_{11}^2 + |\mu|)^{1/2}), \quad (3.11)$$

where τ is a small tolerance to allow for the case that $\mathbf{m} = \mathbf{0}$ and $\mu = 0$, and l_{11} is the leading element of L in (3.2). We allow for two types of bad scaling: one is when the magnitude of G (γ say), is significantly different from unity, and the other when $\xi = \|\mathbf{y}_p\|$ is large. However we assume that the magnitude of Z is around unity. Thus the 1,1 partition of (3.2) has magnitude γ , the 2,1 partition has magnitude $\gamma\xi$ and the 2,2 partition has magnitude $\gamma\xi^2$ (using magnitude in the sense of order of magnitude). We shall see that our choice of β is such that the partitions of $\mathbf{b}\mathbf{b}^T$ have the same magnitude. We note that l_{11}^2 has magnitude γ , \mathbf{m} has magnitude $\gamma\xi$ and μ has magnitude $\gamma\xi^2$. Hence β^2 , which is the 2,2 partition, has magnitude $\gamma\xi^2$. It follows from (3.11) and (3.10) that $k\beta$ is of order unity, so that the 2,1 partition has magnitude $\gamma\xi$. Finally, because k has magnitude β^{-1} , it follows that the 1,1 partition of $\mathbf{b}\mathbf{b}^T$, which is $k^2\mathbf{m}\mathbf{m}^T$, has magnitude γ . The main consequence of all this is that the matrix in curly brackets in (3.5) is scaled in the same way as the matrix on the right hand side of (3.2). Practical experience with the method has been entirely satisfactory.

Another method that was coded is to choose $\beta = 1$ where possible, in which case it is readily shown that $|\det L|$ remains unchanged. However, if $\mu > 1$ then k becomes undefined. In this case the minimum possible value $\beta = \mu^{1/2}$ is chosen. This method does not share the scaling properties of the previous method, although it usually works well in practice. However one occasion was noticed in which G was very small and significant errors relative to G were caused by the update.

A further choice that is appealing at first sight is to choose \mathbf{b} and \mathbf{a} to be the non-trivial eigenvectors of (3.4). This choice is most readily implemented by noting that the condition $\mathbf{b}^T \mathbf{a} = 0$ must hold, since the matrix is symmetric. This leads to the expression

$$k = (4\mathbf{m}^T \mathbf{m} + \mu^2)^{-1/4}.$$

Hence the effect on M is given by

$$M := \begin{bmatrix} 1 & & -d_{p+1}/d_p \\ & \ddots & \vdots \\ & & 1 & -d_n/d_p \\ & & & d_p^{-1} \end{bmatrix} \begin{bmatrix} M & \mathbf{m} \\ \mathbf{m}^T & \mu \end{bmatrix} \begin{bmatrix} 1 & & -d_{p+1}/d_p \\ & \ddots & \vdots \\ & & 1 & -d_n/d_p \\ & & & d_p^{-1} \end{bmatrix}^T. \quad (3.14)$$

If the updated M in Subsection 3.1 is positive definite ($\lambda > 0$), then the effect on the updated Choleski factor (3.3) may be written as

$$\begin{bmatrix} 1 & & -d_{p+1}/d_p \\ & \ddots & \vdots \\ & & 1 & -d_n/d_p \\ & & & d_p^{-1} \end{bmatrix} \begin{bmatrix} L \\ \mathbf{I}^T & \lambda^{1/2} \end{bmatrix}. \quad (3.15)$$

The resulting matrix may again be returned to lower triangular form by applying Givens's rotations from the right. First a backward sequence of rotations is applied which eliminates the elements of last row $[\mathbf{I}^T \lambda^{1/2}]$ in reverse order, so that only the first element is non-zero (see the Appendix). These operations make L into a lower Hessenberg matrix. The resulting matrices in (3.15) may then be multiplied out, still leaving a lower Hessenberg matrix. Finally the lower Hessenberg matrix may be reduced to lower triangular form using a forward sequence of rotations, as described in Subsection 3.1.

If M has the form given by (3.7), then the same procedure is followed to update L , and \mathbf{a} is modified by

$$\mathbf{a} := \begin{bmatrix} 1 & & -d_{p+1}/d_p \\ & \ddots & \vdots \\ & & 1 & -d_n/d_p \\ & & & d_p^{-1} \end{bmatrix} \mathbf{a}. \quad (3.16)$$

This section finishes with some remarks about numerical stability. If the extended matrix in (3.2) is positive definite then there are no numerical difficulties in extending the Choleski factors as in (3.3). If \mathbf{y}_p is large then this is reflected in the last row of (3.3) becoming large. If this matrix is subsequently transformed as described after Equation (3.15), then the backward sequence of Givens' rotation transforms the large last row of (3.3) into a single large element. These large numbers do not affect other rows of the factor. The next step is to multiply out the resulting matrix. At this stage the single large element is multiplied by d_p^{-1} . Since $d_p = \mathbf{y}_p^T \mathbf{a}_q$ is of order ξ , dividing by d_p removes the growth caused by ξ being large. Thus \mathbf{y}_p being large does not cause any numerical difficulties in this case.

If the extended matrix is not positive definite, then the new representation is that given by (3.7). First we note that \mathbf{b} and \mathbf{a} do not depend upon L , so that any ill-conditioning in L does not cause growth. If \mathbf{y}_p is large then we have chosen \mathbf{b} so that

growth is confined to the 2,1 and 2,2 partitions of $\mathbf{b}\mathbf{b}^T$. Thus the updated Choleski factors obtained from the term in curly brackets of (3.5) will have growth confined to the last row of L . The argument of the previous paragraph can be used to show that this growth is safely removed using the procedure following Equation (3.15). Likewise growth in \mathbf{a} is confined to the last element of \mathbf{a} and is removed in a similar way by the operation in (3.16).

4 Reducing the Null-Space

4.1 Updating Z and the factors of M

Reducing the null-space arises during the minor iteration sequence, as new constraints become active. The vertex to vertex case is treated separately so we can assume that initially the dimension of the null-space is at least one. When a new constraint q becomes active, we adjoin the column vector \mathbf{a}_q to A , so in this case we must remove a column from V . If V has just one column then the choice is fixed, V , Z and M become vacuous, and it only remains to update B , as described in the previous section. If V has more than one column then we choose the outgoing column to inhibit growth in Z .

The update of B may again be expressed by (3.12). In this situation we are given \mathbf{a}_q and need to choose \mathbf{a}_p , which is the reverse of the situation in Subsection 3.1. Essentially we are free to choose d_p from amongst the elements of \mathbf{d} that correspond to Z , that is from amongst the elements of $Z^T\mathbf{a}_q$. The update to Z can then be expressed as

$$Z^T := \left[\begin{array}{cccc} 1 & & & r_1 \\ & \ddots & & r_2 \\ & & 1 & \vdots \\ & & & \vdots & 1 \\ & & & \vdots & & \ddots \\ & & & r_t & & & 1 \end{array} \right] Z^T \quad (4.1)$$

where t is the dimension of the new null-space and $r_i = -d_i/d_p$ where i ranges over the elements of the old null-space, excluding the index p . The form of (4.1) clearly indicates that we should choose the maximum modulus element of $Z^T\mathbf{a}_q$ to determine p . It then follows that $|r_i| \leq 1$ which gives the best bound on the growth in Z . We shall continue to use p to denote the column in (4.1) in which the elements of \mathbf{r} appear.

The resulting effect on the lower triangular factor of M may be expressed as

$$\left[\begin{array}{cccc} 1 & & & r_1 \\ & \ddots & & r_2 \\ & & 1 & \vdots \\ & & & \vdots & 1 \\ & & & \vdots & & \ddots \\ & & & r_t & & & 1 \end{array} \right] L \quad (4.2)$$

and we now consider how this matrix may be returned to triangular form. The situation is similar to that in (3.15), except that column p is not usually the last column of the old Z , and column p is absent from the new Z . To handle this situation we first permute the \mathbf{r} column to be the rightmost column and shift left the intermediate unit columns. Likewise we permute row p of L to be the bottom row, and shift up the intermediate rows (see the Appendix). The resulting configuration is then handled as before. A sequence of backward rotations from the right, starting in column p , is used to reduce the new last row of L to a single non-zero in column 1. Then the product is multiplied out and the resulting lower Hessenberg matrix is reduced to a lower triangular matrix by a forward sequence of rotations. If M is represented by (3.7) then \mathbf{a} must also be updated, in a similar manner to (3.16).

A special case of these calculations occurs if the new column \mathbf{a}_q is already a column of V . In this case no update of B is required. Also $Z^T \mathbf{a}_q$ is a unit vector and the elements r_i in (4.1) are all zero. In this case the product may be multiplied out directly and the resulting lower Hessenberg matrix is reduced to lower triangular form by forward rotations starting in column p . If M is represented by (3.7), the update on \mathbf{a} is just to delete the p -th element.

Whenever M is not positive definite, and so is represented by (3.7), it is possible (but not necessary) that the reduction process might result in M becoming positive definite. In this case therefore, a check is made after M is updated to see if the resulting matrix is positive definite. We first solve $L\mathbf{w} = \mathbf{a}$, in which case it follows that

$$M = L(I - \mathbf{w}\mathbf{w}^T)L^T. \quad (4.3)$$

Hence the inertia of M is that of $I - \mathbf{w}\mathbf{w}^T$ and M is positive definite if and only if $\mathbf{w}^T \mathbf{w} < 1$. If so, (4.3) is returned to Choleski factors in the following way. First a sequence of backward rotations is applied from the left to \mathbf{w} which reduces it to $\omega \mathbf{e}_1$. The same sequence of rotations is applied from the right to L giving a lower Hessenberg matrix. Then the first column of the Hessenberg matrix is scaled by $(1 - \omega^2)^{1/2}$. Finally forward rotations are applied to restore the resulting Hessenberg matrix to lower triangular form.

4.2 The minor iteration search direction

If the reduced Hessian matrix M is positive definite then we must choose the search direction to minimize the reduced quadratic function (1.3), that is by $\mathbf{s} = Z\mathbf{u}$ where \mathbf{u} is defined by (1.5). If M has a negative eigenvalue, then the EP (1.2) does not have a solution and we are free to search in any direction, as long as it does not increase the objective function. The representation (3.7) suggests a very convenient search direction, occasioned by the observation that if \mathbf{w} is defined by $L\mathbf{w} = \mathbf{a}$, as in the previous paragraph, and \mathbf{v} is defined by $L^T \mathbf{v} = \mathbf{w}$, then the vector $\mathbf{s} = Z\mathbf{v}$ is a negative curvature search direction if $\mathbf{w}^T \mathbf{w} > 1$, by virtue of

$$\mathbf{s}^T G \mathbf{s} = \mathbf{v}^T Z^T G Z \mathbf{v} = \mathbf{v}^T M \mathbf{v} = \mathbf{v}^T L(I - \mathbf{w}\mathbf{w}^T)L^T \mathbf{v} = \mathbf{w}^T \mathbf{w} - (\mathbf{w}^T \mathbf{w})^2 < 0.$$

Likewise, \mathbf{s} is a zero curvature search direction if $\mathbf{w}^T \mathbf{w} = 1$. In fact, we choose $\mathbf{s} = \sigma Z \mathbf{v}$ as the search direction, where $\sigma = \text{sign}(-\mathbf{v}^T Z^T \mathbf{g})$ and $Z^T \mathbf{g}$ is the current reduced gradient. This ensures that $\mathbf{s}^T \mathbf{g} = \sigma \mathbf{g}^T Z \mathbf{v} \leq 0$ and hence \mathbf{s} is a direction along which the objective function is non-increasing.

We can also derive formulae for updating the reduced gradient after each minor iteration. If M is positive definite, then the update is

$$Z^T \mathbf{g}^+ = (1 - \theta) Z^T \mathbf{g}$$

where θ is the step along \mathbf{s} taken in the line search. If M is not positive definite, and \mathbf{s} is defined as in the previous paragraph, then the update is

$$\begin{aligned} Z^T \mathbf{g}^+ &= Z^T \mathbf{g} + \theta Z^T G \mathbf{s} = Z^T \mathbf{g} + \sigma \theta M \mathbf{v} = Z^T \mathbf{g} + \sigma \theta (LL^T - \mathbf{a} \mathbf{a}^T) L^{-T} L^{-1} \mathbf{a} \\ &= Z^T \mathbf{g} + \sigma \theta (1 - \mathbf{w}^T \mathbf{w}) \mathbf{a}. \end{aligned}$$

Thus a multiple of \mathbf{a} is added to $Z^T \mathbf{g}$ in this case.

5 Discussion

We conclude by considering other canonical forms that might be used to represent the reduced Hessian matrix M . First of all, it is possible to use LDL^T factors in place of LL^T factors. These avoid the use of square roots in the rotations and enable the rotations to be carried out with somewhat fewer floating point operations. However there are some stability and scaling problems which, though not insuperable, are tedious to code. I have heard Morven Gentleman remark that in practice there is very little to be gained by going to LDL^T factors, and this is also my impression.

We have already referred to the fact that LDL^T factors are not satisfactory when M has a negative eigenvalue due to the possibility of growth in the negative element of D . Another possible canonical form is

$$M = L(I - \mathbf{w} \mathbf{w}^T) L^T, \quad (5.1)$$

which can also be updated efficiently. However the possibility of growth in $\mathbf{w} = L^{-1} \mathbf{a}$ is a disincentive. The possibility of using

$$M = \begin{bmatrix} LL^T & \mathbf{m} \\ \mathbf{m}^T & \mu \end{bmatrix} \quad (5.2)$$

itself as a canonical form was also considered. However we observe that it may not be possible to reduce this form satisfactorily, as evidenced by the equation

$$\begin{bmatrix} 1 & & & -1 \\ & 1 & & -1 \\ & & 1 & -1 \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & & & 1 \\ & 1 & & 1 \\ & & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ -1 & -1 & -1 & \end{bmatrix} = \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix}. \quad (5.3)$$

in which symmetric factors of the leading 2×2 submatrix on the right hand side do not exist.

6 Appendix

In this appendix we illustrate some of the procedures using Givens' rotations that are described in the text. Note that each rotation is applied to the right of L , and the transpose rotation is applied to the left of L^T so that the product LL^T is unchanged. We show the effect of the rotations on L . The two columns which are affected by each rotations are shown under the arrows. A newly created zero element is indicated by a 0, and a newly created non-zero element by a +.

Returning (3.6) to triangular form by forward rotations

$$\begin{array}{c}
 \left[\begin{array}{ccccc} \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & & & & \end{array} \right] \xrightarrow{1,2 \text{ rotation}} \left[\begin{array}{ccccc} \times & 0 & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & + & & & \end{array} \right] \xrightarrow{2,3 \text{ rotation}} \left[\begin{array}{ccccc} \times & & & & \\ \times & \times & 0 & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & \times & + & & \end{array} \right] \\
 \\
 \xrightarrow{3,4 \text{ rotation}} \left[\begin{array}{ccccc} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & 0 & \\ \times & \times & \times & \times & \times \\ \times & \times & \times & + & \end{array} \right] \xrightarrow{4,5 \text{ rotation}} \left[\begin{array}{ccccc} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & 0 \\ \times & \times & \times & \times & + \end{array} \right]
 \end{array}$$

Returning (3.15) to triangular form by backward rotations

First we start with rotations on L .

$$\begin{array}{c}
 \left[\begin{array}{ccccc} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \end{array} \right] \xrightarrow{4,5 \text{ rotation}} \left[\begin{array}{ccccc} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & + \\ \times & \times & \times & \times & 0 \end{array} \right] \xrightarrow{3,4 \text{ rotation}} \left[\begin{array}{ccccc} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & + & \\ \times & \times & \times & \times & \times \\ \times & \times & \times & 0 & \end{array} \right] \\
 \\
 \xrightarrow{2,3 \text{ rotation}} \left[\begin{array}{ccccc} \times & & & & \\ \times & \times & + & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & \times & 0 & & \end{array} \right] \xrightarrow{1,2 \text{ rotation}} \left[\begin{array}{ccccc} \times & + & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & 0 & & & \end{array} \right]
 \end{array}$$

Then we multiply out the product in (3.15).

$$\left[\begin{array}{ccccc} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & \times \end{array} \right] \left[\begin{array}{ccccc} \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & & & & \end{array} \right] = \left[\begin{array}{ccccc} \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & & & & \end{array} \right].$$

Finally the lower Hessenberg matrix is reduced to lower triangular form as above.

Returning (4.2) to triangular form by backward rotations

First the product is reordered so that the spike column comes at the end.

$$\begin{bmatrix} 1 & & & & \\ & \times & & & \\ & 1 & \times & & \\ & & \times & 1 & \\ & & & \times & 1 \end{bmatrix} \begin{bmatrix} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \times & \\ & & & \times & \times \\ & & & \times & \times & \times \\ & & & \times & \times & \times & \times \end{bmatrix}$$

Then the last row of the right hand side matrix is reduced to a single non-zero element.

$$\begin{bmatrix} \times & & & & \\ \times & \times & & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & \times & \times & & \end{bmatrix} \xrightarrow{\text{2,3 rotation}} \begin{bmatrix} \times & & & & \\ \times & \times & + & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & \times & 0 & & \end{bmatrix} \xrightarrow{\text{1,2 rotation}} \begin{bmatrix} \times & + & & & \\ \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \\ \times & 0 & & & \end{bmatrix}$$

Finally the product is multiplied out and the 4×5 lower Hessenberg matrix is reduced to a 4×4 lower triangular matrix by forward rotations.

7 References

- [1] Fletcher R., (1987), *Practical Methods of Optimization, 2nd Edition*, John Wiley, Chichester.
- [2] Fletcher R., (1993), Resolving degeneracy in quadratic programming, *Annals of O.R.*, **47**, 307–334.
- [3] Fletcher R., (1997), Dense Factors of Sparse Matrices, in *Approximation Theory and Optimization. Tributes to M.J.D. Powell*, (M.D. Buhmann and A. Iserles, eds), Cambridge University Press (1997), pp. 145–166.
- [4] Fletcher R., (1998), Block Triangular Orderings and Factors for Sparse Matrices in LP, in *Numerical analysis 1997* (D.F. Griffiths, D.J. Higham and G.A. Watson, eds.), Pitman Research Notes in Mathematics 380, Longman, Harlow, pp. 91–110.
- [5] Gill P.E. and Murray W., (1978), Numerically stable methods for quadratic programming, *Mathematical Programming*, **14**, 349–372.
- [6] Golub G.H. and Van Loan C.F., (1989) *Matrix Computations, 2nd Edition*, The Johns Hopkins University Press, Baltimore.
- [7] Wolfe P., (1963). A technique for resolving degeneracy in linear programming, *J. SIAM*, **11**, 205–211.