SIAM J. Optim.
Vol. 9, No. 4, pp. 1041–1063

ON MODIFIED FACTORIZATIONS FOR LARGE-SCALE LINEARLY CONSTRAINED OPTIMIZATION*

NICHOLAS IAN MARK GOULD†

This paper is dedicated to John Dennis, Jr., as a token of my appreciation for his unswerving support for computational mathematical programming

Abstract. We consider the algebraic issues concerning the solution of general, large-scale, linearly constrained nonlinear optimization problems. Particular attention is given to suitable methods for solving the linear systems that occur at each iteration of such methods. The main issue addressed is how to ensure that a quadratic model of the objective function is positive definite in the null-space of the constraints while neither adversely affecting the convergence of Newton's method nor incurring a significant computational overhead. Numerical evidence to support the theoretical developments is provided.

Key words. large-scale problems, linearly constrained optimization, modified matrix factorizations, second-order optimality conditions

AMS subject classifications. 65F05, 65F10, 65F15, 65F50, 65K05, 90C30

PII. S1052623495290660

1. Introduction.

Newton-like line-search methods for unconstrained and linearly constrained optimization may be broadly divided into two categories on the basis of how they deal with nonconvexity. Some methods wholeheartedly embrace nonconvexity by calculating directions of negative curvature as a means of escape from regions of nonconvexity. Others prefer to pretend that the nonconvexity is not present by replacing the Newton model by a convex modification. Although the former approach has theoretical advantages (see, for example, [35]), the latter is often preferred because of its simplicity.

The prototypical example of convex modification is the modified Cholesky method of [25] for unconstrained minimization. Here the second derivatives $\nabla^2_{xx} f$ of the objective function $f(\boldsymbol{x})$ are replaced by a modification $\boldsymbol{B} = \nabla^2_{xx} f + \boldsymbol{D}$ whenever $\nabla^2_{xx} f$ is insufficiently positive definite. The diagonal perturbation $\boldsymbol{D}$ is chosen so that $\boldsymbol{B}$ is sufficiently positive definite, that is,

$$\boldsymbol{p}^T \boldsymbol{B} \boldsymbol{p} \geq \sigma \boldsymbol{p}^T \boldsymbol{p} \text{ for some constant } \sigma > 0 \text{ and all } \boldsymbol{p}. \tag{1.1}$$

Most significantly, the perturbation is determined during an attempted Cholesky factorization of $\nabla^2_{xx} f$. If $\nabla^2_{xx} f$ is itself sufficiently positive definite, $\boldsymbol{D}$ is zero. The cost of finding $\boldsymbol{B}$ is barely more than the cost of a Cholesky factorization, and the norm of the resulting $\boldsymbol{B}$ has a guaranteed bound. More recently, [38] developed a similar method with a better a priori bound, while extensions to large-scale unconstrained and bound constrained optimization, using sparse factorizations, have been proposed by [26], [7, Chapter 3], and [37]. A thorough survey of modified Newton methods for unconstrained optimization is given by [9].

In this paper, we are interested in extending these ideas to linearly constrained optimization. We shall not concern ourselves with inequality constraints but presume

*Received by the editors August 23, 1995; accepted for publication (in revised form) November 17, 1997; published electronically September 24, 1999.
http://www.siam.org/journals/siopt/9-4/29066.html

†Atlas Centre, Rutherford Appleton Laboratory, Oxfordshire OX11 0QX, England (n.gould@rl.ac.uk).

that these are being handled by active set or barrier methods (see, for instance, [27, section 5.2] and [17]). Thus we aim to solve a smooth linearly constrained, nonlinear optimization problem,

$$\underset{x \in \Re^n}{\text{minimize}} \quad f(\boldsymbol{x}), \tag{1.2}$$

subject to m general, linearly independent, linear equations

$$\boldsymbol{Ax} = \boldsymbol{b}. \tag{1.3}$$

We consider a general iteration in which we have a point $\boldsymbol{x}$ satisfying (1.3) and wish to determine an improvement $\boldsymbol{x} + \boldsymbol{p}$. We build a second-order model of the objective function and pick $\boldsymbol{p}$ as the solution of the equality constrained quadratic program

$$\underset{\boldsymbol{p} \in \Re^n}{\text{minimize}} \quad \tfrac{1}{2}\boldsymbol{p}^T\boldsymbol{Bp} + \boldsymbol{p}^T\boldsymbol{g} \quad \text{subject to} \quad \boldsymbol{Ap} = \boldsymbol{0}. \tag{1.4}$$

Here $\boldsymbol{g} \overset{\text{def}}{=} \nabla_x f$ is the gradient of f and $\boldsymbol{B}$ is a suitable symmetric approximation to the Hessian $\boldsymbol{H} \overset{\text{def}}{=} \nabla^2_{xx} f$. We wish to guarantee that $\boldsymbol{p}$ is bounded and thus we require that the model problem is bounded from below. We ensure this by requiring that $\boldsymbol{B}$ be *second-order sufficient*, that is, that

$$\begin{array}{c} \boldsymbol{p}^T\boldsymbol{Bp} \geq \sigma \boldsymbol{p}^T\boldsymbol{p} \ \text{ for some constant } \ \sigma > 0 \\ \text{and all } \ \boldsymbol{p} \ \text{ satisfying } \ \boldsymbol{Ap} = \boldsymbol{0}. \end{array} \tag{1.5}$$

This condition is the natural generalization of (1.1) for the constrained case. To ensure rapid asymptotic convergence, we also require that $\boldsymbol{B}$ be equal to $\boldsymbol{H}$ whenever the latter is itself second-order sufficient.

We shall be concerned with general, large-scale problems so we will not consider methods based solely on dense factorizations. We presume that $\boldsymbol{H}$ and $\boldsymbol{A}$ are sparse, and we consider sparse direct methods. So long as $\boldsymbol{B}$ is second-order sufficient, the solution to (1.4) satisfies the sparse linear system

$$\begin{pmatrix} \boldsymbol{B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix}, \tag{1.6}$$

where $\boldsymbol{\lambda}$ are Lagrange multipliers. Note that the coefficient matrix,

$$\boldsymbol{K} \overset{\text{def}}{=} \begin{pmatrix} \boldsymbol{B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}, \tag{1.7}$$

of (1.6) is inevitably indefinite—it must have at least m positive and m negative eigenvalues (see [28]). Thus any matrix factorization of (1.7) must be capable of handling indefinite matrices. To be efficient, one would normally try to exploit the symmetry of $\boldsymbol{K}$ in the factorization. The natural generalization of the Cholesky (or more precisely $\boldsymbol{LDL}^T$) factorization in the symmetric, indefinite case is that first proposed by [5] and later improved by [4] and [18] in the dense case and [16] and [14] in the sparse case. Here a symmetric matrix $\boldsymbol{K}$ is decomposed as

$$\boldsymbol{K} = \boldsymbol{PLDL}^T\boldsymbol{P}^T, \tag{1.8}$$

where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{L}$ is unit lower triangular, and $\boldsymbol{D}$ is block diagonal with blocks of size at most two. Each diagonal block corresponds to a pivoting operation. We shall refer to the blocks as 1 by 1 and 2 by 2 pivots.

Because we are particularly concerned with the large-scale case, it is the Duff–Reid variant that is of special interest. We note that the permutation matrices are used extensively in the factorization of sparse matrices to keep the fill-in—that is, the introduction of extra nonzeros in the factors—at an acceptable level. Unfortunately, the [30] implementation, `MA27`, [13] of the Duff–Reid variant sometimes proves inefficient when applied to matrices of the form (1.7) as the analysis phase treats the whole diagonal of $\boldsymbol{K}$ as if it contains nonzero entries. Thus a good predicted ordering supplied by the analyze phase is often replaced, for stability reasons, by a less satisfactory ordering when the factorization is performed, resulting in considerable extra work and fill-in. Ways of avoiding these difficulties, and of taking further advantage of the zero block in $\boldsymbol{K}$, have been suggested by [12] and form the basis for a recent Harwell Subroutine Library code `MA47` [15].

In the special case when f is separable, $\boldsymbol{H}$ will be diagonal. In particular, when f is also strictly convex, $\boldsymbol{H}$ will be positive definite and a block elimination of $\boldsymbol{H}$ followed by a sparse Cholesky factorization of the (negative of the) Schur complement $\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^T$ is feasible. Indeed, this approach is fundamental to many interior point methods for linear programming (see, for example, [34], [31], [32], or [6]). However, because such an approach is merely the restriction of a particular pivot order applied to (1.7), and because it is less appealing when $\boldsymbol{H}$ is not diagonal, Fourer and Mehrotra [22] have suggested methods for solving (1.7) using more general pivot sequences for linear programming problems, and Vanderbei and Carpenter [40] do the same for general problems.

If $\boldsymbol{B}$ is known a priori to be second-order sufficient, as for instance would be the case if $f(\boldsymbol{x})$ were strictly convex, we wholeheartedly recommend the use of `MA27`, `MA47`, or the procedure within `loqo` [40] to solve (1.6). When there is a chance that $\boldsymbol{B}$ may not be second-order sufficient, alternatives to blindly solving (1.6) must be sought. We note that it is always possible to determine a posteriori if $\boldsymbol{B}$ is second-order sufficient, as [28] showed that $\boldsymbol{B}$ is second-order sufficient if and only if $\boldsymbol{K}$ has precisely m negative and n positive eigenvalues. The inertia of $\boldsymbol{K}$ is trivially obtained by summing the inertia of the pivots.

Having set the scene, we may now describe our goals. We aim

(i) to determine a matrix $\boldsymbol{B} = \boldsymbol{H} + \boldsymbol{E}$ so that (1.5) is satisfied and such that the perturbation $\boldsymbol{E} = \boldsymbol{0}$ whenever $\boldsymbol{H}$ satisfies (1.5);

(ii) to obtain $\boldsymbol{E}$ without incurring undue overheads above those normally considered acceptable when calculating the search direction;

(iii) to ensure that $\|\boldsymbol{E}\|$ is bounded relative to $\max(\|\boldsymbol{A}\|, \|\boldsymbol{H}\|)$—provided that $\{\boldsymbol{x}\}$ remains bounded, this will ensure that $\boldsymbol{B}$ is uniformly bounded;

(iv) to use the sparsity and structure of (1.6) to derive a sparse factorization; and

(v) to limit numerical growth to acceptable limits to ensure a stable algorithm.

In this paper, we shall show how, to a certain extent, we may achieve these aims. The paper is organized as follows. In section 2 we describe Forsgren and Murray's modified factorization approach [21]. In sections 3 and 4, we describe techniques which are particularly attractive when the systems are large and sparse, and we indicate in section 5 how the ideas presented in this paper behave in practice, using a prototype code.

We shall denote the inertia of the generic symmetric matrix $\boldsymbol{M}$ as

$$\text{In}\,(\boldsymbol{M}) = (m_+, m_-, m_0), \tag{1.9}$$

where m_+, m_-, and m_0 are, respectively, the numbers of positive, negative, and zero eigenvalues of $\boldsymbol{M}$. The n by n identity matrix will be written as $\boldsymbol{I}_n$, or $\boldsymbol{I}$ when the dimension is clear from the context. Finally, $\boldsymbol{e}_i$ will be the ith column of $\boldsymbol{I}$. We stress that throughout the paper, the matrix $\boldsymbol{B}$ denotes a second-order sufficient approximation to $\boldsymbol{H}$ which will actually be $\boldsymbol{H}$ whenever the latter is itself second-order sufficient.

2. Forsgren and Murray's sufficient pivoting conditions.

As far as we are aware, the only serious attempt to generalize the modified Cholesky methods for unconstrained optimization to the general, large-scale, linearly constrained case is that by Forsgren and Murray [21]. All other methods we are aware of either are appropriate only for small-scale calculations because they disregard problem structure (see [19, section 11.1] or [27, section 5.1] for example) or implicitly assume that $n-m$ is sufficiently small that coping with dense matrices of order $n-m$ is practicable (see, for instance, [36]).

We say that the first n rows of $\boldsymbol{K}$ are $\boldsymbol{B}$-rows and the remaining m rows are $\boldsymbol{A}$-rows. Forsgren and Murray show that, if the pivots are restricted to be of certain types until all of the $\boldsymbol{A}$-rows of $\boldsymbol{K}$ have been eliminated, the remaining uneliminated (Schur-complement) matrix, $\boldsymbol{S}$, is sufficiently positive definite if and only if $\boldsymbol{B}$ is second-order sufficient. Until all $\boldsymbol{A}$-rows of $\boldsymbol{K}$ have been exhausted, Forsgren and Murray allow only the following types of pivots:

$\boldsymbol{b}_+$ pivots: strictly positive 1 by 1 pivots occurring in $\boldsymbol{B}$-rows of $\boldsymbol{K}$.

$\boldsymbol{a}_-$ pivots: strictly negative 1 by 1 pivots occurring in $\boldsymbol{A}$-rows of $\boldsymbol{K}$.

$\boldsymbol{ba}$ pivots: 2 by 2 pivots with a strictly negative determinant, one of whose rows is a $\boldsymbol{B}$-row and the other of whose rows is an $\boldsymbol{A}$-row of $\boldsymbol{K}$.

Forsgren and Murray further restrict the pivot so that the absolute value of its determinant is greater than a small positive constant so as to bound the elements in $\boldsymbol{L}$ and limit any growth in $\boldsymbol{S}$. The motivation behind this choice of pivot is simply that if i $\boldsymbol{A}$-rows have been eliminated, the factorized matrix has exactly i negative eigenvalues. Thus, when all $\boldsymbol{A}$-rows have been eliminated, the factorized matrix has precisely m negative eigenvalues and hence any further negative eigenvalues in $\boldsymbol{S}$ can occur only because $\boldsymbol{B}$ is not second-order sufficient.

Once $\boldsymbol{S}$ has been determined, Forsgren and Murray form a stable symmetric indefinite factorization. If the factors reveal that $\boldsymbol{S}$ is sufficiently positive definite, the (quasi-) Newton equations (1.6) are subsequently solved using the factorization. If $\boldsymbol{S}$ has insufficiently many positive eigenvalues, Forsgren and Murray show how both a direction of sufficient descent and a direction of negative curvature may be recovered from the factors, and they form a search arc as a linear or nonlinear combination of these two directions.

An obvious variation is, instead, to form a modified Cholesky factorization of $\boldsymbol{S}$. If no modification is performed, the true Hessian $\boldsymbol{H}$ must be second-order sufficient. Otherwise, a suitable perturbation $\boldsymbol{E}$ will have been produced. In either case, the Newton equations (1.6) are solved using the complete factorization.

The main difficulty with Forsgren and Murray's approach is that any restriction on the pivot order can disqualify potentially advantageous sparsity orderings. While it is always possible to choose a pivot according to the Forsgren–Murray recipe, the available choices all may lead to considerable fill-in. Nonetheless, we shall consider a number of variations of this scheme.

3. Methods using *ba* pivots.

In this section, we consider a scheme which uses a restricted version of [21]'s pivoting rules. Specifically, we consider what happens if we choose the first m pivots to be *ba* pivots. This choice of pivots is covered by Forsgren and Murray's analysis. We are particularly interested in these pivots because the fill-in is easy to predict and, most important, the stability of the method is determined entirely by $\boldsymbol{A}$. Hence, for linearly constrained problems, the same sequence of *ba* pivots may be used at each iteration.

Since we are primarily concerned with large problems, it is essential to try to ensure that the chosen permutation $\boldsymbol{P}$ introduces as little fill-in as possible. Notice that each *ba* pivot requires that we select a row and a column of $\boldsymbol{A}$ and that the selected column of $\boldsymbol{A}$ defines the row of $\boldsymbol{B}$ used.

Without loss of generality, we describe how the first *ba* pivot is determined. The same procedure then may be applied recursively to the Schur-complement of this pivot in $\boldsymbol{K}$ to determine *ba* pivots $2, \dots, m$. Suppose that we consider the permutation

$$\boldsymbol{P}_1^T \boldsymbol{K} \boldsymbol{P}_1 = \left(\begin{array}{cc|cc} \beta & \alpha & \boldsymbol{b}_c^T & \boldsymbol{a}_c^T \\ \alpha & 0 & \boldsymbol{a}_r^T & \boldsymbol{0} \\ \hline \boldsymbol{b}_c & \boldsymbol{a}_r & \boldsymbol{B}_R & \boldsymbol{A}_R^T \\ \boldsymbol{a}_c & \boldsymbol{0} & \boldsymbol{A}_R & \boldsymbol{0} \end{array} \right), \tag{3.1}$$

where $\alpha \neq 0$ and β are scalars, $\boldsymbol{b}_c$ and $\boldsymbol{a}_r$ are $(n-1)$-vectors, $\boldsymbol{a}_c$ is an $(m-1)$-vector, and $\boldsymbol{B}_R$ and $\boldsymbol{A}_R$ are $n-1$ by $n-1$ and $m-1$ by $n-1$ matrices, respectively. Then a simple calculation reveals that the Schur-complement of the *ba* pivot in $\boldsymbol{P}_1^T \boldsymbol{K} \boldsymbol{P}_1$ is

$$\begin{aligned} \boldsymbol{S}_1 = & \begin{pmatrix} \boldsymbol{B}_R & \boldsymbol{A}_R^T \\ \boldsymbol{A}_R & \boldsymbol{0} \end{pmatrix} \\ & - \frac{1}{\alpha^2} \left[\alpha \begin{pmatrix} \boldsymbol{b}_c \\ \boldsymbol{a}_c \end{pmatrix} (\boldsymbol{a}_r^T \quad \boldsymbol{0}) + \alpha \begin{pmatrix} \boldsymbol{a}_r \\ \boldsymbol{0} \end{pmatrix} (\boldsymbol{b}_c^T \quad \boldsymbol{a}_c^T) - \beta \begin{pmatrix} \boldsymbol{a}_r \\ \boldsymbol{0} \end{pmatrix} (\boldsymbol{a}_r^T \quad \boldsymbol{0}) \right]. \end{aligned} \tag{3.2}$$

Notice that no fill-in occurs in the zero, bottom block of $\boldsymbol{S}_1$. Furthermore, suppose that we have picked α so that

$$|\alpha| \geq \upsilon \|\boldsymbol{a}_r\|_\infty \tag{3.3}$$

for some pivot tolerance $0 < \upsilon \leq 1$. Then it follows from (3.2) and (3.3) that the largest entry in the updated $\boldsymbol{A}$ can grow by a factor of at most $1 + 1/\upsilon$, while that in the updated $\boldsymbol{B}$ can grow by at most $(1 + 1/\upsilon)^2$. While these factors may be large, they do provide an upper bound on the overall growth factor after a sequence of *ba* pivots. Indeed, if we perform m *ba* pivots, then it is easy to show that

$$\|\boldsymbol{S}\| \leq (1 + \|\boldsymbol{A}_1^{-1} \boldsymbol{A}_2\|)^2 \|\boldsymbol{B}\|, \tag{3.4}$$

where $\boldsymbol{A} = (\boldsymbol{A}_1 \quad \boldsymbol{A}_2)\boldsymbol{P}$ (see [29]). Hence element growth may be controlled by repeated use of (3.3), and if one of the modified Cholesky methods cited in the introduction is subsequently employed to factorize $\boldsymbol{S}$, the perturbation matrix $\boldsymbol{E}$ will remain bounded in terms of the initial $\boldsymbol{A}$ and $\boldsymbol{B}$.

As the same permutation may be used at *every* iteration of the nonlinear programming algorithm, it is worth investing considerable effort in producing a good ordering.

We now follow [33] by picking the *ba* pivot to modify the least number of coefficients in the remaining $n+m-2$ order block of $\boldsymbol{P}_1^T\boldsymbol{K}\boldsymbol{P}_1$ as the Schur complement is formed. Thus we aim to minimize the number of nonzeros, n_s, in the matrix

$$\alpha\begin{pmatrix}\boldsymbol{b}_c\\ \boldsymbol{a}_c\end{pmatrix}(\boldsymbol{a}_r^T \quad \boldsymbol{0}) + \alpha\begin{pmatrix}\boldsymbol{a}_r\\ \boldsymbol{0}\end{pmatrix}(\boldsymbol{b}_c^T \quad \boldsymbol{a}_c^T) - \beta\begin{pmatrix}\boldsymbol{a}_r\\ \boldsymbol{0}\end{pmatrix}(\boldsymbol{a}_r^T \quad \boldsymbol{0}). \tag{3.5}$$

There are two cases to consider.

Following [12], we call a *ba* pivot a *tile* pivot if $\beta \neq 0$ and an *oxo* pivot when $\beta = 0$. We let $n_z(\boldsymbol{v})$ denote the number of nonzeros in the vector $\boldsymbol{v}$, and $n_o(\boldsymbol{v}, \boldsymbol{w})$ give the number of overlaps (the number of indices i for which both v_i and w_i are nonzero) between the vectors $\boldsymbol{v}$ and $\boldsymbol{w}$.

A simple computation reveals that if we choose an oxo pivot, the number of nonzeros in the matrix (3.5) is

$$n_s = 2n_z(\boldsymbol{a}_r)[n_z(\boldsymbol{a}_c) + n_z(\boldsymbol{b}_c)] - n_o(\boldsymbol{a}_r, \boldsymbol{b}_c)^2, \tag{3.6}$$

while a tile pivot yields

$$n_s \leq 2n_z(\boldsymbol{a}_r)[n_z(\boldsymbol{a}_c) + n_z(\boldsymbol{b}_c)] - n_o(\boldsymbol{a}_r, \boldsymbol{b}_c)^2 + [n_z(\boldsymbol{a}_r) - n_o(\boldsymbol{a}_r, \boldsymbol{b}_c)]^2. \tag{3.7}$$

(The inequality in (3.7) accounts for the possibility of exact cancellation between the terms in (3.5).) Thus if $\boldsymbol{A}$ has rows $\boldsymbol{a}_{r_i}$, $i = 1, \ldots, m$, and columns $\boldsymbol{a}_{c_j}$, $j = 1, \ldots, n$, and $\boldsymbol{B}$ has columns $\boldsymbol{b}_{c_j}$, $j = 1, \ldots, n$, one possibility is to pick the *ba* pivot for which

$$|a_{i,j}| \geq v \max_{1\leq l\leq n} |a_{i,l}| \tag{3.8}$$

and for which

$$\sigma^e_{i,j} = \begin{cases} 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 1) - n_o(\boldsymbol{a}_{r_i}, \boldsymbol{b}_{c_i})^2 & \text{when} \quad b_{j,j} = 0, \\ 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 2) & \\ \quad - (n_o(\boldsymbol{a}_{r_i}, \boldsymbol{b}_{c_i}) - 1)^2 + (n_z(\boldsymbol{a}_{r_i}) - n_o(\boldsymbol{a}_{r_i}, \boldsymbol{h}_{c_j}) - 2)^2 & \text{when} \quad b_{j,j} \neq 0 \end{cases} \tag{3.9}$$

is smallest. However, since computing $n_o(\boldsymbol{a}_{r_i}, \boldsymbol{b}_{c_j})$ may prove to be unacceptably expensive, we follow [12] and overestimate (3.6) and (3.7) by assuming that, except in the pivot rows, there are no overlaps and thus pick the pivot for which

$$\sigma^a_{i,j} = \begin{cases} 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 1) & \text{when} \quad b_{j,j} = 0, \\ 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 2) + (n_z(\boldsymbol{a}_{r_i}) - 1)^2 & \text{when} \quad b_{j,j} \neq 0 \end{cases} \tag{3.10}$$

is smallest. It is relatively straightforward to compute and update the nonzero counts required to use (3.10). Indeed, since $n_z(\boldsymbol{a}_{r_i})$ and $n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j})$ are, respectively, the row and column counts for the matrix

$$\begin{pmatrix}\boldsymbol{B}\\ \boldsymbol{A}\end{pmatrix}, \tag{3.11}$$

the schemes described by [11, section 9.2] are appropriate.

The main disadvantage of the schemes described in this section is that by restricting the pivot order, the fill-in within the Schur complement may prove unacceptable. This will be the case if $\boldsymbol{A}$ contains dense rows since then the Schur complement will almost certainly be completely dense.

A possible way of alleviating this difficulty is to allow all of the pivot types suggested by [21] (see section 2). A drawback is that by allowing b_+ and a_- pivots, we may generate nonzeros (fill-ins) in the "zero" block of (1.7) and thereafter the Markowitz costs (3.9) and (3.10) would be inappropriate. Appropriate Markowitz costs in this case have been suggested by [12]. Preference still should be given to pivots involving $\boldsymbol{A}$-rows if at all possible. A more serious complication is that $\boldsymbol{B}$ will contaminate $\boldsymbol{A}$ if these additional pivot types are allowed, and thus it may no longer be possible to use the same pivot order for a sequence of related problems.

Even if we allow all types of pivots suggested by Forsgren and Murray, there will certainly be cases where the Schur complement becomes unacceptably dense. In the next section, we consider methods that aim to avoid such difficulties.

4. Modified pivoting methods.

Suppose that $\boldsymbol{A}$ contains m_d rows with a large number of nonzeros and that the remaining $m_e \equiv m - m_d$ rows are sparse. Then it is likely that if any of the dense $\boldsymbol{A}$-rows is included in an early pivot, the remaining Schur complement will substantially fill in. It therefore makes sense to avoid pivoting on these rows until the closing stages of the elimination when the Schur complement may be treated as a dense matrix. However, the Forsgren–Murray pivoting rules may conspire to make this impossible.

Let us suppose that we have eliminated the sparse m_e rows of $\boldsymbol{A}$ using Forsgren and Murray's pivoting rules and that the remaining Schur complement $\boldsymbol{S}$ is relatively sparse excepting the m_d $\boldsymbol{A}$-rows. Thus, we may no longer use ba or a_- pivots and are restricted to using b pivots, that is, 1 by 1 pivots occurring in $\boldsymbol{B}$-rows of $\boldsymbol{S}$.

We may continue the factorization of $\boldsymbol{S}$ in two ways. First, we might pick a favorable pivoting sequence for 1 by 1 pivots from the $\boldsymbol{B}$-rows of $\boldsymbol{S}$ purely from a sparsity (fill-in) point of view. Such an approach implicitly assumes that the defined pivot sequence will be acceptable from a numerical viewpoint and is typical of the symbolic analysis phase of the sparse factorization of positive definite matrices (see, for example, [23] or [11]). Having determined the pivot sequence, a numerical (Cholesky or LDL^T) factorization stage proceeds either to completion or until an unacceptable numerical pivot is encountered. In our case, we view any pivot less than a small positive threshold as unacceptable and, slightly abusing notation, shall refer to this pivot as a b_- pivot. If a b_- pivot is encountered, a readjustment of the pivot order may allow the factorization to proceed further, but this is likely to introduce extra fill-in and merely delays us from facing up to an unacceptable pivot.

Second, we might use a combined analysis-factorization strategy, more typical of unsymmetric factorizations (again, see [11]), in which the pivot order is determined as the factorization proceeds and numerically unacceptable pivots moved down the pivot order. Ultimately, once again, if the b-rows/columns of $\boldsymbol{S}$ are insufficiently positive definite, this process will ultimately break down since all remaining b pivots will be b_- pivots. More fill-in may be predicted with this strategy than with the last, and, in the worst case, restrictions on the pivot order may produce an unacceptable level of fill-in within $\boldsymbol{B}$. Our preference is for the first (separate analysis and factorization phases) strategy because the second strategy is likely to prove a considerable overhead in optimization applications when many systems with the same structure are to be solved.

Thus our remaining concern is when the pivot we wish to use, or are forced to use, next is a b_- pivot. We shall refer to this as a *potential breakdown.* At this stage, we are no longer able to take Forsgren–Murray pivots. We now assume that the b_- pivot would be acceptable from the point of view of fill-in. We aim to investigate the consequences of attempting to use this pivot. Remember that our goal is ultimately only to modify $\boldsymbol{B}$ if it fails to be second-order sufficient.

4.1. Implicit modifications. In this section, we consider *always* modifying b_- pivots, but with the knowledge that we can reverse the effect of these modifications at a later stage.

4.1.1. Pseudo modification of b_- pivots. Suppose the uneliminated Schur-complement when we encounter potential breakdown is of the form

$$\begin{pmatrix} \beta_- & \boldsymbol{s}^T \\ \boldsymbol{s} & \boldsymbol{S} \end{pmatrix}, \tag{4.1}$$

where $\beta_- < \sigma_1$ is the candidate b_- pivot. Now suppose that

$$\beta_+ = \max(\sigma_2, ||\boldsymbol{s}||_\infty), \tag{4.2}$$

where $0 < \sigma_1 \leq \sigma_2$, and let

$$\Delta\beta \overset{\text{def}}{=} \beta_+ - \beta_-. \tag{4.3}$$

Then if we could replace β_- by β_+, the latter would be an acceptable pivot. This is precisely what we do, leaving the consequences for later. We call such a modification of $\boldsymbol{B}$ a *pseudo modification* since it is not yet clear that such a modification is actually required to guarantee that $\boldsymbol{B}$ is second-order sufficient.

We propose continuing such a strategy of replacing b_- pivots with acceptable b_+ pivots until the remaining Schur complement is sufficiently small that it may be treated by dense factorization methods. Thereafter, the Forsgren–Murray strategy may be applied to remove the remaining dense $\boldsymbol{A}$-rows, and a modified Cholesky factorization may then be applied to whatever remains. Thus the resulting (modified) Hessian matrix will be second-order sufficient. However, when replacing any b_- pivots with acceptable b_+ pivots, we may have unnecessarily altered elements and must now reverse any damage caused.

Stewart [39] suggested using pseudo modifications as an alternative to pivoting in Gaussian elimination, and he provided a satisfactory error analysis when a single modification is made. Such an analysis may, of course, be used recursively to cover the scheme suggested here. He comments that this strategy may be particularly beneficial for sparse problems, where altering the pivot sequence could lead to undesirable fill-in.

We will have formed a stable factorization of

$$\boldsymbol{N} \overset{\text{def}}{=} \begin{pmatrix} \boldsymbol{B} + \boldsymbol{\Delta B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}, \tag{4.4}$$

where $\boldsymbol{B} = \boldsymbol{H} + \boldsymbol{\Delta H}$, the diagonal matrix $\boldsymbol{\Delta B}$ corresponds to the m_- (say) modified b_- pivots, and the other diagonal matrix $\boldsymbol{\Delta H}$ corresponds to those diagonals changed using the dense modified Cholesky factorization. These later diagonal modifications are necessary to ensure that $\boldsymbol{B}$ is second-order sufficient, while it is not clear that the former modifications are so. Thus we investigate the consequences of removing these modifications.

4.1.2. Countering the effects of pseudo modifications. The system we wish to solve is (1.6), while we have a factorization of (4.4). Suppose that the ith pseudo modification ($1 \le i \le m_-$) occurred in column j_i of $\boldsymbol{H}$ and that the modification was $\Delta\beta_{j_i} > 0$. Let

$$\boldsymbol{\Delta B} = \boldsymbol{V}^T\boldsymbol{V}, \tag{4.5}$$

where $\boldsymbol{V}^T$ is the n by m_- matrix whose columns are $\boldsymbol{v}_i \overset{\text{def}}{=} \sqrt{\Delta\beta_{j_i}}\boldsymbol{e}_{j_i}$. Then we may write (1.6) as

$$\begin{pmatrix} \boldsymbol{B}+\boldsymbol{\Delta B}-\boldsymbol{V}^T\boldsymbol{V} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} \tag{4.6}$$

or equivalently as

$$\begin{pmatrix} \boldsymbol{B}+\boldsymbol{\Delta B} & \boldsymbol{A}^T & \boldsymbol{V}^T \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{V} & \boldsymbol{0} & \boldsymbol{I}_{m_-} \end{pmatrix} \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{\lambda} \\ \boldsymbol{s} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix} \tag{4.7}$$

for some auxiliary vector $\boldsymbol{s}$. A standard block-decomposition of (4.7) shows that we may determine the solution to (1.6) by solving, in order,

$$\begin{pmatrix} \boldsymbol{B}+\boldsymbol{\Delta B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{q} \\ \boldsymbol{\pi} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix}, \tag{4.8}$$

$$\boldsymbol{G}\boldsymbol{s} = \boldsymbol{V}\boldsymbol{q}, \tag{4.9}$$

and

$$\begin{pmatrix} \boldsymbol{B}+\boldsymbol{\Delta B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g}-\boldsymbol{V}^T\boldsymbol{s} \\ \boldsymbol{0} \end{pmatrix}, \tag{4.10}$$

where

$$\boldsymbol{G} = \boldsymbol{I}_{m_-} - (\boldsymbol{V} \quad \boldsymbol{0}) \begin{pmatrix} \boldsymbol{B}+\boldsymbol{\Delta B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{V}^T \\ \boldsymbol{0} \end{pmatrix}. \tag{4.11}$$

Thus to solve (1.6) via the stable factorization (4.4), we also need to form and factorize $\boldsymbol{G}$. This factorization also reveals whether the modification $\boldsymbol{\Delta B}$ is necessary. Thus we have the following proposition.

PROPOSITION 4.1. *$\boldsymbol{B}$ is second-order sufficient if and only if $\boldsymbol{G}$ is positive definite.*

Proof. The result follows from Sylvester's law of inertia (see, e.g., [8]) by considering different block decompositions of

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{B}+\boldsymbol{\Delta B} & \boldsymbol{A}^T & \boldsymbol{V}^T \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{V} & \boldsymbol{0} & \boldsymbol{I}_{m_-} \end{pmatrix}. \tag{4.12}$$

Pivoting on the first two blocks of $\boldsymbol{M}$ and using the definitions (4.4) and (4.11) reveals that

$$\text{In}(\boldsymbol{M}) = \text{In}(\boldsymbol{N}) + \text{In}(\boldsymbol{G}), \tag{4.13}$$

while pivoting on the last block and using the definition (1.7) gives that

$$\operatorname{In}(\boldsymbol{M}) = \operatorname{In}\left(\boldsymbol{I}_{m_-}\right) + \operatorname{In}(\boldsymbol{K}). \tag{4.14}$$

But, since $\boldsymbol{I}_{m_-}$ is clearly positive definite and $\boldsymbol{N}$ is second-order sufficient,

$$\operatorname{In}\left(\boldsymbol{I}_{m_-}\right) = (m_-, 0, 0) \quad \text{and} \quad \operatorname{In}(\boldsymbol{N}) = (n, m, 0). \tag{4.15}$$

Thus combining (4.13)–(4.15), we see that $\operatorname{In}(\boldsymbol{K}) = (n, m, 0)$ if and only if $\operatorname{In}(\boldsymbol{G}) = \operatorname{In}\left(\boldsymbol{G}^{-1}\right) = (m_-, 0, 0)$. The required result then follows since $\boldsymbol{B}$ is second-order sufficient if and only if $\operatorname{In}(\boldsymbol{K}) = (n, m, 0)$ (see [28]). ☐

This result suggests that $\boldsymbol{G}$ should be factorized using a modified Cholesky factorization. If no modification to $\boldsymbol{G}$ is made, $\boldsymbol{B}$ is second-order sufficient. Suppose, on the other hand, that the ith pseudo modification involved column j_i of $\boldsymbol{H}$ and that in the subsequent modified Cholesky factorization the ith diagonal of $\boldsymbol{G}$ was increased by $\Delta\gamma_i$. Then this is equivalent to *actually* modifying $\boldsymbol{B}$ by

$$\left(\frac{\Delta\gamma_i}{1 + \Delta\gamma_i}\right) \boldsymbol{v}_i \boldsymbol{v}_i^T. \tag{4.16}$$

Thus modification of $\boldsymbol{G}$ gives an implicit modification of $\boldsymbol{B}$, and the actual modification is no larger than the pseudo modification.

We note that another possibility is to attempt a Cholesky factorization of $\boldsymbol{G}$ but to retain all the pseudo modifications if the factorization reveals that $\boldsymbol{G}$ is not sufficiently positive definite. This is equivalent to solving (4.8) and then setting $\boldsymbol{p} = \boldsymbol{q}$ and $\boldsymbol{\lambda} = \boldsymbol{\pi}$. However, instinctively we feel that it is better to remove as many pseudo modifications as possible, and thus we prefer to use the modified Cholesky factorization and (4.8)–(4.10).

4.1.3. The pseudo-modification algorithm.

In summary, we propose the following algorithm:

1. Perform a symbolic/numerical analysis and factorization to obtain a good ordering for the complete numerical factorization.

(i) First, construct k_2 2 by 2 ba pivots, using the strategy outlined in section 3 (this involves processing the values of $\boldsymbol{A}$ but not $\boldsymbol{B}$). Stop once the resulting Schur-complement reaches a specified density, i.e., the proportion of its nonzero entries exceeds a given threshold.

(ii) Next, construct k_1 1 by 1 b pivots from the remaining Schur-complement using, for instance, the minimum degree ordering (see, for example, [23] or [11]). Stop once the resulting Schur-complement reaches a specified density.

(iii) The remaining Schur complement will be considered to be dense.

2. Perform the complete numerical factorization.

(i) Perform k_2 2 by 2 sparse eliminations, using the pivots specified in 1(i) above.

(ii) Perform k_1 1 by 1 sparse eliminations, using the pivots specified in 1(ii) above. Modify any b_- pivots encountered to ensure that they are sufficiently positive, using the scheme of, for example, [38]. Record any pseudo modifications made.

(iii) For the remaining dense block, factorize using the scheme of [21] until all the $\boldsymbol{A}$-rows have been eliminated. Thereafter, use a dense modified Cholesky factorization to eliminate the remaining $\boldsymbol{B}$-rows.

(iv) If any pseudo modifications were made in 2(ii) above, form the matrix $\boldsymbol{G}$. Perform a modified Cholesky factorization of $\boldsymbol{G}$.

3. Perform any solves, using the factors obtained in 2 above, by solving the sequence of equations (4.8)–(4.10).

One would normally anticipate performing only a single symbolic/numerical analysis and factorization per minimization, while many complete numerical factorizations and solves might be required. Thus, a good ordering will pay handsome dividends, and one might be prepared to expend considerable effort in step 1.

We should stress that (3.2) indicates that the Schur-complement of the $\boldsymbol{A}$-rows following the elimination of the *ba* pivots is independent of $\boldsymbol{B}$ and thus, since $\boldsymbol{A}$ is independent of $\boldsymbol{x}$, need be formed only once per minimization. This is the only numerical processing involved in the symbolic/numerical analysis and factorization phase.

Notice that the effectiveness of such a scheme depends upon the dimension of $\boldsymbol{G}$. Although the number of pseudo modifications will not be known until the numerical factorization phase, it may be possible to influence this by overriding the pivot selection outlined in section 3 to favor incorporating potentially small or negative elements within the initial *ba* pivots. For instance, if a diagonal of $\boldsymbol{B}$ is known to be negative, it may be worth trying to encourage this element to lie within a *ba* pivot so that it will not be available for pseudo modification in step 1(ii).

4.1.4. Generalizations. When $\boldsymbol{H}$ is strictly positive definite, no pseudo modifications should be necessary. In other cases, it is possible that the dimension of $\boldsymbol{G}$ might be unacceptably high. However, considering (4.1), it is clear that rather in addition to modifying the diagonal β_-, we are free to modify *as many nonzero elements of* $\boldsymbol{s}$ *as we like* without introducing extra fill in the factorization. Thus, if the diagonal of $\boldsymbol{S}$, in a row in which $\boldsymbol{s}$ has a nonzero element, is also small or negative, we should modify the corresponding element of $\boldsymbol{s}$ to increase the offending diagonal. All that we have said in this section about diagonal modifications equally applies for the more general perturbation, but the ith column of the matrix $\boldsymbol{V}$ now contains nonzeros in all positions for which the j_ith pivot column was modified.

Nonetheless, we have to recognize that there are some matrices for which this strategy is inappropriate, as the following example shows.

Example 4.1. Suppose $\boldsymbol{H} = -I_n$ and $\boldsymbol{A} = \boldsymbol{e}^T$, the vector of ones. Then a *ba* pivot is out of the question because the resulting Schur complement would be completely dense. But since each diagonal of $\boldsymbol{H}$ is negative, and there is no connectivity between the diagonals, n pseudo modifications will be required. Unfortunately, $\boldsymbol{G}$ will then be a dense n by n matrix.

Another possibility is to replace $\boldsymbol{G}$ by a simpler matrix as soon as $\boldsymbol{G}$ is found to be indefinite. If we replaced $\boldsymbol{G}$ by $\boldsymbol{G} + \Delta\boldsymbol{G}$, it is straightforward to show that this is equivalent to an actual modification of $\boldsymbol{B}$ by

$$\boldsymbol{V}^T \left(I_{m_-} - (I_{m_-} + \Delta\boldsymbol{G})^{-1} \right) \boldsymbol{V}. \tag{4.17}$$

Thus, provided that $\Delta\boldsymbol{G}$ is positive semidefinite, the actual modification will again be smaller than the pseudo modification. A simple scheme would be to replace $\boldsymbol{G}$ by $\tau\boldsymbol{I}_{m_-}$, where τ is chosen so large that $\tau\boldsymbol{I}_{m_-} - \boldsymbol{G}$ is positive definite whenever $\boldsymbol{G}$ is not positive definite. The advantage of this replacement is that the storage and factorization overheads associated with $\boldsymbol{G}$ may be considerably reduced. The disadvantages are that the size of the actual modification made may be higher than really necessary and that it is not obvious how to choose a satisfactory value for τ.

4.2. Explicit modifications. In the previous sections, we always chose to modify b_- pivots, with the knowledge that we could reverse the effect of the modification at a later stage. As we have seen, it may happen that a considerable number of pseudo modifications will be made and this may be undesirable because of the space and effort required to factorize $\boldsymbol{G}$. In this section, we take the opposite point of view and consider changing b_- pivots *only* when we know it is necessary to modify them. The intention with this alternative is thus to remove, or at least lessen, the need for pseudo modifications.

We now assume that a b_- pivot would be acceptable from the point of view of fill-in *and* stability. This is tantamount to assuming that the pivot is negative and of a reasonable size compared to the remaining entries in its row. We aim to investigate the consequences of using this pivot. Remember that our goal remains only to modify $\boldsymbol{B}$ if it fails to be second-order sufficient.

4.2.1. The condemned submatrix. Recall that we are supposing that we have eliminated the sparse m_e rows of $\boldsymbol{A}$ using Forsgren and Murray's pivoting rules and that $m_d \equiv m - m_e$ $\boldsymbol{A}$-rows remain within $\boldsymbol{S}$. Suppose that we have also eliminated n_e rows of $\boldsymbol{B}$ using Forsgren and Murray's rules.

We pick a nonsingular, square submatrix, the *condemned* submatrix, $\boldsymbol{C}$, of $\boldsymbol{S}$, which contains all the $\boldsymbol{A}$-rows and perhaps some of the $\boldsymbol{B}$-rows (but not the b_- pivot row) of $\boldsymbol{S}$ and has precisely m_d negative eigenvalues. The condemned submatrix will be eliminated last of all and thus any $\boldsymbol{B}$-rows included in $\boldsymbol{C}$ will not be generally available as pivots. The aim is that when only the rows that make up $\boldsymbol{C}$ remain to be eliminated, the uneliminated Schur complement will have precisely m_d negative eigenvalues and hence $\boldsymbol{K}$ will have exactly m negative eigenvalues. The Schur complement $\boldsymbol{S}$ has at least m_d negative eigenvalues. A suitable $\boldsymbol{C}$ may be obtained, for instance, by picking $m_{a_-} \leq \min(n_e - m_e, m_d)$ a_- followed by $m_d - m_{a_-}$ ba pivots. We shall show how such a matrix may be obtained in section 4.2.4.

A factorization of the condemned submatrix should be obtained. As the $\boldsymbol{C}$-rows of $\boldsymbol{S}$ will ultimately invariably be dense, a full matrix factorization is appropriate and, because we may subsequently need to modify the factors, we recommend a QR or LQ factorization. This of course limits the scope of the current proposal because of the size of $\boldsymbol{C}$ which can be accommodated. We note that the dimension of $\boldsymbol{C}$ as constructed above is $2m_d - m_{a_-}$ and hence lies between m_d and $2m_d$.

4.2.2. The consequences of pivoting. With this choice of $\boldsymbol{C}$, $\boldsymbol{S}$ is a permutation of the matrix

$$\left(\begin{array}{cc|c} \beta_- & \boldsymbol{s}_1^T & \boldsymbol{s}_2^T \\ \boldsymbol{s}_1 & \boldsymbol{C} & \boldsymbol{S}_{21}^T \\ \hline \boldsymbol{s}_2 & \boldsymbol{S}_{21} & \boldsymbol{S}_{22} \end{array}\right), \tag{4.18}$$

where $\beta_- < 0$ is the candidate b_- pivot. If we were now to pivot on $\boldsymbol{C}$ instead of β_-, we would have eliminated all m $\boldsymbol{A}$-rows of $\boldsymbol{K}$ and, because of the choice of $\boldsymbol{C}$, the factorized matrix (the submatrix of $\boldsymbol{K}$ corresponding to eliminated rows) would have exactly m negative eigenvalues. Thus $\boldsymbol{B}$ is second-order sufficient if and only if the matrix

$$\begin{pmatrix} \beta_- & \boldsymbol{s}_2^T \\ \boldsymbol{s}_2 & \boldsymbol{S}_{22} \end{pmatrix} - \begin{pmatrix} \boldsymbol{s}_1^T \\ \boldsymbol{S}_{21} \end{pmatrix} \boldsymbol{C}^{-1} \begin{pmatrix} \boldsymbol{s}_1 & \boldsymbol{S}_{21}^T \end{pmatrix} \tag{4.19}$$

is sufficiently positive definite. In particular, if $\beta_- - \boldsymbol{s}_1^T \boldsymbol{C}^{-1} \boldsymbol{s}_1$ is insufficiently positive, $\boldsymbol{B}$ is not second-order sufficient and should be modified.

With this in mind, if

$$\beta_- - \boldsymbol{s}_1^T \boldsymbol{C}^{-1} \boldsymbol{s}_1 < \sigma_1, \tag{4.20}$$

we modify $\boldsymbol{B}$ by replacing β_- by

$$\beta_+ \overset{\text{def}}{=} \max\left(\sigma_2 + \boldsymbol{s}_1^T \boldsymbol{C}^{-1} \boldsymbol{s}_1, ||\boldsymbol{s}||\right), \tag{4.21}$$

where $\boldsymbol{s}^T = \begin{pmatrix} \boldsymbol{s}_1^T & \boldsymbol{s}_2^T \end{pmatrix}$ and, as before, $0 < \sigma_1 \leq \sigma_2$. Conversely, if

$$\beta_- - \boldsymbol{s}_1^T \boldsymbol{C}^{-1} \boldsymbol{s}_1 \geq \sigma_1 > 0, \tag{4.22}$$

it is safe to pivot on β_-. Moreover, although this implies an increase (by one) in the number of negative eigenvalues that have been recorded, the increase is counteracted by a corresponding reduction in the number of available negative eigenvalues in the Schur complement $\boldsymbol{C} - \boldsymbol{s}_1 \boldsymbol{s}_1^T / \beta_-$. This follows directly from the inertial identity

$$\text{In}\left(\boldsymbol{C} - \boldsymbol{s}_1 \boldsymbol{s}_1^T / \beta_-\right) = \text{In}\,(\boldsymbol{C}) + \text{In}\left(\beta_- - \boldsymbol{s}_1^T \boldsymbol{C}^{-1} \boldsymbol{s}_1\right) - \text{In}\,(\beta_-) \tag{4.23}$$

for block decompositions of

$$\begin{pmatrix} \beta_- & \boldsymbol{s}_1^T \\ \boldsymbol{s}_1 & \boldsymbol{C} \end{pmatrix} \tag{4.24}$$

(see, e.g., [8]). We then pivot on the possibly modified value of β_- and replace $\boldsymbol{C}$ by $\boldsymbol{C} - \boldsymbol{s}_1 \boldsymbol{s}_1^T / \beta_-$—we update the matrix factorization to account for this (see [24]). We repeat this procedure until we have eliminated the remaining $\boldsymbol{S}_{22}$ rows, at which point the only noneliminated portion of $\boldsymbol{K}$ is the (updated) matrix $\boldsymbol{C}$.

Alternatively, once it has been determined that $\boldsymbol{B}$ is not second-order sufficient, we might modify all remaining $\boldsymbol{B}$ pivots. One possibility, in the same vein as [38], is to insist that all diagonals are larger than the sum of the absolute values of the (remaining) off-diagonal terms in $\boldsymbol{B}$-rows.

For the case of Example 4.1, the explicit modification scheme considered here would be preferable. The condemned submatrix might be made up from the last row of $\boldsymbol{H}$—indeed, any row of $\boldsymbol{H}$ will do—and the single $\boldsymbol{A}$-row. Examining (4.20) for each diagonal pivot in turn, it follows that $\boldsymbol{H}$ is not second-order sufficient, every pivot will be modified, but no fill-in takes place.

4.2.3. Other pivot types. If the only possible pivots in $\boldsymbol{B}$-rows are zero or small, we may again test them one at a time to see if they might be modified and then used as pivots. If the test reveals that the matrix is not second-order sufficient, we may modify the tested pivot and pivot on it. But if the test is inconclusive, we must either add a pseudo modification (see section 4.1.1) or reject the potential pivot and pass to the next.

It may be better to consider 2 by 2 pivots,

$$\begin{pmatrix} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{pmatrix}, \tag{4.25}$$

arising from the $\boldsymbol{B}$-rows of $\boldsymbol{S}$, especially when the only possible 1 by 1 pivots are small or zero. Then $\boldsymbol{S}$ is a permutation of the matrix

$$\left(\begin{array}{ccc|c} \beta_{11} & \beta_{21} & \boldsymbol{s}_{11}^T & \boldsymbol{s}_{21}^T \\ \beta_{21} & \beta_{22} & \boldsymbol{s}_{12}^T & \boldsymbol{s}_{22}^T \\ \boldsymbol{s}_{11} & \boldsymbol{s}_{12} & \boldsymbol{C} & \boldsymbol{S}_{21} \\ \hline \boldsymbol{s}_{21} & \boldsymbol{s}_{22} & \boldsymbol{S}_{21} & \boldsymbol{S}_{22} \end{array}\right) \tag{4.26}$$

and $\boldsymbol{B}$ is second-order sufficient only if the matrix

$$\left(\begin{array}{cc} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{array}\right) - \left(\begin{array}{c} \boldsymbol{s}_{11}^T \\ \boldsymbol{s}_{12}^T \end{array}\right) \boldsymbol{C}^{-1} (\boldsymbol{s}_{11} \quad \boldsymbol{s}_{12}) \tag{4.27}$$

is sufficiently positive definite. As before, if (4.27) is indefinite, the potential pivot (4.25) should be modified before use. The inertial result

$$\begin{aligned} &\operatorname{In}\left(\boldsymbol{C} - (\boldsymbol{s}_{11}^T \quad \boldsymbol{s}_{12}^T) \left(\begin{array}{cc} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{array}\right)^{-1} \left(\begin{array}{c} \boldsymbol{s}_{11} \\ \boldsymbol{s}_{12} \end{array}\right)\right) \\ &= \operatorname{In}(\boldsymbol{C}) + \operatorname{In}\left(\left(\begin{array}{cc} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{array}\right) - \left(\begin{array}{c} \boldsymbol{s}_{11}^T \\ \boldsymbol{s}_{12}^T \end{array}\right) \boldsymbol{C}^{-1} (\boldsymbol{s}_{11} \quad \boldsymbol{s}_{12})\right) - \operatorname{In}\left(\begin{array}{cc} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{array}\right) \end{aligned} \tag{4.28}$$

once again indicates that the updated $\boldsymbol{C}$ after the pivot inherits the correct number of negative eigenvalues.

4.2.4. Calculating the condemned submatrix. In this section, we consider one way in which the initial condemned submatrix, $\boldsymbol{C}$, may be found. We should stress that the definition of $\boldsymbol{C}$ is by no means unique.

Let the m_d uneliminated $\boldsymbol{A}$-rows in the Schur complement, $\boldsymbol{S}_{ba}$, following the m_e *ba* pivots, be $\boldsymbol{A}_{ba}$. Similarly, let the $n - n_e$ uneliminated $\boldsymbol{B}$-rows and columns in $\boldsymbol{S}_{ba}$ following these *ba* pivots be $\boldsymbol{B}_{ba}$. Furthermore, let

$$\mathcal{O} = \{i_1, i_2, \cdots, i_{n-m_e}\} \tag{4.29}$$

be the ordered pivot sequence for the elimination of $\boldsymbol{B}_{ba}$. Now define the ordered sets

$$\mathcal{P}_1 = \{i_1, i_2, \cdots, i_{n_e-m_e}\} \quad \text{and} \quad \mathcal{P}_2 = \{i_{n-m_e}, i_{n-n_e-1}, \cdots, i_{n_e-m_e+1}\} \tag{4.30}$$

and the ordered set of *preferences*

$$\mathcal{P} = \mathcal{P}_1 \bigcup \mathcal{P}_2. \tag{4.31}$$

For example, suppose that $\boldsymbol{B}$-rows 1, 6, and 4 were involved in *ba* pivots; that the remaining *b* pivots were requested from rows 3, 7, 5, 8, and 2 in order (thus, $\mathcal{O} = \{3, 7, 5, 8, 2\}$); that the pivots from rows 3 and 7 were satisfactory; but that row 5 is a b_- pivot. Then $\mathcal{P}_1 = \{3, 7\}$, $\mathcal{P}_2 = \{2, 8, 5\}$ and $\mathcal{P} = \{3, 7, 2, 8, 5\}$.

Our intention is to find a well-conditioned, nonsingular subset, $\mathcal{C}$, of the columns of $\boldsymbol{A}_{ba}$ by pivoting. The row and column indices of the pivots would provide satisfactory *ba* pivots, if such pivots had not been disqualified on sparsity grounds, for $\boldsymbol{S}_{ba}$. Moreover, the submatrix, $\boldsymbol{C}_{ba}$, formed by taking the rows and columns of $\boldsymbol{S}_{ba}$ corresponding to these 2 by 2 pivots is nonsingular and has precisely m_d negative eigenvalues. If we now consider the subsets $\mathcal{C}_1 = \mathcal{C} \bigcap \mathcal{P}_1$ and $\mathcal{C}_2 = \mathcal{C} \bigcap \mathcal{P}_2$, and pivot on all $\boldsymbol{B}$-rows of $\boldsymbol{C}_{ba}$ whose indices occur in $\mathcal{C}_1$, the remaining Schur-complement still has precisely m_d negative eigenvalues and provides us with a suitable condemned submatrix, $\boldsymbol{C}$. This matrix has the correct inertia as the subblock of $\boldsymbol{C}_{ba}$ corresponding to the $\mathcal{C}_1$ pivots is contained within the subblock of $\boldsymbol{S}_{ba}$ corresponding to the $\mathcal{P}_1$ pivots, and the latter subblock is positive definite since the first $n_e - m_e$ *b* pivots on $\boldsymbol{S}_{ba}$ are positive.

It remains to describe how $\mathcal{C}$ is calculated. We consider how the first element is obtained, the remaining $m_d - 1$ elements following in exactly the same way. The set $\mathcal{C}$ is initially empty and the matrix $\boldsymbol{A}_c$ is initialized as $\boldsymbol{A}_{ba}$. The columns of

$\boldsymbol{A}_c$ are considered one at a time, in the order defined by $\mathcal{P}$. The nonzeros in the current column are examined one at a time. If the entry in row i and column j is that currently under examination and if the stability restriction (3.8) holds (where here $a_{i,j}$ are the entries of $\boldsymbol{A}_c$), column j is added to $\mathcal{C}$ and removed from $\mathcal{P}$, and $\boldsymbol{A}_c$ is reset to the Schur complement of $\boldsymbol{A}_c$ following a pivot on $a_{i,j}$. On the other hand, if (3.8) fails to hold, attention passes to the next nonzero in column j or, if there are no further unexamined entries in the column, to the next column in $\mathcal{P}$.

The order of the preferences $\mathcal{P}$ is chosen deliberately. It first encourages ba pivots whose b_+ component has already been used—the resulting a_- pivot is then available and reduces the possible dimension of $\boldsymbol{C}$. If $\mathcal{P}$ is not entirely made up from $\mathcal{P}_1$, the preference then encourages pivots from those $\boldsymbol{B}$-rows which are last in the elimination ordering—the intention here is that these are unlikely to be good pivots from a fill-in point of view and so it is better to include them in the dense matrix $\boldsymbol{C}$ from the outset.

A disadvantage of the preceding approach is that the order of the set $\mathcal{P}$ depends on at which stage a b_- pivot appears. This may be significant if more than one matrix factorization is required as changes in $\boldsymbol{B}$ may affect $\mathcal{P}$. It may, therefore, be preferable to redefine the preference as

$$\mathcal{P} = \{i_{n-m_e}, i_{n-m_e-1}, \cdots, i_1\}\,. \tag{4.32}$$

This will have the effect that the resulting $\boldsymbol{C}$ will generally be of dimension $2m_d$ but the advantage that the selection of $\mathcal{C}$ is made only once. As before, it will favor including disadvantageous $\boldsymbol{B}$-rows within the condemned submatrix.

5. Numerical experiments.

We are currently planning to implement a code to solve systems of the form (1.6) for the Harwell Subroutine Library. A key requirement is that $\boldsymbol{B}$ should be a second-order sufficient modification of $\boldsymbol{H}$. To test the efficacy of some of the ideas presented in this paper, we report on experiments conducted with a prototype, `KKTSOL`, of this code.

5.1. Implementation details.

We have written a prototype implementation of the algorithm outlined in section 4.1.3. This implicit modification algorithm divides naturally into an analysis, a factorization, and a solve phase.

The analysis phase need be performed only once for a sequence of systems so long as the matrix $\boldsymbol{A}$ and the sparsity structure of $\boldsymbol{H}$ are unchanged. Some numerical processing of the matrix $\boldsymbol{A}$ is performed in the analysis phase. There are several control parameters, in particular the pivot threshold tolerance υ (see (3.8)), the density δ_a of the Schur complement of $\boldsymbol{A}$ during ba pivoting at which the remaining rows of $\boldsymbol{A}$ may be treated as dense, and the density δ_b of the Schur complement of $\boldsymbol{B}$ during b pivoting at which the remaining rows of $\boldsymbol{B}$ may be treated as dense. We choose to switch to full-matrix code as soon as the density of the Schur complement of $\boldsymbol{B}$ during b pivoting exceeds δ_b. However, experience has shown that switching to b pivoting as soon as the density δ_a of the Schur complement of $\boldsymbol{A}$ during ba pivoting exceeds δ_a is sometimes inappropriate since further cheap ba pivots may be possible—remember that it helps to eliminate as many $\boldsymbol{A}$ rows as possible before the b pivoting stage. In particular, if the matrix $\boldsymbol{A}$ is highly structured, many essentially identical ba pivots occur and switching solely on the basis of δ_a may interrupt a promising sequence of pivots. Thus, we actually choose to switch as soon as the density has exceeded its tolerance and the Markowitz cost (3.10) next changes. This heuristic has worked well in our tests. Default values of $\upsilon = 0.0001$, $\delta_a = 0.1$, and $\delta_b = 0.25$ have proved quite reliable. We will indicate the effect of δ_a on the algorithm in section 5.3.

Table 5.1

Problem characteristics. Key: n = number of variables; m = number of equations; nnz A, H *= number of nonzeros in A and H(x);* -eval *= number of negative eigenvalues of reduced Hessian;* nullity *= nullity of K;* convex? *= is the Hessian H(x) positive definite?*

Problem	n	m	nnz A	nnz H	-eval	nullity	convex?
AUG2DCQP	3280	1600	6400	3280	0	0	yes
AUG2DQP	3280	1600	6400	3120	0	0	yes
AUG3DCQP	3873	1000	6546	3873	0	0	yes
AUG3DQP	3873	1000	6546	2673	0	0	yes
BLOCKQP1	2006	1001	9006	1005	1000	0	no
BLOCKQP2	2006	1001	9006	1005	1	0	no
BLOCKQP3	2006	1001	9006	1005	900	1	no
GOULDQP2	699	349	1047	697	0	0	yes
GOULDQP3	699	349	1047	1395	0	0	yes
HAGER2	2001	1000	3000	3001	0	0	yes
KSIP	1021	1001	21002	20	0	0	yes
MINC44	1113	1032	1203	0	0	0	yes
MOSARQP1	1500	600	3530	945	0	0	yes
NCVXQP1	1000	500	1498	3984	438	0	no
NCVXQP2	1000	500	1498	3984	320	0	no
NCVXQP3	1000	500	1498	3984	163	0	no
NCVXQP4	1000	250	749	3984	610	0	no
NCVXQP5	1000	250	749	3984	428	0	no
NCVXQP6	1000	250	749	3984	224	0	no
NCVXQP7	1000	750	2247	3984	250	0	no
NCVXQP8	1000	750	2247	3984	192	0	no
NCVXQP9	1000	750	2247	3984	127	0	no
QPCBOEI1	726	351	3827	384	0	0	yes
QPCBOEI2	305	166	1358	143	0	0	yes
QPCSTAIR	614	356	4003	467	0	0	yes
QPNBOEI1	726	351	3827	384	30	0	no
QPNBOEI2	305	166	1358	143	12	0	no
QPNSTAIR	614	356	4003	467	13	0	no
SOSQP1	2000	1001	4000	1000	0	0	no
UBH1	909	600	2400	303	0	0	yes

During the factorization phase, once a b pivot for which $\beta < \sigma_1$ has been detected, any b pivot which is smaller than the sum of absolute values of the off-diagonal terms in its column is pseudo modified. The pseudo modification is chosen to satisfy (4.2), where $\sigma_1 = 10^{-8}$ and $\sigma_2 = 1$. The rows of $\boldsymbol{A}$ and $\boldsymbol{H}$ which are left over following the sparse ba and b pivoting steps, along with the matrix $\boldsymbol{G}$, are treated as dense matrices. The highest appropriate levels of BLAS (see, for instance, [10]) are used to perform the dense operations wherever possible.

In addition, we have also implemented the explicit modification scheme suggested in section 4.2. This differs from the implicit modification scheme described above in two respects. First, the ordering of the b pivots may be altered to provide a nonsingular condemned matrix, if it is needed. We have implemented the method described in section 4.2.4 using the preference (4.32). Second, during the b phase of the factorization, b_+ pivots are used so long as no b_- pivot is detected. If a b_- pivot appears, the condemned matrix is formed and factorized, and the resulting QR decomposition is used to see if this pivot is acceptable or if it should be modified. Subsequent b_- pivots are treated in the same way, except that now the factors of the condemned matrix are obtained from its predecessor by updating. Slightly modified LAPACK routines (see [1]) are used to compute and update the QR factors.

All our tests were performed on an IBM RISC System/6000 3BT workstation

Table 5.2

*Dependence on the allowed density of A. Key: δ_a = density of updated A at which remaining rows are treated as dense (*no dense *means that no dense rows of A are allowed);* fill-in A, H, dense *= fill-in within A, H, and the final dense block;* dense rows A, H = *number of rows of A and H which are treated as dense;* dense rows G = *number of pseudo modifications made (dimension of G); #* mod = *number of diagonals of H actually modified;* anal., fact., solve = *times for analyze, factorize, and solve (cpu seconds).*

`AUG3DQP`:

	Fill-in			Dense rows			#	Time		
δ_a	A	H	dense	A	H	G	mod	anal.	fact.	solve
0.01	1043	16231	348195	777	57	0	0	.33	3.93	.05
0.05	3161	49458	144991	297	241	0	0	.60	2.30	.03
0.1	3860	67921	171991	198	388	0	0	.71	2.92	.04
0.2	4333	87143	180901	130	471	0	0	.80	3.07	.04
0.5	4773	109644	223446	73	595	0	0	.97	4.22	.04
1.0	5058	138324	245350	47	653	0	0	1.09	5.01	.04
no dense	5806	331483	245350	0	700	0	0	3.99	4.57	.05

`QPNBOEI1`:

	Fill-in			Dense rows			#	Time		
δ_a	A	H	dense	A	H	G	mod	anal.	fact.	solve
0.01	0	3816	73920	347	3	34	34	.02	.84	.02
0.05	4	3844	43365	250	13	31	33	.02	.40	.01
0.1	4	5229	21115	147	25	33	33	.04	.17	.01
0.2	4	5877	13203	98	38	26	33	.04	.11	.00
0.5	4	6577	9180	63	51	21	36	.04	.08	.00
1.0	16	6574	10296	46	78	19	38	.05	.08	.00
no dense	153	71225	70500	0	375	0	32	.92	.77	.01

with 64 megabytes of RAM; the codes are all double-precision Fortran-77, compiled under xlf with -O optimization, and IBM library BLAS are used.

5.2. Test examples. We considered all of the larger quadratic programming examples in the current CUTE test set (see [3]), except that we excluded those which are minor variants (namely `BLOCKQP4`, `BLOCKQP5`, `HAGER4`, `MOSARQP2`, and `SOSQP2`). The characteristics of this test set are described in Table 5.1. All general inequality constraints were converted to equations by the addition of slack variables. To simulate a typical early iteration of a barrier function method, a small value (one-tenth) was added to each diagonal entry of the given Hessian. For each test, the given matrix was factorized and modified if necessary. A right-hand side was then generated so that the required solution is a vector of ones.

5.3. Results. We first illustrate the effect of δ_a, the density of the Schur complement of $\boldsymbol{A}$ during *ba* pivoting at which the remaining rows of $\boldsymbol{A}$ may be treated as dense, on the performance of our algorithm. We consider two examples, `AUG3DQP` and `QPNBOEI1`, from our test set; the first is strictly convex while the reduced Hessian of the second has a few negative eigenvalues. The behavior on these examples is representative of the whole set.

In Table 5.2 we give our results on runs which used the explicit modification algorithm; similar results were observed for the implicit modification scheme. Examining the times taken during the analyze and factorize stages, we see that it is important not to let δ_a be too large, as the remaining Schur complement of $\boldsymbol{K}$ is then too dense. On the other hand, skipping pivoting on rows of $\boldsymbol{A}$ when δ_a is too small is also undesirable since the dimension of the resulting dense matrix is then large. Thus a compromise is necessary and we have found, empirically, that a density of around 10% is reasonable.

Table 5.3

Performance of `MA27` *and* `MA47` *(default settings). Key:* fill-in = *fill-in during factorization;* anal., fact., solve = *times for analyze, factorize, and solve (cpu seconds).*

	`MA27`				`MA47`			
Problem	fill-in	anal.	fact.	solve	fill-in	anal.	fact.	solve
AUG2DCQP	11038	.09	.09	.01	36179	.21	.13	.04
AUG2DQP	11198	.09	.09	.01	36339	.21	.13	.03
AUG3DCQP	10797	.11	.16	.01	50401	.25	.25	.04
AUG3DQP	11997	.10	.16	.01	51601	.25	.25	.04
BLOCKQP1	2015	1.26	.05	.00	16989	1.97	.09	.02
BLOCKQP2	2015	1.27	.06	.00	16989	1.97	.09	.02
BLOCKQP3	2015	1.27	.06	.01	16982	1.97	.08	.02
GOULDQP2	1749	.01	.01	.01	2927	.03	.02	.01
GOULDQP3	2787	.02	.02	.01	3357	.32	.03	.01
HAGER2	9	.04	.03	.01	6004	.07	.04	.02
KSIP	3029	.13	.13	.01	17860	1.94	.12	.02
MINC44	2241	.01	.01	.00	1548	.03	.02	.00
MOSARQP1	6466	.04	.07	.00	29104	.11	.10	.01
NCVXQP1	12539	.13	1.01	.02	273646	1.87	30.26	.06
NCVXQP2	12539	.12	1.01	.02	241150	1.83	29.00	.06
NCVXQP3	12539	.13	.98	.02	272372	1.83	31.53	.06
NCVXQP4	8461	.07	.45	.01	110796	.43	4.00	.02
NCVXQP5	8461	.07	.45	.01	104070	.42	3.53	.03
NCVXQP6	8461	.07	.44	.01	108867	.43	3.69	.02
NCVXQP7	15913	.20	2.60	.02	404465	2.84	61.43	.08
NCVXQP8	15913	.19	2.61	.03	419779	2.89	77.55	.09
NCVXQP9	15913	.20	2.57	.03	406633	2.84	68.19	.09
QPCBOEI1	3886	.08	.03	.00	13333	.16	.05	.00
QPCBOEI2	941	.01	.01	.00	4421	.03	.02	.00
QPCSTAIR	3318	.05	.05	.00	12444	.14	.08	.01
QPNBOEI1	3886	.08	.04	.00	13333	.15	.06	.01
QPNBOEI2	941	.01	.02	.00	4421	.03	.02	.00
QPNSTAIR	3318	.05	.05	.00	12444	.14	.08	.01
SOSQP1	5003	.16	.04	.00	3001	1.25	.06	.01
UBH1	2109	.02	.01	.00	3915	.05	.02	.01

As a yardstick, all of the test examples were factorized using the Harwell codes `MA27` and `MA47`, using default settings. Of course, these codes make no effort to modify $\boldsymbol{H}$ to produce a second-order sufficient $\boldsymbol{B}$; these results are included to indicate the sort of times we consider acceptable for a good factorization and thus the sort of times that we should be aiming for in our modified factorization. The results are given in Table 5.3. We note that although `MA47` was especially designed to cope with augmented systems of the form (1.6), it is often less efficient than the general purpose method `MA27`. In its defense, we sometimes observed that `MA47` obtained accurate solutions to (1.6) while its older sister failed to do so; the `NCVXQP` problems are cases in point.

In Table 5.4, we report on the performance of the implicit modification option from our prototype code, `KKTSOL`, on the test set. For these and subsequent runs, we restrict the total number of dense rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ to be at most 350, although this means that the target densities δ_a or δ_b may be exceeded. We have found that although dense matrices are processed using high-performance BLAS, this restriction often has a beneficial effect on execution times. A value of roughly 350 has been observed empirically to give a good compromise between increased dense storage and the advantages of direct addressing of data.

We make two observations. First, `KKTSOL` performs well in many cases, at least in

Table 5.4

Performance of the implicit modification variant of KKTSOL. *Key:* fill-in A, H, dense = *fill-in within A, H, and the final dense block;* dense rows A, H = *number of rows of A and H which are treated as dense;* dense rows G = *number of pseudo modifications made (dimension of G);* # mod = *number of diagonals of H actually modified;* anal., fact., solve = *times for analyze, factorize, and solve (cpu seconds).*

	Fill-in			Dense rows			#	Time		
Problem	A	H	dense	A	H	G	mod	anal.	fact.	solve
AUG2DCQP	5369	42539	61425	168	182	0	0	.31	.58	.02
AUG2DQP	5369	42539	61425	168	182	0	0	.31	.58	.01
AUG3DCQP	3860	116416	61425	198	152	0	0	.55	1.36	.03
AUG3DQP	3860	116416	61425	198	152	0	0	.55	1.36	.03
BLOCKQP1	0	10998	507528	1	8	998	1003	1.16	16.29	.06
BLOCKQP2	0	10998	45	1	8	0	5	1.17	.04	.00
BLOCKQP3	0	10998	412686	1	8	899	904	1.17	12.46	.06
GOULDQP2	348	1862	253	0	22	0	0	.03	.01	.00
GOULDQP3	348	3479	703	0	37	0	0	.05	.01	.00
HAGER2	0	990	66	0	11	0	0	.06	.01	.00
KSIP	0	190	210	0	20	0	0	.32	.04	.00
MINC44	0	187	253	19	3	0	0	.01	.01	.00
MOSARQP1	0	11703	23005	0	214	0	0	.13	.06	.01
NCVXQP1	1378	13940	208981	73	277	296	515	.64	2.85	.03
NCVXQP2	1378	13940	203841	73	277	288	507	.64	2.78	.02
NCVXQP3	1378	13940	145530	73	277	189	400	.66	1.84	.02
NCVXQP4	288	8730	345696	49	301	481	771	.26	5.37	.03
NCVXQP5	288	8730	278631	49	301	396	682	.27	4.09	.03
NCVXQP6	288	8730	194376	49	301	273	556	.27	2.68	.03
NCVXQP7	2535	16182	80200	75	219	106	259	.90	.86	.02
NCVXQP8	2535	16182	76636	75	219	97	252	.89	.84	.01
NCVXQP9	2535	16182	76245	75	219	96	237	.89	.82	.02
QPCBOEI1	4	5229	14878	147	25	0	0	.03	.08	.00
QPCBOEI2	0	1349	6903	110	7	0	0	.01	.02	.00
QPCSTAIR	687	10021	23653	186	31	0	0	.07	.17	.00
QPNBOEI1	4	5229	21115	147	25	33	33	.04	.17	.01
QPNBOEI2	0	1349	8515	110	7	13	13	.01	.04	.00
QPNSTAIR	687	10021	28441	186	31	21	21	.06	.26	.01
SOSQP1	0	997	10	1	3	0	0	.12	.01	.00
UBH1	1288	5066	9316	97	39	0	0	.06	.03	.00

comparison with MA47. Clearly, restricting the pivot order has some detrimental effect on the fill-in. This is often compensated by our not requiring further pivoting during the factorization, to correct for an inappropriate pivot sequence from the analysis phase, which sometimes hampers MA47.

Second, for the nonconvex problems, a large number of pseudo modifications is required, but many of these later turn into actual modifications. This is especially noticeable for the BLOCK and NCVXQP problems. For many of these problems, significantly more actual modifications are needed than are strictly required to counter the negative eigenvalues in the reduced Hessian, but this is difficult to avoid without having good approximations to their related eigenvectors. BLOCKQP1 and BLOCKQP3 are generalizations of Example 4.1, and, as predicted, the implicit modification scheme is slow precisely because $\boldsymbol{G}$ is large.

In Table 5.5, we consider the performance of the explicit modification variant on the test set. We first note that the alteration of the b pivot order sometimes has a slightly detrimental effect on the analysis times, but this is not significant. However, the main differences are observed on the BLOCK and QPN examples. For the former, the explicit modification scheme clearly helps. Rather than requiring the factorization of

Table 5.5

Performance of the explicit modification variant of KKTSOL. *Key:* fill-in A, H, dense = *fill-in within A, H, and the final dense block;* dense rows A, H = *number of rows of A and H which are treated as dense;* # mod = *number of diagonals of H actually modified;* anal., fact., solve = *times for analyze, factorize, and solve (cpu seconds).*

				Dense					
		Fill-in		rows		#		Time	
Problem	A	H	dense	A	H	mod	anal.	fact.	solve
AUG2DCQP	5369	56098	61425	168	182	0	.75	.89	.02
AUG2DQP	5369	56098	61425	168	182	0	.76	.89	.02
AUG3DCQP	3860	145843	61425	198	152	0	1.93	2.19	.03
AUG3DQP	3860	145843	61425	198	152	0	1.94	2.14	.04
BLOCKQP1	0	10998	45	1	8	1003	1.17	.04	.00
BLOCKQP2	0	10998	45	1	8	5	1.17	.02	.01
BLOCKQP3	0	10998	45	1	8	904	1.17	.04	.01
GOULDQP2	348	1862	253	0	22	0	.03	.00	.01
GOULDQP3	348	3479	703	0	37	0	.04	.01	.00
HAGER2	0	990	66	0	11	0	.05	.01	.00
KSIP	0	190	210	0	20	0	.32	.03	.00
MINC44	0	162	741	19	19	0	.02	.00	.00
MOSARQP1	0	11703	23005	0	214	0	.12	.07	.00
NCVXQP1	1378	13940	61425	73	277	515	.66	3.93	.01
NCVXQP2	1378	13940	61425	73	277	507	.65	3.87	.01
NCVXQP3	1378	13940	61425	73	277	397	.66	3.91	.01
NCVXQP4	288	8730	61425	49	301	769	.28	3.16	.01
NCVXQP5	288	8730	61425	49	301	682	.27	3.14	.01
NCVXQP6	288	8730	61425	49	301	554	.28	3.13	.01
NCVXQP7	2535	16182	43365	75	219	259	.91	1.70	.01
NCVXQP8	2535	16182	43365	75	219	251	.91	1.70	.00
NCVXQP9	2535	16182	43365	75	219	237	.91	1.70	.01
QPCBOEI1	4	4040	43365	147	147	0	.17	.31	.01
QPCBOEI2	0	861	24310	110	110	0	.04	.11	.00
QPCSTAIR	687	1465	61425	186	164	0	.36	.33	.01
QPNBOEI1	4	4040	43365	147	147	41	.18	17.93	.01
QPNBOEI2	0	861	24310	110	110	24	.05	4.06	.00
QPNSTAIR	687	1465	61425	186	164	81	.36	22.02	.01
SOSQP1	0	997	10	1	3	0	.12	.01	.00
UBH1	1288	5041	18915	97	97	0	.10	.06	.00

a large matrix $\boldsymbol{G}$, the factorization and update of a trivial (2 by 2) condemned matrix is performed. For the QPNBOEI1 and QPNSTAIR examples, the roles are reversed. The condemned matrices are now large (of orders 292 and 372, respectively) and the updates quite inefficient. The only difference between the QPC and QPN examples is that the former are (strictly) convex. Thus, the differences in factorization times in Table 5.5 for these examples are purely because the QPN examples form and update their condemned submatrices, while the QPC examples do not need to.

Thus, we see that both the implicit and explicit modification schemes have their advantages and disadvantages. In many cases, these methods are able to compete with the nonmodification methods, and of course the proposals here have extra functionality. However, there are clearly some instances where there is a significant overhead caused by the restriction on the allowable pivots. Thus we must conclude that, so far, we have been only partially successful in fulfilling our stated aims.

6. Conclusions and further comments. In this paper we have shown that a number of modified factorization methods for linearly constrained optimization calculations may be derived, and we have indicated that these techniques hold some promise for large-scale computations. Our next task is to complete our code for the Harwell Subroutine Library. Because this code is of general interest, we intend to release a version, `KKTSOL`, into the public domain. Our ultimate goal is to provide implementations of barrier function-based methods for solving general linearly constrained nonlinear optimization problems with the Harwell Subroutine Library.

An outstanding theoretical question remains. It is relatively straightforward to obtain an upper bound on the (possible) perturbation $\boldsymbol{E}$ made to $\boldsymbol{H}$. Thus so long as $\boldsymbol{x}$ remains bounded and f has a continuous Hessian, $\boldsymbol{B}$ will remain bounded. However, to use the factorization with confidence within a general linearly constrained optimization algorithm, one also needs to ensure that $\boldsymbol{B}$ is *uniformly* second-order sufficient, i.e., the constant σ in (1.5) is independent of the iteration. We have been unable to provide such a bound for the methods suggested here, nor do we believe that it is likely to be easy. This is in contrast to the method of [21], where such a bound is obtained. As we have already mentioned, the difficulty with the Forsgren–Murray approach for sparse problems is that the required pivots may all prove unacceptable from a sparsity viewpoint. It remains an open question as to whether there is a satisfactory method for sparse problems from both the theoretical and practical perspectives.

We have purposely not attempted to derive directions of sufficient negative curvature for such problems (see, for example, [21], [20], and the references contained therein), although algorithms which use them offer stronger convergence guarantees—specifically, convergence to points for which second-order necessary optimality conditions hold. We intend to investigate this possibility for large-scale problems in future.

Acknowledgments. This paper is the successor to the technical report by Arioli et al. [2]. The author is extremely grateful to Mario Arioli, Tony Chan, Iain Duff, Jacek Gondzio, Phil Gill, John Reid, and Bobby Schnabel for stimulating discussions on much of the material contained here, and to four anonymous referees for a number of helpful suggestions. He would also like to thank CERFACS for the environment and facilities which made much of this research possible.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. DuCroz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995.

[2] M. Arioli, T. F. Chan, I. S. Duff, N. I. M. Gould, and J. K. Reid, *Computing a Search Direction for Large-Scale Linearly Constrained Nonlinear Optimization Calculations*, Tech. Rep. RAL-93-066, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1993.

[3] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *CUTE: Constrained and unconstrained testing environment*, ACM Trans. Math. Software, 21 (1995), pp. 123–160.

[4] J. R. Bunch and L. C. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear equations*, Math. Comp., 31 (1977), pp. 163–179.

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

[6] T. J. Carpenter, I. J. Lustig, J. M. Mulvey, and D. F. Shanno, *Higher-order predictor-corrector interior point methods with application to quadratic objectives*, SIAM J. Optim., 3 (1993), pp. 696–725.

[7] A. R. Conn, N. I. M. Gould, and Ph. L. Toint, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer Ser. Comput. Math. 17, Springer-Verlag, Heidelberg, Berlin, New York, 1992.

[8] R. W. Cottle, *Manifestations of the Schur complement*, Linear Algebra Appl., 8 (1974), pp. 189–211.

[9] J. E. Dennis and R. B. Schnabel, *A view of unconstrained optimization*, in Handbook of Operations Research and Management Science, vol. 1. Optimization, G. L. Nemhauser, A. H. G. Rinnooy Kan, and M. J. Todd, eds., North–Holland, Amsterdam, 1989, pp. 1–72.

[10] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, PA, 1991.

[11] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct methods for sparse matrices*, Oxford University Press, Oxford, UK, 1986.

[12] I. S. Duff, N. I. M. Gould, J. K. Reid, J. A. Scott, and K. Turner, *The factorization of sparse symmetric indefinite matrices*, IMA J. Numer. Anal., 11 (1991), pp. 181–204.

[13] I. S. Duff and J. K. Reid, *MA27 : A Set of Fortran Subroutines for Solving Sparse Symmetric Sets of Linear Equations*, Report R-10533, AERE Harwell Laboratory, Harwell, UK, 1982.

[14] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[15] I. S. Duff and J. K. Reid, *Exploiting zeros on the diagonal in the direct solution of indefinite sparse symmetric systems,* ACM Trans. Math. Software, 22 (1996), pp. 227–257.

[16] I. S. Duff, J. K. Reid, N. Munksgaard, and H. B. Neilsen, *Direct solution of sets of linear equations whose matrix is sparse, symmetric and indefinite*, J. Inst. Math. Appl., 23 (1979), pp. 235–250.

[17] A. V. Fiacco and G. P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, Chicester, New York, 1968; reprinted as Classics Appl. Math. 4, SIAM, Philadelphia, PA, 1990.

[18] R. Fletcher, *Factorizing symmetric indefinite matrices*, Linear Algebra Appl., 14 (1976), pp. 257–272.

[19] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., John Wiley, Chicester, New York, 1987.

[20] A. Forsgren, P. E. Gill, and W. Murray, *Computing modified Newton directions using a partial Cholesky factorization*, SIAM J. Sci. Comput., 16 (1995), pp. 139–150.

[21] A. L. Forsgren and W. Murray, *Newton methods for large-scale linear equality-constrained minimization*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 560–587.

[22] R. Fourer and S. Mehrotra, *Solving symmetrical indefinite systems in an interior-point method for linear programming*, Math. Programming, 62 (1993), pp. 15–39.

[23] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

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

[25] P. E. Gill and W. Murray, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Programming, 7 (1974), pp. 311–350.

[26] P. E. Gill, W. Murray, D. B. Ponceléon, and M. A. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292–311.

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

[28] N. I. M. Gould, *On practical conditions for the existence and uniqueness of solutions to the general equality quadratic-programming problem*, Math. Programming, 32 (1985), pp. 90–99.

[29] N. I. M. Gould, *Constructing Appropriate Models for Large-Scale, Linearly-Constrained, Nonconvex, Nonlinear, Optimization Algorithms*, Tech. Rep. RAL-TR 95-037, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1995.

[30] Harwell Subroutine Library, *A Catalogue of Subroutines (Release* 10*)*, Advanced Computing Department, Harwell Laboratory, Harwell, UK, 1990.

[31] I. J. Lustig, R. E. Marsten, and D. F. Shanno, *Computational experience with a primal-dual interior point method for linear programming*, Linear Algebra Appl., 152 (1991), pp. 191–222.

[32] I. J. Lustig, R. E. Marsten, and D. F. Shanno, *On implementing Mehrotra's predictor-corrector interior-point method for linear programming*, SIAM J. Optim., 2 (1992), pp. 435–449.

[33] H. M. Markowitz, *The elimination form of the inverse and its application to linear programming*, Management Sci., 3 (1957), pp. 255–269.

[34] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM J. Optim., 2 (1992), pp. 575–601.

[35] J. J. Moré and D. C. Sorensen, *On the use of directions of negative curvature in a modified Newton method*, Math. Programming, 16 (1979), pp. 1–20.

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

[37] T. Schlick, *Modified Cholesky factorizations for sparse preconditioners*, SIAM J. Sci. Comput., 14 (1993), pp. 424–445.

[38] R. B. Schnabel and E. Eskow, *A new modified Cholesky factorization*, SIAM J. Sci. Statist. Comput., 11 (1991), pp. 1136–1158.

[39] G. W. Stewart, *Modifying pivot elements in gaussian elimination*, Math. Comp., 28 (1967), pp. 537–542.

[40] R. J. Vanderbei and T. J. Carpenter, *Symmetrical indefinite systems for interior point methods*, Math. Programming, 58 (1993), pp. 1–32.