Technical Report
RAL-TR-95-037

Constructing Appropriate Models for Large-scale, Linearly-constrained, Nonconvex, Nonlinear Optimization Algorithms

N I M Gould

August 1995

COUNCIL FOR THE CENTRAL LABORATORY OF THE RESEARCH COUNCILS

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

The Central Laboratory for the Research Councils
Library and Information Services
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire
OX11 0QX
Tel: 01235 445384 Fax: 01235 446403
E-mail library@rl.ac.uk

ISSN 1358-6254

RAL-TR-95-037

Constructing appropriate models for large-scale, linearly-constrained, nonconvex, nonlinear optimization algorithms

by

Nicholas Ian Mark Gould[1]

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 which 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 not adversely affecting the convergence of Newton's method nor incurring a significant computational overhead. Numerical evidence to support the theoretical developments is provided.

Keywords: large-scale problems, linearly constrained optimization, modified matrix factorizations, second-order optimality conditions

AMS(MOS) subject classifications: 65F05, 65F10, 65F15, 65F50, 65K05, 90C30.

Running title: Large-Scale Linearly Constrained Optimization

---

[1] Current reports available by anonymous ftp from joyous-gard.cc.rl.ac.uk (internet 130.246.9.91) in the directory "pub/reports".

Computing and Information Systems Department,
Atlas Centre, Rutherford Appleton Laboratory,
Oxfordshire OX11 0QX, England.

August 16, 1995.

1 Introduction

In this paper, we consider algebraic issues which arise when attempting to solve a smooth linearly-constrained, nonlinear-optimization problem,

$$\underset{\boldsymbol{x}\in\Re^n}{\text{minimize}} \quad f(\boldsymbol{x}) \tag{1.1}$$

subject to the m general, linearly-independent, linear equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}. \tag{1.2}$$

Many of these issues particularly arise when f is non-convex, and thus we make no assumption on f other than that it is twice continuously differentiable; we denote its gradient and Hessian matrix by $\boldsymbol{g}(\boldsymbol{x}) \stackrel{\text{def}}{=} \nabla_x f(\boldsymbol{x})$ and $\boldsymbol{H}(\boldsymbol{x}) \stackrel{\text{def}}{=} \nabla_{xx} f(\boldsymbol{x})$ respectively. We shall furthermore assume that $\boldsymbol{H}(\boldsymbol{x})$ is available, and that we wish to exploit this curvature information. We contend that, contrary to popular belief, exact second derivatives are frequently available, but all too frequently ignored.

Problems of the form (1.1)–(1.2) sometimes arise in their own right (see, for instance, the collection by Bongartz, Conn, Gould and Toint, 1995). However, such problems more commonly occur as subproblems within more general nonlinear programming calculations (see, for example, Murtagh and Saunders, 1978, Gill, Murray and Wright, 1981, and Conn, Gould, Sartenaer and Toint, 1995). Although these latter subproblems may at first appear to have a different form from (1.1)–(1.2), it is often possible to convert them to such a form so that algorithms discussed in this paper are appropriate.

We shall not directly consider inequality constraints in this paper. However the methods are still relevant in this case, as many inequality constrained problems may be solved as a sequence of equality constrained ones. For instance, suppose that, in addition to the linear constraints (1.2), the variables are also required to satisfy the simple bound constraints

$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}. \tag{1.3}$$

These inequalities should be understood componentwise, and any (or all) of the components of $\boldsymbol{l}$ and $\boldsymbol{u}$ may be infinite. There are then two main classes of methods for solving the problem (1.1)–(1.3): active set methods and barrier methods.

In active set methods (see, for instance, Gill et al., 1981, Section 5.2), a subset of the simple bound constraints are treated as equations, while the remainder are temporarily discarded. The solution to the resulting equality constrained problem is then sought. This may lead to the solution of the original problem, or, more frequently, to a revision of the active set. The code MINOS (see, Murtagh and Saunders, 1987) is an archetypical example of this class of method.

In barrier methods (see, for instance, Fiacco and McCormick, 1968), problem (1.1)–(1.3) is solved by a sequential minimization of a *barrier function* (BF)

$$f_b(\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{s}) = f(\boldsymbol{x}) + \sum_{i=1}^{n} \beta(x_i, l_i, u_i, w_i^l, s_i^l, w_i^u, s_i^u), \tag{1.4}$$

subject to the constraints (1.2), where the weights $\boldsymbol{w} = (\boldsymbol{w}^l, \boldsymbol{w}^u)$ and shifts $\boldsymbol{s} = (\boldsymbol{s}^l, \boldsymbol{s}^u)$ are used in the definition of barrier terms, β. Each barrier term β has the form

$$\beta(x_i, l_i, u_i, w_i^l, s_i^l, w_i^u, s_i^u) = w_i^l \phi(x_i - l_i + s_i^l) + w_i^u \phi(u_i - x_i + s_i^u) \qquad (1.5)$$

where $\phi(\alpha)$ is C^2 for all $\alpha > 0$ and has a singularity at the origin – examples are the logarithmic function $\phi(\alpha) = -\log(\alpha)$ and the reciprocal function $\phi(\alpha) = \alpha^{-p}$ for any $p > 0$. The shifts are all nonnegative, the weights for infinite bounds are zero while those corresponding to finite bounds are strictly positive. Particular examples are the traditional barrier functions proposed by Frisch (1955) and popularized by Fiacco and McCormick (1968) ($s_i = 0$, $w_i = \mu$, for some parameter $\mu > 0$), the modified barrier function of Jittorntrum and Osborne (1980) ($s_i = 0$, $w_i = \lambda_i$, for appropriate Lagrange multiplier estimates λ_i) and the shifted/modified/Lagrangian barrier functions of Gill, Murray, Saunders and Wright (1988), Polyak (1992) and Conn, Gould and Toint (1992*a*, 1992*c*) ($s_i > 0$, $w_i = \lambda_i s_i$, for appropriate Lagrange multiplier estimates λ_i). Strong convergence results are available for general nonlinear programming problems for these methods, and, as such they are well suited for solving the inequality problem of interest here.

Our intention here is not to analyse particular algorithms, rather to develop good techniques for solving the inevitable systems of linear equations which arise at each iteration. However, as the problem under consideration may not be convex, careful attention must be paid to the model on which the iteration is based and this clearly affects the way in which we solve it.

The inertia of the generic symmetric matrix $\boldsymbol{M}$ will be denoted by

$$\text{In}\,(\boldsymbol{M}) = (m_+, m_-, m_0), \qquad (1.6)$$

where m_+, m_- and m_0 are, respectively, the numbers of positive, negative and zero eigenvalues of $\boldsymbol{M}$. We shall denote the n by n identity matrix by $\boldsymbol{I}_n$, or $\boldsymbol{I}$ when the dimension is clear from the context. Finally, $\boldsymbol{e}_i$ will be the i-th column of $\boldsymbol{I}$.

The paper is organized as follows. In Section 2, we discuss general issues of convergence for algorithms for linearly constrained optimization, and outline the generic systems of equations which arise. In Section 3, we review existing direct methods for solving these systems and indicate how and why it is often necessary to modify the relevant coefficient matrix. In Sections 4 and 5, we describe techniques which are particularly attractive when the systems are large and sparse, and indicate in Section 6 how the ideas presented in this paper behave in practice, using a prototype code. We comment in Section 7 on other alternatives. and conclude in Section 8.

2 General solution methods

We consider iterative methods for finding a local solution to the problem (1.1) subject to the linear constraints (1.2). We let x be the current iterative approximation and consider how an improved iterate x^+ may be found.

2.1 General considerations

Suppose that x satisfies the general constraints (1.2). A typical "linesearch" type iteration would

- compute a *search direction* p for the objective function, which satisfies the constraints
$$Ap = 0 \tag{2.1}$$
and for which $p^T g(x)$ is "sufficiently" negative, and;
- perform a *linesearch* for the objective function along the search direction to obtain $x^+ = x + \alpha p$, for some suitable *stepsize* $\alpha > 0$, such that $f(x^+)$ is "sufficiently" smaller than $f(x)$.

The linesearch should ensure that one of the classical sets of "sufficient decrease" conditions is satisfied at x^+ (see, for instance, Dennis and Schnabel, 1983, Section 6.3). Furthermore, whenever possible, advantage should be taken of the "shape" of the linesearch function to derive an efficient algorithm (see, for example, Lasdon, Fox and Ratner, 1973 and Murray and Wright, 1994).

If the Hessian matrix of the objective function is positive definite, a search direction close to that given by Newton's method is desirable. However, in general the Hessian may be indefinite and we need to take precautions to ensure that the search direction is a sufficiently good descent direction.

The *(quasi-)Newton search direction*, p_n, and the corresponding Lagrange multiplier estimates, λ_n, satisfy the equations

$$\begin{pmatrix} B & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} p_n \\ \lambda_n \end{pmatrix} = \begin{pmatrix} -g(x) \\ 0 \end{pmatrix}, \tag{2.2}$$

where B is a suitable symmetric matrix. In order to provoke rapid asymptotic convergence, it is desirable that B should be a reasonable approximation to the Hessian, $H(x)$.

We say that a matrix B is *second-order sufficient* if the condition

$$\begin{aligned} p^T B p \geq \sigma p^T p \quad &\text{for some constant} \quad \sigma > 0 \\ \text{and all} \quad p \quad &\text{satisfying} \quad Ap = 0 \end{aligned} \tag{2.3}$$

is satisfied. We aim to use a second-order sufficient matrix B as our Hessian approximation in (2.2). In this case, equations (2.2) are both necessary and sufficient for the solution p_n to be the global solution of the (possibly nonconvex) equality constrained quadratic programming problem

$$\underset{p \in \Re^n}{\text{minimize}} \quad \tfrac{1}{2} p^T B p + p^T g(x) \quad \text{subject to} \quad Ap = 0 \tag{2.4}$$

(see, for example, Gill et al., 1981).

So long as B is second-order sufficient and the matrix

$$E \stackrel{\text{def}}{=} B - H(x) \tag{2.5}$$

is bounded, it is straightforward to show that the gradient of the Lagrangian function, $f(x) + \lambda^T(Ax - b)$, converges to zero, for any bounded sequence of iterates $x^{(k)}$ generated by the search direction obtained from (2.2) and a "sufficient" linesearch. For, picking any full-rank n by $n - m$ matrix Z such that $AZ = 0$ (see, for instance, Gill et al., 1981), Gill and Murray (1974) show that the minimization of (1.1) subject to (1.2) is equivalent to the unconstrained minimization of $f(x + Zp_z)$ with respect to the variables p_z so long as x satisfies (1.2). Moreover, equations (2.2) are equivalent to the (quasi-)Newton equations

$$Z^T B^{(k)} Z p_z^{(k)} = -Z^T g(x^{(k)}). \tag{2.6}$$

We note that $Z^T B^{(k)} Z$ is positive definite whenever $B^{(k)}$ is second-order sufficient. Typical global convergence — that is, convergence to a first-order stationary point — results for unconstrained optimization (see, for instance, Dennis and Schnabel, 1983, Theorem 6.3.3), then imply that

$$\lim_{k \to \infty} Z^T g(x^{(k)}) = 0 \tag{2.7}$$

provided the condition number of the *reduced Hessian*, $Z^T B^{(k)} Z$, is bounded. This latter condition is satisfied whenever $B^{(k)}$ is second-order sufficient and $Z^T E^{(k)} Z$ is bounded. As Z is of full rank, we finally infer from (2.7) that

$$\lim_{k \to \infty} g(x^{(k)}) + A^T \lambda^{(k)} = 0 \tag{2.8}$$

for some sequence of Lagrange multipliers $\{\lambda^{(k)}\}$.

2.2 Computing a search direction

The most obvious way of determining a search direction that is close to the Newton direction is to use a matrix factorization (a direct method) to compute the solution to (2.2). Such a factorization should take into account the symmetric but indefinite structure of the coefficient matrix.

A second possibility is to observe that the reduced-variable (quasi-)Newton equations (2.6) provide the global solution to the unconstrained quadratic minimization problem

$$\underset{p_z \in \Re^{n-m}}{\text{minimize}} \;\; q(p_z) \stackrel{\text{def}}{=} \tfrac{1}{2} p_z^T Z^T B Z p_z + p_z^T Z^T g(x) \tag{2.9}$$

whenever B is second-order sufficient. While it is certainly easy to imagine solving (2.6) using a factorization of the reduced Hessian, this may be inappropriate if $n - m$ is large and $Z^T B Z$ dense. In this case, it may still be possible to calculate an approximation to the Newton direction by applying an *iterative* method so

long as it is feasible to form matrix-vector products of the form $Z^T B Z v$. This may well be the case if Z is carefully chosen to allow rapid computation of the subsidiary products Zv, $B(Zv)$ and $Z^T(BZv)$. Although an optimally sparse representation of Z may be computationally expensive (see Coleman and Pothen, 1984, and Gilbert and Heath, 1987), good choices may often be obtained (see, for instance, Murtagh and Saunders, 1978, Coleman and Pothen, 1987, and Stern and Vavasis, 1993).

When considering iterative methods, we choose to restrict our attention to the method of conjugate gradients. The application of the method to linearly constrained problems has been suggested by a number of authors (see, for instance Gill et al., 1981, Section 5.6.2.1). Moreover, as the conjugate-gradient method produces a monotonically decreasing sequence of values of the quadratic objective function, $q(p_z)$, it is possible to *truncate* the conjugate gradient iteration at a sub-optimal point for (2.9) while still maintaining a fast asymptotic convergence rate (see, Dembo, Eisenstat and Steihaug, 1982)).

We shall concentrate on direct methods in this paper, although we shall make comments on the obvious connections to iterative methods as we proceed. Work is in progress on general purpose iterative methods for large-scale nonconvex problems.

2.3 Aims of the paper

We state our aims as follows. We wish to:

- determine a matrix $B = H(x) + E$ so that (2.3) is satisfied and such that the perturbation $E = 0$ whenever $H(x)$ satisfies (2.3);
- obtain E without incurring undue overheads above those normally considered acceptable when calculating the search direction;
- ensure that $\|E\|$ is bounded relative to $\max(\|A\|, \|H(x)\|)$ — provided that $\{x\}$ remains bounded, this will ensure that B is uniformly bounded;
- use the sparsity and structure of (2.2) to derive a sparse factorization or effective iterative procedure; and
- 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.

For brevity, we shall write g and H for $g(x)$ and $H(x)$, respectively, where no confusion can arise. We stress that, throughout the paper, the matrix B denotes a second-order sufficient approximation to H which will actually be H whenever the latter is itself second-order sufficient.

3 Hessian approximations

If we wish our linearly-constrained optimization algorithm to inherit the fast asymptotic convergence rate of Newton's method, we might try to ensure that B converges to the true Hessian matrix of the objective function. In particular, we may choose B to be H whenever the latter is second-order sufficient. The difficulty is, of course, in achieving this goal in an efficient manner.

3.1 Hessian approximations and unconstrained optimization

Before considering the linearly constrained case, we first briefly review methods which have been used successfully in *unconstrained* optimization. We do this for two reasons. Firstly, such methods may be viewed as prototypes for possible extensions to the constrained case. Secondly, as we have already mentioned in Section 2.2, an implicit elimination of the constraints results in an unconstrained problem.

We recall the precedent set from research in unconstrained optimization by Greenstadt (1967), Gill and Murray (1974) and Schnabel and Eskow (1991) amongst others, in which a modification to the exact second derivative matrix in a Newton method is only made when the exact derivatives are insufficiently positive definite. When there are no constraints present, (2.3) is equivalent to requiring that B be *sufficiently positive definite*, that is, that

$$p^T B p \geq \sigma p^T p \quad \text{for some constant} \quad \sigma > 0 \quad \text{and all} \quad p. \tag{3.1}$$

In particular, Gill and Murray (1974) and Schnabel and Eskow (1991) suggest *modified-Cholesky factorization* methods which decide whether, and by how much, the diagonals of the Hessian matrix need to be modified as the Cholesky factorization of the matrix proceeds. The factors are then used to solve the resulting modified Newton equations. Both pairs of authors are very careful to ensure that the modifications made are bounded and that no modification ensues when the Hessian is sufficiently positive definite. Extensions to large-scale unconstrained and bound constrained optimization, using sparse factorizations, have been proposed by Gill, Murray, Ponceléon and Saunders (1992), Conn, Gould and Toint (1992*b*, Chapter 3), and Schlick (1993). Iterative methods, in which modifications to H are made during the course of the conjugate-gradient algorithm have been proposed by Nash (1984) and Arioli, Chan, Duff, Gould and Reid (1993).

We should also mention the philosophically-different but mechanically-similar class of ℓ_2 trust-region methods (see, e.g., Hebden, 1973, Moré, 1978, Sorensen, 1980 and Gay, 1981). Here perturbations of the form $E = \mu I$ may be made to H for some appropriate scalar μ in order to make B positive semi-definite. We note the difference of approach between these methods and the modified Cholesky methods, in that perturbations to the Hessian may only be made in certain low-rank subspaces with the modification methods while any perturbation in an ℓ_2 trust-region method produces a rank n change to H.

A thorough survey of modified Newton methods for unconstrained optimization is given by Dennis and Schnabel (1989).

3.2 Hessian approximations and constrained optimization

As far as we are aware, the only serious attempt to generalize the methods of Section 3.1 to solve general, large-scale, linearly-constrained optimization problems is that by Forsgren and Murray (1993). All other methods we are aware of are either only really appropriate for small-scale calculations, as they disregard problem structure (see Fletcher, 1987, Section 11.1, or Gill et al., 1981, 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, Murtagh and Saunders, 1978).

Firstly, note that the coefficient matrix,

$$\boldsymbol{K} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}, \tag{3.2}$$

of (2.2) is inevitably indefinite — it must have at least m positive and m negative eigenvalues. Gould (1985) showed that $\boldsymbol{B}$ is a second-order sufficient matrix if and only if $\boldsymbol{K}$ has precisely m negative and n positive eigenvalues. Thus any matrix factorization of (3.2) must be capable of handling indefinite matrices. Moreover, in order 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 Bunch and Parlett (1971) and later improved by Bunch and Kaufman (1977) and Fletcher (1976) in the dense case and Duff, Reid, Munksgaard and Neilsen (1979) and Duff and Reid (1983) in the sparse case. Here a symmetric matrix $\boldsymbol{K}$ is decomposed as

$$\boldsymbol{K} = \boldsymbol{PLDL}^T\boldsymbol{P}^T, \tag{3.3}$$

where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{L}$ unit lower triangular and $\boldsymbol{D}$ 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. Notice that the inertia of $\boldsymbol{K}$ is trivially obtained by summing the inertia of the pivots.

As we are particularly concerned with the large-scale case in this paper, 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 Harwell Subroutine Library (1990) implementation, `MA27`, (Duff and Reid, 1982) of the Duff-Reid variant sometimes proves inefficient when applied to matrices of the form (3.2) 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 analyse 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 Duff, Gould, Reid, Scott and Turner (1991), and form the basis for a recent Harwell Subroutine Library code `MA47` (Duff and Reid, 1995).

In the special case when f is separable, $\boldsymbol{H}$ will be diagonal. In particular, when f is also 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

$AH^{-1}A^T$ is feasible. Indeed, this approach is fundamental to many interior point methods for linear programming (see, for example, Mehrotra, 1992, Lustig, Marsten and Shanno, 1991, Lustig, Marsten and Shanno, 1992, or Carpenter, Lustig, Mulvey and Shanno, 1993). However, as such an approach is merely the restriction of a particular pivot order applied to (3.2), and as it is less appealing when H is not diagonal, Fourer and Mehrotra (1993) have suggested methods for solving (3.2) using more general pivot sequences for linear programming problems and Vanderbei and Carpenter (1993) do the same for general problems.

If B is known *a priori* to be second-order sufficient, as for instance would be the case if $f(x)$ were convex, we wholeheartedly recommend the use of `MA27`, `MA47` or the procedure within `loqo` (Vanderbei and Carpenter (1993)) to solve (2.2). When there is a chance that B may not be second-order sufficient, alternatives to blindly solving (2.2) must be sought.

3.3 Forsgren and Murray's sufficient pivoting conditions

We say that the first n rows of K are B-rows, and the remaining m rows are A-rows. Forsgren and Murray (1993) show that, if the pivots are restricted to be of certain types until all of the A-rows of K have been eliminated, the remaining un-eliminated (Schur-complement) matrix, S, is sufficiently positive definite if and only if B is second-order sufficient.

Until all A-rows of K have been exhausted, Forsgren and Murray only allow the following types of pivots:

b_+ pivots: strictly positive 1 by 1 pivots occurring in B-rows of K.

a_- pivots: strictly negative 1 by 1 pivots occurring in A-rows of K.

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

They 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 L and limit any growth in S. The motivation behind this choice of pivot is simply that if i A-rows have been eliminated, the factorized matrix has exactly i negative eigenvalues. Thus, when all A-rows have been eliminated, the factorized matrix has precisely m negative eigenvalues and hence any further negative eigenvalues in S can only occur because B is not second-order sufficient.

Once S has been determined, Forsgren and Murray form a partial LDL^T factorization of it, stopping if a pivot is insufficiently positive. If the factorization runs to completion, B must be second-order sufficient. The (quasi-)Newton equations (2.2) are subsequently solved using the factorization. If an insufficiently positive pivot is encountered, a search arc is obtained as a nonlinear combination of a search direction derived from the partial factorization and a direction of negative curvature from the remaining unfactorized part.

An obvious variation is, instead, to form a modified Cholesky factorization of S. If no modification is performed, the true Hessian H must be second-order sufficient. Otherwise, a suitable perturbation E will have been produced. In either case, the Newton equations (2.2) 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 may all lead to considerable fill-in. Nonetheless, we shall consider a number of variations of this scheme.

4 Methods using ba pivots

In this section, we consider a scheme which uses a restricted version of Forsgren and Murray's (1993) pivoting rules. Specifically, we consider what happens if we choose the first m pivots to be *ba* pivots.

4.1 Algebraic considerations

Let us first suppose that we have chosen a pivot sequence so that the first m pivots are *ba* pivots. Algebraically, this is equivalent to constructing a permutation $\boldsymbol{P}$ for which

$$\boldsymbol{P}^T\boldsymbol{K}\boldsymbol{P} = \begin{pmatrix} \boldsymbol{B}_{11} & \boldsymbol{A}_1^T & \boldsymbol{B}_{21}^T \\ \boldsymbol{A}_1 & \boldsymbol{0} & \boldsymbol{A}_2 \\ \boldsymbol{B}_{21} & \boldsymbol{A}_2^T & \boldsymbol{B}_{22} \end{pmatrix}, \tag{4.1}$$

where $\boldsymbol{B}_{11}$ and $\boldsymbol{A}_1$ are both square matrices of order m, $\boldsymbol{A}_1$ is non-singular and $\boldsymbol{P}$ may be partitioned as

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{P}_{11} & \boldsymbol{0} & \boldsymbol{P}_{13} \\ \boldsymbol{P}_{21} & \boldsymbol{0} & \boldsymbol{P}_{23} \\ \boldsymbol{0} & \boldsymbol{P}_{32} & \boldsymbol{0} \end{pmatrix}. \tag{4.2}$$

(The actual pivot sequence would interlace the i-th and $m+i$-th rows of $\boldsymbol{P}$ for $i = 1, \cdots, m$). This permutation implies a corresponding partitioning

$$\begin{pmatrix} \boldsymbol{p} \\ \boldsymbol{\lambda} \end{pmatrix} = \boldsymbol{P}\begin{pmatrix} \boldsymbol{p}_1 \\ \boldsymbol{\lambda} \\ \boldsymbol{p}_2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \boldsymbol{g} \\ \boldsymbol{0} \end{pmatrix} = \boldsymbol{P}\begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{0} \\ \boldsymbol{g}_2 \end{pmatrix} \tag{4.3}$$

of the solution and right-hand side of (2.2). Thus, to solve (2.2), we obtain auxiliary variables $\boldsymbol{q}_1$ and $\boldsymbol{\lambda}_2$ from the equation

$$\begin{pmatrix} \boldsymbol{B}_{11} & \boldsymbol{A}_1^T \\ \boldsymbol{A}_1 & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \boldsymbol{q}_1 \\ \boldsymbol{\lambda}_2 \end{pmatrix} = -\begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{0} \end{pmatrix}, \tag{4.4}$$

and subsequently solve the equations

$$\left(\boldsymbol{B}_{22} - (\boldsymbol{B}_{21} \quad \boldsymbol{A}_2^T)\begin{pmatrix} \boldsymbol{B}_{11} & \boldsymbol{A}_1^T \\ \boldsymbol{A}_1 & \boldsymbol{0} \end{pmatrix}^{-1}\begin{pmatrix} \boldsymbol{B}_{21}^T \\ \boldsymbol{A}_2 \end{pmatrix}\right)\boldsymbol{p}_2 = -\boldsymbol{g}_2 - (\boldsymbol{B}_{21} \quad \boldsymbol{A}_2^T)\begin{pmatrix} \boldsymbol{q}_1 \\ \boldsymbol{\lambda}_2 \end{pmatrix} \tag{4.5}$$

and

$$\begin{pmatrix} \boldsymbol{B}_{11} & \boldsymbol{A}_1^T \\ \boldsymbol{A}_1 & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \boldsymbol{p}_1 \\ \boldsymbol{\lambda} \end{pmatrix} = -\begin{pmatrix} \boldsymbol{g}_1 \\ \boldsymbol{0} \end{pmatrix} - \begin{pmatrix} \boldsymbol{B}_{21}^T \\ \boldsymbol{A}_2 \end{pmatrix}\boldsymbol{p}_2 \tag{4.6}$$

and therefore require decompositions of the matrix

$$\begin{pmatrix} B_{11} & A_1^T \\ A_1 & 0 \end{pmatrix}, \tag{4.7}$$

and its Schur-complement,

$$S = B_{22} - (B_{21} \quad A_2^T) \begin{pmatrix} B_{11} & A_1^T \\ A_1 & 0 \end{pmatrix}^{-1} \begin{pmatrix} B_{21}^T \\ A_2 \end{pmatrix}, \tag{4.8}$$

in $P^T K P$. These decompositions would be performed implicitly if we factorized K as (3.3) with the pivot sequence defined by P, but a number of salient points may be made if we consider (4.7) and (4.8) explicitly.

4.2 An equivalence to reduced-variable methods

Since A_1 is non-singular, we have an explicit form for the inverse of (4.7),

$$\begin{pmatrix} B_{11} & A_1^T \\ A_1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & A_1^{-1} \\ A_1^{-T} & -A_1^{-T} B_{11} A_1^{-1} \end{pmatrix}. \tag{4.9}$$

Gill, Murray, Saunders and Wright (1990, Theorem 7.2) observe that this enables us to rewrite (4.8) as

$$S = Z^T B Z, \quad \text{where} \quad Z = \begin{pmatrix} P_{11}^T & P_{21}^T \\ P_{13}^T & P_{23}^T \end{pmatrix} \begin{pmatrix} I \\ -A_1^{-1} A_2 \end{pmatrix}, \tag{4.10}$$

and the matrix Z satisfies $AZ = 0$. The equations (4.4)–(4.6) are then equivalent to the reduced (quasi-)Newton equations (2.6), together with the extra equation

$$A_1^T \lambda = -g_1 - B_{11} p_1 - B_{21}^T p_2 \tag{4.11}$$

for the Lagrange multipliers. Thus forcing *ba* pivots until we have exhausted all the A-rows of K is equivalent to finding a representation of the null-space of A and using the reduced-variable method described in Section 2.1. In particular, the Schur-complement matrix, S, is a reduced Hessian matrix.

4.3 Boundedness of perturbations

Because of the relationship (4.10), the norm of S satisfies

$$\|S\| \leq (1 + \|A_1^{-1} A_2\|)^2 \|B\|, \tag{4.12}$$

and thus element growth may be controlled by using an appropriate threshold-pivoting tolerance when factorizing A_1. Therefore, if one of the modified Cholesky methods cited in Section 3.1 is subsequently employed to factorize S, the perturbation matrix E will remain bounded.

4.4 Appropriate orderings

As the same permutation may be used at *every* iteration of the nonlinear programming algorithm, it may be worth investing considerable effort in producing a good ordering. As 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 within the Schur complement and the factorization of $\boldsymbol{A}_1$ as possible. Notice that each *ba* pivot requires that we select a row and 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 may then be applied recursively to the Schur-complement of this pivot in $\boldsymbol{K}$ to determine *ba* pivots $2, \cdots, m$. Suppose that we consider the permutation

$$\boldsymbol{P}_1^T \boldsymbol{K} \boldsymbol{P}_1 = \left(\begin{array}{cc|cc} \beta & \alpha & \boldsymbol{b}_c^T & \boldsymbol{a}_c^T \\ \alpha & 0 & \boldsymbol{a}_r^T & \boldsymbol{0} \\ \hline \boldsymbol{b}_c & \boldsymbol{a}_r & \boldsymbol{B}_R & \boldsymbol{A}_R^T \\ \boldsymbol{a}_c & \boldsymbol{0} & \boldsymbol{A}_R & \boldsymbol{0} \end{array} \right), \tag{4.13}$$

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

$$\boldsymbol{S}_1 = \begin{pmatrix} \boldsymbol{B}_R & \boldsymbol{A}_R^T \\ \boldsymbol{A}_R & \boldsymbol{0} \end{pmatrix} - \frac{1}{\alpha^2} \left[\alpha \begin{pmatrix} \boldsymbol{b}_c \\ \boldsymbol{a}_c \end{pmatrix} (\boldsymbol{a}_r^T \quad \boldsymbol{0}) + \alpha \begin{pmatrix} \boldsymbol{a}_r \\ \boldsymbol{0} \end{pmatrix} (\boldsymbol{b}_c^T \quad \boldsymbol{a}_c^T) - \beta \begin{pmatrix} \boldsymbol{a}_r \\ \boldsymbol{0} \end{pmatrix} (\boldsymbol{a}_r^T \quad \boldsymbol{0}) \right]. \tag{4.14}$$

Notice that no fill-in occurs in the zero, bottom block of $\boldsymbol{S}_1$. We now follow Markowitz (1957) 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

$$\alpha \begin{pmatrix} \boldsymbol{b}_c \\ \boldsymbol{a}_c \end{pmatrix} (\boldsymbol{a}_r^T \quad \boldsymbol{0}) + \alpha \begin{pmatrix} \boldsymbol{a}_r \\ \boldsymbol{0} \end{pmatrix} (\boldsymbol{b}_c^T \quad \boldsymbol{a}_c^T) - \beta \begin{pmatrix} \boldsymbol{a}_r \\ \boldsymbol{0} \end{pmatrix} (\boldsymbol{a}_r^T \quad \boldsymbol{0}). \tag{4.15}$$

There are two cases to consider.

Following Duff et al. (1991), 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 (4.15) is

$$n_s = 2n_z(\boldsymbol{a}_r)[n_z(\boldsymbol{a}_c) + n_z(\boldsymbol{b}_c)] - n_o(\boldsymbol{a}_r, \boldsymbol{b}_c)^2, \tag{4.16}$$

while a tile pivot yields

$$n_s \leq 2n_z(\boldsymbol{a}_r)[n_z(\boldsymbol{a}_c) + n_z(\boldsymbol{b}_c)] - n_o(\boldsymbol{a}_r, \boldsymbol{b}_c)^2 + [n_z(\boldsymbol{a}_r) - n_o(\boldsymbol{a}_r, \boldsymbol{b}_c)]^2 \tag{4.17}$$

(the inequality in (4.17) accounts for the possibility of exact cancellation between the terms in (4.15)). Thus, if $\boldsymbol{A}$ has rows $\boldsymbol{a}_{r_i}$, $i = 1, \cdots, m$ and columns $\boldsymbol{a}_{c_j}$,

$j = 1, \cdots, n$ and $\boldsymbol{B}$ has columns $\boldsymbol{b}_{c_j}$, $j = 1, \cdots, n$, one possibility is to pick the ba pivot for which

$$|a_{i,j}| \geq v \max_{1 \leq l \leq n} |a_{i,l}| \tag{4.18}$$

for some pivot tolerance $0 < v \leq 1$ and for which

$$\sigma^e_{i,j} = \begin{cases} 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 1) - n_o(\boldsymbol{a}_{r_i}, \boldsymbol{b}_{c_i})^2 & \text{when} \quad b_{j,j} = 0 \\ 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 2) - \\ (n_o(\boldsymbol{a}_{r_i}, \boldsymbol{b}_{c_i}) - 1)^2 + (n_z(\boldsymbol{a}_{r_i}) - n_o(\boldsymbol{a}_{r_i}, \boldsymbol{h}_{c_j}) - 2)^2 & \text{when} \quad b_{j,j} \neq 0 \end{cases} \tag{4.19}$$

is smallest. However, as computing $n_o(\boldsymbol{a}_{r_i}, \boldsymbol{b}_{c_j})$ may prove to be unacceptably expensive, we follow Duff et al. (1991) and overestimate (4.16) and (4.17) by assuming that, except in the pivot rows, there are no overlaps and thus pick the pivot for which

$$\sigma^a_{i,j} = \begin{cases} 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 1) & \text{when} \quad b_{j,j} = 0 \\ 2(n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j}) - 2) + (n_z(\boldsymbol{a}_{r_i}) - 1)^2 & \text{when} \quad b_{j,j} \neq 0 \end{cases} \tag{4.20}$$

is smallest. It is relatively straightforward to compute and update the nonzero counts required to use (4.20). Indeed, as $n_z(\boldsymbol{a}_{r_i})$ and $n_z(\boldsymbol{a}_{c_j}) + n_z(\boldsymbol{b}_{c_j})$ are, respectively, the row and column counts for the matrix

$$\begin{pmatrix} \boldsymbol{B} \\ \boldsymbol{A} \end{pmatrix}, \tag{4.21}$$

the schemes described by Duff, Erisman and Reid (1986, Section 9.2) are appropriate.

Although this section has been concerned with direct methods of solution, we observed in Section 4.2 that the use of m ba pivots is equivalent to calculating a change of basis so that a reduced variable method can be used. If, after performing such pivots, our aim is subsequently to use an *iterative* method to (approximately) solve the resulting unconstrained problem (2.9), a different strategy for selecting the ba pivots is appropriate. For then, our aim should be to obtain as sparse a factorization of $\boldsymbol{A}_1$ as possible — so that the matrix-vector product $\boldsymbol{Z}^T\boldsymbol{B}\boldsymbol{Z}$ can be formed as economically as possible (see Section 2.2) — and the interaction between $\boldsymbol{A}$ and $\boldsymbol{B}$ is mostly irrelevant. A good ordering in this case may be obtained by minimizing the Markowitz count

$$\sigma^o_{i,j} = (n_z(\boldsymbol{a}_{r_i}) - 1)(n_z(\boldsymbol{a}_{c_j}) - 1) \tag{4.22}$$

over all indices $i = 1, \cdots, m$ and $j = 1, \cdots, n$ which satisfy (4.18). Alternatively, one might select the column index j to minimize $n_z(\boldsymbol{a}_{c_j})$ and then pick any i which satisfies (4.18).

4.5 Dense rows

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 Forsgren and Murray (1993) (see Section 3.3). A drawback is that, by allowing b_+ and a_- pivots, we may introduce fill-ins into the "zero" block of (3.2) and, thereafter the Markowitz costs (4.19) and (4.20) are inappropriate. Appropriate Markowitz costs in this case have been suggested by Duff et al. (1991). Preference should still be given to pivots involving $\boldsymbol{A}$-rows if at all possible.

However, even if we allow all types of pivots suggested by Forsgren and Murray, there are still cases where the Schur complement becomes unacceptably dense. In the next section, we consider methods which aim to avoid such difficulties.

5 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. Firstly, we might pick a favourable 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, George and Liu, 1981, or Duff et al., 1986). 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.

Secondly, we might use a combined analysis-factorization strategy, more typical of unsymmetric factorizations (again, see Duff et al., 1986), 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 as 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 a unacceptable level of fill-in within $\boldsymbol{B}$. Our preference is for the first (separate analysis and factorization phases) strategy as 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.

5.1 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.

5.1.1 Pseudo modification of b_- pivots

Suppose the uneliminated Schur-complement when we encounter potential breakdown is of the form

$$\begin{pmatrix} \beta_- & \boldsymbol{s}^T \\ \boldsymbol{s} & \boldsymbol{S} \end{pmatrix}, \tag{5.1}$$

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

$$\beta_+ \geq \max(\sigma, ||\boldsymbol{s}||_\infty), \tag{5.2}$$

and let

$$\beta \stackrel{\text{def}}{=} \beta_+ - \beta_-. \tag{5.3}$$

Then if we could replace β_- by β_+, the latter would be an acceptable pivot. But this is precisely what we do, leaving the consequences for later. We call such a modification of $\boldsymbol{B}$ a *pseudo modification* as 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 then 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 (1967) suggested using pseudo modifications as an alternative to pivoting in Gaussian elimination, and 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 may lead to undesirable fill-in.

We will have formed a stable factorization of

$$N \stackrel{\text{def}}{=} \begin{pmatrix} B + \Delta B & A^T \\ A & 0 \end{pmatrix}, \tag{5.4}$$

where $B = H + \Delta H$, the diagonal matrix ΔB corresponds to the m_- (say) modified b_- pivots and the other diagonal matrix ΔH corresponds to those diagonals changed using the dense modified Cholesky factorization. These later diagonal modifications are necessary to ensure that 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.

5.1.2 Countering the effects of pseudo modifications

The system we wish to solve is (2.2), while we have a factorization of (5.4). Suppose that the i-th pseudo modification ($1 \leq i \leq m_-$) occurred in column j_i of H and that the modification was $\beta_{j_i} > 0$. Let

$$\Delta B = V^T V, \tag{5.5}$$

where V^T is the n by m_- matrix whose columns are $v_i \stackrel{\text{def}}{=} \sqrt{\beta_{j_i}} e_{j_i}$. Then, we may write (2.2) as

$$\begin{pmatrix} B + \Delta B - V^T V & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} p_n \\ \lambda_n \end{pmatrix} = \begin{pmatrix} -g \\ 0 \end{pmatrix} \tag{5.6}$$

or equivalently as

$$\begin{pmatrix} B + \Delta B & A^T & V^T \\ A & 0 & 0 \\ V & 0 & I_{m_-} \end{pmatrix} \begin{pmatrix} p_n \\ \lambda_n \\ s_n \end{pmatrix} = \begin{pmatrix} -g \\ 0 \\ 0 \end{pmatrix} \tag{5.7}$$

for some auxiliary vector s_n. A standard block-decomposition of (5.7) shows that we may determine the solution to (2.2) by solving, in order,

$$\begin{pmatrix} B + \Delta B & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} q_n \\ \pi_n \end{pmatrix} = \begin{pmatrix} -g \\ 0 \end{pmatrix}, \tag{5.8}$$

$$G s_n = V q_n, \tag{5.9}$$

and

$$\begin{pmatrix} B + \Delta B & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} p_n \\ \lambda_n \end{pmatrix} = \begin{pmatrix} -g - V^T s_n \\ 0 \end{pmatrix}, \tag{5.10}$$

where

$$G = I_{m_-} - (V \quad 0) \begin{pmatrix} B + \Delta B & A^T \\ A & 0 \end{pmatrix}^{-1} \begin{pmatrix} V^T \\ 0 \end{pmatrix}. \tag{5.11}$$

Thus to solve (2.2) via the stable factorization (5.4), we also need to form and factorize G. But this factorization also reveals whether or not the modification ΔB is necessary. For we have

Proposition 5.1 *B is second-order sufficient if and only if G is positive definite.*

Proof. The result follows from Sylvester's law of inertia (see, e.g. Cottle, 1974) by considering different block decompositions of

$$M = \begin{pmatrix} B + \Delta B & A^T & V^T \\ A & 0 & 0 \\ V & 0 & I_{m_-} \end{pmatrix}. \tag{5.12}$$

Pivoting on the first two blocks of M and using the definitions (5.4) and (5.11), reveals that

$$\text{In}(M) = \text{In}(N) + \text{In}(G), \tag{5.13}$$

while pivoting on the last block, and using the definition (3.2) gives that

$$\text{In}(M) = \text{In}\left(I_{m_-}\right) + \text{In}(K). \tag{5.14}$$

But, as I_{m_-} is clearly positive definite, and N is second-order sufficient,

$$\text{In}\left(I_{m_-}\right) = (m_-, 0, 0) \quad \text{and} \quad \text{In}(N) = (n, m, 0). \tag{5.15}$$

Thus combining (5.13)—(5.15), we see that $\text{In}(K) = (n, m, 0)$ if and only if $\text{In}(G) = \text{In}\left(G^{-1}\right) = (m_-, 0, 0)$. The required result then follows as B is second-order sufficient if and only if $\text{In}(K) = (n, m, 0)$ (see Gould, 1985). ∎

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

$$\left(\frac{\gamma_i}{1 + \gamma_i}\right) v_i v_i^T. \tag{5.16}$$

Thus, modification of G gives an implicit modification of B, and the actual modification is no larger than the pseudo modification.

5.1.3 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.

   (a) Firstly, construct k_2 2 by 2 ba pivots, using the strategy outlined in Section 4.4 (This involves processing the values of A but not B). Stop once the resulting Schur-complement has reached a specified density.

   (b) Next, construct k_1 1 by 1 b pivots from the remaining Schur-complement using, for instance, the minimum degree ordering (see, for example, George and Liu, 1981, or Duff et al., 1986). Stop once the resulting Schur-complement has reached a specified density.

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

2. Perform the complete numerical factorization.

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

   (b) Perform k_1 1 by 1 sparse eliminations, using the pivots specified in 1(b) above. Modify any b_- pivots encountered to ensure that they are sufficiently positive, using the scheme of, for example, Schnabel and Eskow (1991). Record any pseudo modifications made.

   (c) For the remaining dense block, factorize using the scheme of Forsgren and Murray (1993) until all the A-rows have been eliminated. Thereafter, use a dense modified Cholesky factorization to eliminate the remaining B-rows.

   (d) If any pseudo modifications were made in 2(b) above, form the matrix G. Perform a modified Cholesky factorization of G.

3. Perform any solves, using the factors obtained in 2 above, by solving the sequence of equations (5.8)–(5.10).

One would normally anticipate only performing 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 also stress that (4.14) indicates that the Schur complement of the A-rows following the ba pivots is independent of B and thus, as A is independent of x, need only be formed 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 G. Although the number of pseudo modifications will not be known until the numerical factorization phase, it may be possible to influence this by over-riding the pivot selection outlined in Section 4.4 to favour incorporating potentially small or negative elements within the initial ba pivots. For instance, if a diagonal of 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(b).

5.1.4 Generalizations

When f is strictly convex, no pseudo modifications should be necessary. In other cases, it is possible that the dimension of G might be unacceptably high. However, considering (5.1), it is clear that rather in addition to just modifying the diagonal β_-, we are free to modify *as many nonzero elements of* s *as we like* without introducing extra fill in the factorization Thus, if the diagonal of S, in a row in which s has a nonzero element, is also small or negative, we should modify the corresponding element of 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 the i-th column of the matrix V now contains nonzeros in all positions for which the j_i-th pivot column was modified.

None the less, we have to recognize that there are some matrices for which this strategy is inappropriate, as the following example shows:

Example 5.1 *Suppose $H = -I_n$ and $A = e^T$, the vector of* 1s. *Then a ba pivot is out of the question as the resulting Schur complement would be completely dense. But, as each diagonal of H is negative, and as there is no connectivity between the diagonals, n pseudo modifications will be required. Unfortunately, G will then be a dense n by n matrix.*

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

$$V^T \left(I_{m_-} - (I_{m_-} + G)^{-1} \right) V. \tag{5.17}$$

Thus, provided that G is positive semi-definite, the actual modification will again be smaller than the pseudo modification. A simple scheme would be to replace G by τI_{m_-}, where τ is chosen so large that $\tau I_{m_-} - G$ is positive definite whenever G is not positive definite. The advantage of this replacement is that the storage and factorization overheads associated with 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.

5.2 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 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 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 B if it fails to be second-order sufficient.

5.2.1 The condemned submatrix

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

We pick a nonsingular, square submatrix, the *condemned* submatrix, C, of S, which contains all the A-rows and perhaps some of the B-rows (but not the b_- pivot row) of S and has precisely m_d negative eigenvalues. The condemned submatrix will be eliminated last of all and thus any B-rows included in C will not be generally available as pivots. The aim is that, when only the rows which make up C remain to be eliminated, the uneliminated Schur complement will have precisely m_d negative eigenvalues and hence K will have exactly m negative eigenvalues. The Schur complement S has at least m_d negative eigenvalues. A suitable 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 5.2.4.

A factorization of the condemned submatrix should be obtained. As the C-rows of 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 C which can be accommodated. We note that the dimension of C as constructed above is $2m_d - m_{a_-}$ and hence lies between m_d and $2m_d$.

5.2.2 The consequences of pivoting

With this choice of C, S is a permutation of the matrix

$$\left(\begin{array}{cc|c} \beta_- & s_1^T & s_2^T \\ s_1 & C & S_{21}^T \\ \hline s_2 & S_{21} & S_{22} \end{array}\right), \tag{5.18}$$

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

$$\left(\begin{array}{cc} \beta_- & s_2^T \\ s_2 & S_{22} \end{array}\right) - \left(\begin{array}{c} s_1^T \\ S_{21} \end{array}\right) C^{-1} \left(\begin{array}{cc} s_1 & S_{21}^T \end{array}\right) \tag{5.19}$$

is sufficiently positive definite. In particular, if $\beta_- - s_1^T C^{-1} s_1$ is insufficiently positive, B is not second-order sufficient and should be modified.

With this in mind, if

$$\beta_- - s_1^T C^{-1} s_1 < \sigma, \tag{5.20}$$

we modify B by replacing β_- by at least

$$\beta_+ \stackrel{\text{def}}{=} \max\left(\sigma + s_1^T C^{-1} s_1, ||s||\right), \tag{5.21}$$

where $s^T = \left(s_1^T \quad s_2^T\right)$. Conversely, if

$$\beta_- - s_1^T C^{-1} s_1 \geq \sigma > 0, \tag{5.22}$$

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 $C - s_1 s_1^T/\beta_-$. This follows directly from the inertial identity

$$\text{In}\left(C - s_1 s_1^T/\beta_-\right) = \text{In}\,(C) + \text{In}\left(\beta_- - s_1^T C^{-1} s_1\right) - \text{In}\,(\beta_-)\,, \tag{5.23}$$

for block decompositions of

$$\begin{pmatrix} \beta_- & s_1^T \\ s_1 & C \end{pmatrix} \tag{5.24}$$

(see, e.g. Cottle, 1974). We then pivot on the possibly modified value of β_- and replace C by $C - s_1 s_1^T/\beta_-$ — we update the matrix factorization to account for this (see, Gill, Golub, Murray and Saunders, 1974). We repeat this procedure until we have eliminated the remaining S_{22} rows, at which point, the only non eliminated portion of K is the (updated) matrix C.

Alternatively, once it has been determined that B is not second-order sufficient, we might modify all remaining B pivots. One possibility, in the same vein as Schnabel and Eskow (1991), is to insist that all diagonals are larger than the sum of the absolute values of the (remaining) off diagonal terms in B-rows.

For the case of Example 5.1 in Section 5.1, the explicit modification scheme considered here would be preferable. The condemned submatrix might be made up from the last row of H — indeed, any row of H will do — and the single A-row. Examining (5.20) for each diagonal pivot in turn, it follows that H is not second-order sufficient, every pivot will be modified, but no fill-in takes place.

5.2.3 Other pivot types

If the only possible pivots in 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 5.1.1) or reject the potential pivot and pass to the next.

It may be better to consider 2 by 2 pivots,

$$\begin{pmatrix} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{pmatrix}, \tag{5.25}$$

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

$$\left(\begin{array}{ccc|c} \beta_{11} & \beta_{21} & s_{11}^T & s_{21}^T \\ \beta_{21} & \beta_{22} & s_{12}^T & s_{22}^T \\ s_{11} & s_{12} & C & S_{21} \\ \hline s_{21} & s_{22} & S_{21} & S_{22} \end{array}\right), \tag{5.26}$$

and B is second-order sufficient only if the matrix

$$\begin{pmatrix} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{pmatrix} - \begin{pmatrix} s_{11}^T \\ s_{12}^T \end{pmatrix} C^{-1} (s_{11} \quad s_{12}) \tag{5.27}$$

is sufficiently positive definite. As before, if (5.27) is indefinite, the potential pivot (5.25) should be modified before use. The inertial result

$$\begin{aligned} \text{In}\left(C - (s_{11}^T \quad s_{12}^T) \begin{pmatrix} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{pmatrix}^{-1} \begin{pmatrix} s_{11} \\ s_{12} \end{pmatrix} \right) = \\ \text{In}\,(C) + \text{In}\left(\begin{pmatrix} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{pmatrix} - \begin{pmatrix} s_{11}^T \\ s_{12}^T \end{pmatrix} C^{-1} (s_{11} \quad s_{12}) \right) - \text{In} \begin{pmatrix} \beta_{11} & \beta_{21} \\ \beta_{21} & \beta_{22} \end{pmatrix} \end{aligned} \tag{5.28}$$

once again indicates that the updated C after the pivot inherits the correct number of negative eigenvalues.

5.2.4 Calculating the condemned submatrix

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

Let the m_d uneliminated A-rows in the Schur complement, S_{ba}, following the m_e ba pivots, be A_{ba}. Similarly, let the $n - n_e$ uneliminated B-rows and columns in S_{ba} following these ba pivots be B_{ba}. Further, let

$$\mathcal{O} = \{i_1, i_2, \cdots, i_{n-m_e}\} \tag{5.29}$$

be the ordered pivot sequence for the elimination of B_{ba}. Now define the ordered sets

$$\mathcal{P}_1 = \{i_1, i_2, \cdots, i_{n_e-m_e}\} \quad \text{and} \quad \mathcal{P}_2 = \{i_{n-m_e}, i_{n-n_e-1}, \cdots, i_{n_e-m_e+1}\} \tag{5.30}$$

and the ordered set of *preferences*

$$\mathcal{P} = \mathcal{P}_1 \bigcup \mathcal{P}_2. \tag{5.31}$$

For example, suppose that 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 from 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, non-singular subset, $\mathcal{C}$, of the columns of 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 S_{ba}. Moreover, the submatrix, C_{ba}, formed by taking the rows and columns of 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 B-rows of 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, C. This matrix has the correct inertia as the sub-block of C_{ba} corresponding to the $\mathcal{C}_1$ pivots is contained within

the sub-block of S_{ba} corresponding to the $\mathcal{P}_1$ pivots, and the latter sub-block is positive definite as the first $n_e - m_e$ b pivots on 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 A_c is initialized as A_{ba}. The columns of 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 (4.18) holds (where here $a_{i,j}$ are the entries of A_c), column j is added to $\mathcal{C}$ and removed from $\mathcal{P}$, and A_c is reset to the Schur complement of A_c following a pivot on $a_{i,j}$. On the other hand, if (4.18) 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 firstly encourages ba pivots whose b_+ component has already been used — the resulting a_- pivot is then available and reduces the possible dimension of C. If $\mathcal{P}$ is not entirely made up from $\mathcal{P}_1$, the preference secondly encourages pivots from those 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 C from the outset.

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

$$\mathcal{P} = \{i_{n-m_e}, i_{n-m_e-1}, \cdots, i_1\}\,. \tag{5.32}$$

This will have the effect that the resulting 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 favour including disadvantageous B-rows within the condemned submatrix.

5.3 Modifications during the ba-pivot stage

The methods we have considered so far wait until we have performed all possible (i.e., possible given sparsity constraints) ba pivots before considering modifications to H. In this section, we comment on another alternative in which modifications are considered in the ba pivot phase. This technique has been mentioned by others — see, for example, Hestenes (1969) and Powell (1969) — in the context of augmented Lagrangian methods for small-scale computations. However, as we shall see, it appears that such a technique is unlikely to offer improvements over our previously mentioned methods in the large-scale case.

Firstly, observe that the solution, p_r, and related Lagrange multipiers, λ_r, to the generic system

$$\begin{pmatrix} B & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} p_r \\ \lambda_r \end{pmatrix} = \begin{pmatrix} r_p \\ r_\lambda \end{pmatrix}, \tag{5.33}$$

also satisfy the equations

$$\begin{pmatrix} \boldsymbol{B} + \boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{p}_r \\ \boldsymbol{\lambda}_r \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_p + \boldsymbol{W}\boldsymbol{r}_\lambda \\ \boldsymbol{r}_\lambda \end{pmatrix}, \tag{5.34}$$

where $\boldsymbol{W}$ is any m by m matrix. Now, consider the case where $\boldsymbol{W}$ is a positive semi-definite diagonal matrix. Then, following (for instance) Bertsekas (1982, Lemma 1.25), it is straightforward to see that $\boldsymbol{B}$ is second-order sufficient if and only if $\boldsymbol{B} + \boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ is positive definite for some sufficiently large $\boldsymbol{W}$. This suggests that one might simply add large contributions $\boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ to $\boldsymbol{B}$ and solve (5.34) instead of (5.33). So long as $\boldsymbol{B}$ is second-order sufficient and $\boldsymbol{W}$ is sufficiently large, this strategy will then ensure that no further modification will be required during the subsequent factorization. Such a strategy has been advocated by a number of authors (see, for example, Gill et al., 1981, Section 5.4.2.1) as a possible way to solve small-scale nonconvex problems.

In the large-scale case, one needs to be cautious as the matrix $\boldsymbol{B} + \boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ may be condiderably more dense than $\boldsymbol{B}$. Indeed, if $\boldsymbol{A}$ has any relatively dense rows, $\boldsymbol{B} + \boldsymbol{A}^T\boldsymbol{W}\boldsymbol{A}$ will be rather dense. However, examining (4.14), it is immediately clear that the $\boldsymbol{B}$-rows of the Schur-complement following a *ba* pivot already contains a contribution from $\boldsymbol{a}_r\boldsymbol{a}_r^T$. Thus no further fill-in will result from an additional contribution $\omega\boldsymbol{a}_r\boldsymbol{a}_r^T$, for some $\omega \geq 0$, and one would then hope to be able to influence future pivots by a judicious choice of ω. Unfortunately, it is straightforward to show from (4.14) that the resulting Schur complement, $\boldsymbol{S}_1$, is independent of ω and thus that an additional term $\omega\boldsymbol{a}_r\boldsymbol{a}_r^T$ has no effect. Thus there is no benefit to be made from adding rank-one terms involving $\boldsymbol{A}$-rows involved in *ba* pivots. As we advocate stopping *ba* pivoting only when the remaining $\boldsymbol{A}$-rows are too dense, any additional rank-one term involving an uneliminated $\boldsymbol{A}$-row will cause significant fill-in and is therefore not recommended.

6 Numerical experiments

We are currently planning to implement a code to solve systems of the form (2.2) for the Harwell Subroutine Library. A key requirement is that $\boldsymbol{B}$ should be a second-order sufficient modification of $\boldsymbol{H}$. In order 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.

6.1 Implementation details

We have written a prototype implementation of the algorithm outlined in Section 5.1.3. This implicit modification algorithm divides naturally into an analysis, a factorization and a solve phase.

The analysis phase needs only be performed 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 a number control parameters, in particular the pivot threshold tolerance v (see (4.18)), the density δ_a of the Schur complement of $\boldsymbol{A}$ during *ba* pivoting at

which the remaining rows of A may be treated as dense, and the density δ_b of the Schur complement of B during b pivoting at which the remaining rows of B may be treated as dense. We choose to switch to full-matrix code as soon as the density of the Schur complement of 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 A during ba pivoting exceeds δ_a is sometimes inappropriate as further cheap ba pivots may be possible — remember that it helps to eliminate as many A rows as possible before the b pivoting stage. In particular, if the matrix 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 (4.20) 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 6.3.

During the factorization phase, once a negative b pivot 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 rows of A and H which are left over following the sparse ba and b pivoting steps, along with the matrix G, are treated as dense matrices. The highest appropriate levels of BLAS (see, for instance, Dongarra, Duff, Sorensen and van der Vorst, 1991) are used to perform the dense operations wherever possible.

In addition, we have also implemented the explicit modification scheme suggested in Section 5.2. This differs from the implicit modification scheme described above in two respects. Firstly, the ordering of the b pivots may be altered to provide a non-singular condemned matrix, if it is needed. We have implemented the method described in Section 5.2.4 using the preference (5.32). Secondly, 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 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 Anderson, Bai, Bischof, Demmel, Dongarra, DuCroz, Greenbaum, Hammarling, McKenney, Ostrouchov and Sorensen, 1995) are used to compute and update the QR factors.

All our tests were performed on an IBM RS6000 3BT workstation; the codes are all double precision fortran 77, compiled under xlf with -O optimization, and IBM library BLAS are used.

6.2 Test examples

We considered all of the larger quadratic programming examples in the current CUTE test set (see Bongartz et al., 1995), excepting 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 6.1. In order 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

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

Table 6.1: 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 eiegnvalues of reduced Hessian, *nullity* = nullity of K, *convex?* = is the Hessian $H(x)$ positive definite?.

then generated so that the required solution is a vector of ones.

6.3 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 convex while the reduced Hessian of the second has a few negative eigenvalues. The behaviour on these examples is representative of the whole set.

`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

Table 6.2: 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 analyse, factorize and solve (cpu seconds).

We give our results in Table 6.2 on runs which used the explicit modification algorithm; similar results were observed for the implicit modification scheme. Examining the times taken during the analyse 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

	MA27				MA47			
Problem	fill-in	amal.	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

Table 6.3: Performance of MA27 and MA47 (default settings).
Key: *fill-in* = fill-in during factorization, *anal.*, *fact.*, *solve* = times for analyse, factorize and solve (cpu seconds).

Problem	fill-in A	fill-in H	fill-in dense	dense rows A	dense rows H	dense rows G	# mod	time anal.	time fact.	time 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

Table 6.4: 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 analyse, factorize and solve (cpu seconds).

Problem	fill-in A	fill-in H	dense	dense rows A	dense rows H	# mod	time anal.	time fact.	time 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

Table 6.5: 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 analyse, factorize and solve (cpu seconds).

is too small is also undesirable as 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.

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 H to produce a second-order sufficient 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 6.3. We note that, although MA47 was especially designed to cope with augmented systems of the form (2.2), it is often less efficient than the general purpose method MA27. In its defence, we sometimes observed that MA47 obtained accurate solutions to (2.2) while its older sister failed to do so; the NCVXQP problems are cases in point.

In Table 6.4, 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 A and B to be at most 350 even though this means that the target densities δ_a or δ_b may be exceeded. We have found that, even though 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. Firstly, at least in comparison with MA47, KKTSOL performs acceptably well in many cases. Clearly, restricting the pivot order has some detrimental effect on the fill-in. This is somewhat 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.

Secondly, for the nonconvex problems, a large number of pseudo-modifications are 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 5.1 in Section 5.1, and, as predicted, the implicit modification scheme is slow precisely because G is large.

In Table 6.5, we consider the performance of the explicit modification variant on the test set. We firstly note that the alteration of the b pivot order sometimes has a slightly detrimental effect on the analysis times, but that 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 a large matrix 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 matrice 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 convex. Thus, the differences in factorization times in Table 6.5 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 non-modification 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 only been partially successful in fulfilling our stated aims.

7 Other possibilities

7.1 Mixed direct-iterative methods

Another possibility is to combine the iterative methods of Arioli et al. (1993) with a variation on the direct methods of Sections 4 and 5 so as to counteract the latter's difficulties with unacceptable fill-in.

We suppose we have started a factorization of $\boldsymbol{K}$ using Forsgren and Murray's (1993) pivot scheme, have eliminated all the $\boldsymbol{A}$-rows and n_e $\boldsymbol{B}$-rows, perhaps modifying pivots as we proceed, but that the remaining Schur complement matrix is too large and dense to proceed further. This is equivalent to permuting the matrix $\boldsymbol{K}$ to produce a partition (4.1), where $\boldsymbol{B}_{11}$ and $\boldsymbol{A}_1$ are now n_e by n_e and m by n_e matrices, respectively. As before, a similar permutation and partition of the solution and right-hand side vectors enables us to decompose the solution process into successively solving the three systems of equations (4.4)–(4.6). We have been careful to arrange that the matrix (4.7) has a sparse factorization but unfortunately the matrix (4.8) is too large and dense for us to factorize. However, the relationship (4.8) indicates that matrix-vector products between $\boldsymbol{S}$ and a given vector $\boldsymbol{v}$ are possible, each product requiring the solution of an intermediate linear system, whose coefficient matrix is (4.7), sandwiched between a pair of sparse-matrix-vector products. Thus we may contemplate solving (4.4)–(4.6) by solving (4.4) and (4.6), using the factorization of (4.7), and (4.5), using an appropriate iterative method.

It is convenient to view the solution of (4.5) as the solution of the related problem

$$\underset{\boldsymbol{p}_2 \in \Re^{n-n_e}}{\text{minimize}} \quad \tfrac{1}{2}\boldsymbol{p}_2^T \boldsymbol{B}_z \boldsymbol{p}_2 + \boldsymbol{p}_2^T \boldsymbol{g}_z, \tag{7.1}$$

where $\boldsymbol{B}_z = \boldsymbol{S}$ and $\boldsymbol{g}_z = \boldsymbol{g}_2 + \boldsymbol{B}_{21}\boldsymbol{q}_1 + \boldsymbol{A}_2^T\boldsymbol{\lambda}_2$. Then we may use any of the iterative methods discussed by Arioli et al. (1993) to compute an appropriate, approximate solution to (7.1), modifying $\boldsymbol{B}_z$ if necessary.

7.2 A postiori modifications

A final possibility is to abandon the aim of modifying the matrix as the factorization, and instead to try to modify the matrix after the factorization has revealed that $\boldsymbol{H}$ is not second-order sufficient. Suppose that we initialize $\boldsymbol{B} = \boldsymbol{H}$, we form a stable factorization of (3.2) — using for instance `MA27` or `MA47` — and we find

that the inertia of $\boldsymbol{K}$ is $(n-l, m+l, 0)$, where $l > 0$. Then $\boldsymbol{B}$ is not second-order sufficient, and the reduced Hessian has l negative eigenvalues.

The simplest remedy under these circumstances, is to perform a second factorization, modifying each b pivot encountered so that it is positive. However, this does not tally with our stated aim (Section 2.3) that the calculation of the perturbation $\boldsymbol{E}$ should be an acceptable overhead in comparison with that of the initial factorization. The following alternative suggests itself.

Let $\boldsymbol{v}$ be any n-vector, and consider the modified matrix $\boldsymbol{B} + \boldsymbol{v}\boldsymbol{v}^T$. We then have,

Proposition 7.1 *Suppose that* $In(\boldsymbol{K}) = (n-l, m+l, 0)$. *Then*

$$In\begin{pmatrix} \boldsymbol{B}+\boldsymbol{v}\boldsymbol{v}^T & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} = (n-l+1, m+l-1, 0) \tag{7.2}$$

if and only if

$$(\boldsymbol{v}^T \quad \boldsymbol{0})\begin{pmatrix} \boldsymbol{B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}^{-1}\begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{0} \end{pmatrix} < -1. \tag{7.3}$$

Proof. The result again follows from Sylvester's law of inertia (see, e.g. Cottle, 1974) by considering different block decompositions of

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{A}^T & \boldsymbol{v}^T \\ \boldsymbol{A} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{v} & \boldsymbol{0} & -1 \end{pmatrix}. \tag{7.4}$$

Pivoting on the first two blocks of $\boldsymbol{M}$ reveals that

$$\mathrm{In}(\boldsymbol{M}) = \mathrm{In}(\boldsymbol{K}) + \mathrm{In}\left(-1 - (\boldsymbol{v}^T \quad \boldsymbol{0})\begin{pmatrix} \boldsymbol{B} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}^{-1}\begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{0} \end{pmatrix}\right), \tag{7.5}$$

while pivoting on the last block, and using the definition (3.2) gives that

$$\mathrm{In}(\boldsymbol{M}) = \mathrm{In}\begin{pmatrix} \boldsymbol{B}+\boldsymbol{v}\boldsymbol{v}^T & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix} + (0,1,0). \tag{7.6}$$

Comparing (7.5) and (7.6) then gives the required result. ∎

Proposition 7.1 may then be used recursively to produce a second-order sufficient matrix $\boldsymbol{B}$. Starting with $\boldsymbol{B}_0 = \boldsymbol{H}$, the update $\boldsymbol{B}_i = \boldsymbol{B}_{i-1} + \boldsymbol{v}_i\boldsymbol{v}_i^T$, $i = 1, \cdots, l$, is performed. Each vector v_i must, in turn, be chosen so that

$$(\boldsymbol{v}_i^T \quad \boldsymbol{0})\begin{pmatrix} \boldsymbol{B}_{i-1} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}^{-1}\begin{pmatrix} \boldsymbol{v}_i \\ \boldsymbol{0} \end{pmatrix} < -1. \tag{7.7}$$

The required $\boldsymbol{B}$ is then $\boldsymbol{B}_l$.

There are now two important observations. Firstly, in order to satisfy (7.7), it suffices to find a vector $\boldsymbol{w}_i$ such that

$$(\boldsymbol{w}_i^T \quad \boldsymbol{0})\begin{pmatrix} \boldsymbol{B}_{i-1} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{pmatrix}^{-1}\begin{pmatrix} \boldsymbol{w}_i \\ \boldsymbol{0} \end{pmatrix} < 0. \tag{7.8}$$

For then v_i is simply a suitable scaling of w_i. Secondly, letting r_i and s_i be the solution of

$$\begin{pmatrix} B_{i-1} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} r_i \\ s_i \end{pmatrix} = \begin{pmatrix} w_i \\ 0 \end{pmatrix}, \tag{7.9}$$

we see that satisfying (7.8) is equivalent to finding a vector r_i for which both $Ar_i = 0$ and $r_i^T B_{i-1} r_i < 0$, i.e., to finding a direction of negative curvature for the current model problem.

Unfortunately, there does not, as yet, appear to be an inexpensive method for finding such a direction, let alone the l separate directions required here. Both Forsgren and Murray (1993) and Forsgren, Gill and Murray (1995) suggest how such directions may be found provided that one is prepared to form a (second) partial factorization of K in which the pivot selection is restricted. Similarly, Conn and Gould (1984) give a method which requires that a "triangular-like" matrix be formed from the existing factors of K and this matrix then factorized to obtain a direction of negative curvature. Alas, none of these options really falls within our aim of finding a cheap modification.

Of course, if one were really able to find a good direction of negative curvature, it might be preferable to exploit it directly (see, e.g., Goldfarb, 1980, and Moré and Sorensen, 1979).

8 Conclusions and further comments

In this paper, we have showed that a number of modified factorization methods for linearly constrained optimization calculations made be derived, and 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. As 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 quadratic programming and linearly-constrained nonlinear optimization problems with the Harwell Subroutine Library.

We have purposely not attempted to derive directions of sufficient negative curvature for such problems (see, for example, Forsgren and Murray, 1993, Forsgren et al., 1995 and the references contained therein), even though algorithms which use them offer stronger convergence guarantees — specifically, convergence to a points for which second-order necessary optimality conditions hold. We intend to investigate this possibility for large-scale problems in future.

9 Acknowledgement

This paper is the successor to the technical report by Arioli et al. (1993). The author is extremely grateful to Mario Arioli, Tony Chan, Iain Duff, Jacek Gondzio, John Reid and Bobby Schnabel, for stimulating discussions on much of the material contained here, and to an anonymous referee of the Arioli et al. (1993) paper, 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

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*. SIAM, Philadelphia, USA, second edn, 1995.

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. Technical Report RAL-93-066, Rutherford Appleton Laboratory, Chilton, England, 1993.

D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, London, 1982.

I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, **21**(1), 123 – 160, 1995.

J. R. Bunch and L. C. Kaufman. Some stable methods for calculating inertia and solving symmetric linear equations. *Mathematics of Computation*, **31**, 163–179, 1977.

J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, **8**, 639–655, 1971.

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 Journal on Optimization*, **3**(4), 696–725, 1993.

T. F. Coleman and A. Pothen. The sparse null space basis problem. Technical Report CS-84-12, Department of Computer Science, The Pennsylvania State University, University Park, USA, 1984.

T. F. Coleman and A. Pothen. The null space problem II: Algorithms. *SIAM Journal on Algebraic and Discrete Methods*, **8**, 544–563, 1987.

A. R. Conn and N. I. M. Gould. On the location of directions of infinite descent for nonlinear programming algorithms. *SIAM Journal on Numerical Analysis*, **21**(6), 302–325, 1984.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. A globally convergent Lagrangian barrier algorithm for optimization with general inequality constraints and simple bounds. Technical Report RAL-92-067, Rutherford Appleton Laboratory, Chilton, England, 1992*a*.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT: *a Fortran package for large-scale nonlinear optimization (Release A)*. Number 17 *in* 'Springer Series in Computational Mathematics'. Springer Verlag, Heidelberg, Berlin, New York, 1992*b*.

A. R. Conn, N. I. M. Gould, and Ph. L. Toint. On the number of inner iterations per outer iteration of a globally convergent algorithm for optimization with general nonlinear inequality constraints and simple bounds. Technical Report RAL-92-068, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 1992*c*.

A. R. Conn, N. I. M. Gould, A. Sartenaer, and Ph. L. Toint. Convergence properties of an augmented Lagrangian algorithms for optimization with a combination of general equality and linear constraints. Technical Report RAL-95-009, Rutherford Appleton Laboratory, Chilton, England, 1995. To appear in SIAM Journal on Optimization.

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

R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact-Newton methods. *SIAM Journal on Numerical Analysis*, **19**(2), 400–408, 1982.

J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice-Hall, Englewood Cliffs, USA, 1983.

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

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, USA, 1991.

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.

I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software*, **9**(3), 302–325, 1983.

I. S. Duff and J. K. Reid. MA47: A Fortran code for the direct solution of indefinite symmetric linear systems of equations. Technical Report RAL-95-001, Rutherford Appleton Laboratory, Chilton, England, 1995.

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

I. S. Duff, N. I. M. Gould, J. K. Reid, J. A. Scott, and K. Turner. The factorization of sparse symmetric indefinite matrices. *IMA Journal of Numerical Analysis*, **11**, 181–204, 1991.

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. *Journal of the Institute of Mathematics and its Applications*, **23**, 235–250, 1979.

A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. J. Wiley and Sons, New-York, 1968. Reprinted as *Classics in Applied Mathematics 4*, SIAM, 1990.

R. Fletcher. Factorizing symmetric indefinite matrices. *Linear Algebra and its Applications*, **14**, 257–272, 1976.

R. Fletcher. *Practical Methods of Optimization*. J. Wiley and Sons, Chichester, second edn, 1987.

A. Forsgren, P. E. Gill, and W. Murray. Computing modified Newton directions using a partial Cholesky factorization. *SIAM Journal on Scientific Computing*, **16**, 139–150, 1995.

A. L. Forsgren and W. Murray. Newton methods for large-scale linear equality-constrained minimization. *SIAM Journal on Matrix Analysis and Applications*, **14**(2), 560–587, 1993.

R. Fourer and S. Mehrotra. Solving symmetric indefinite systems for an interior point method for linear programming. *Mathematical Programming*, **62**(1), 15–40, 1993.

K. R. Frisch. The logarithmic potential method of convex programming. Memorandum of May 13, University Institute of Economics, Oslo, Norway, 1955.

D. M. Gay. Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing*, **2**, 186–197, 1981.

A. George and J. W. H. Liu. *Computer solution of large sparse positive definite systems*. Prentice-Hall, Englewood Cliffs, USA, 1981.

J. R. Gilbert and M. T. Heath. Computing a sparse basis for the null-space. *SIAM Journal on Algebraic and Discrete Methods*, **8**, 446–459, 1987.

P. E. Gill and W. Murray. Newton-type methods for unconstrained and linearly constrained optimization. *Mathematical Programming*, **28**, 311–350, 1974.

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

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

P. E. Gill, W. Murray, D. B. Ponceléon, and M. A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM Journal on Matrix Analysis and Applications*, **13**, 292–311, 1992.

P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Shifted barrier methods for linear programming. Technical Report SOL88-9, Department of Operations Research, Stanford University, Stanford, California 94305, USA, 1988.

P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. A Schur complement method for sparse quadratic programming. *In* M. G. Cox and S. Hammarling, eds, 'Reliable Numerical Computation', pp. 113–138, Clarendon Press, Oxford, 1990.

D. Goldfarb. Curvilinear path steplength algorithms for minimization which use directions of negative curvature. *Mathematical Programming*, **18**, 31–40, 1980.

N. I. M. Gould. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem. *Mathematical Programming*, **32**, 90–99, 1985.

J. Greenstadt. On the relative efficiencies of gradient methods. *Mathematics of Computation*, **21**, 360–367, 1967.

Harwell Subroutine Library. *A catalogue of subroutines (release 10)*. Advanced Computing Department, Harwell Laboratory, Harwell, UK, 1990.

M. D. Hebden. An algorithm for minimization using exact second derivatives. Technical Report T. P. 515, AERE Harwell Laboratory, Harwell, UK, 1973.

M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, **4**, 303–320, 1969.

K. Jittorntrum and M. Osborne. A modified barrier function method with improved rate of convergence for degenerate problems. *Journal of the Australian Mathematical Society (Series B)*, **21**, 305–329, 1980.

L. S. Lasdon, R. L. Fox, and M. W. Ratner. An efficient one-dimensional search procedure for barrier functions. *Mathematical Programming*, **4**(3), 279–296, 1973.

I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and its Applications*, **152**, 191–222, 1991.

I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM Journal on Optimization*, **2**(1), 435–449, 1992.

H. M. Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, **3**, 255–269, 1957.

S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, **2**(4), 575–601, 1992.

J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. *In* G. A. Watson, ed., 'Proceedings Dundee 1977', Springer Verlag, Berlin, 1978. Lecture Notes in Mathematics.

J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, **16**, 1–20, 1979.

W. Murray and M. H. Wright. Line search procedures for the logarithmic barrier function. *SIAM Journal on Optimization*, **4**(2), 229–246, 1994.

B. A. Murtagh and M. A. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, **14**, 41–72, 1978.

B. A. Murtagh and M. A. Saunders. MINOS 5. 1 USER'S GUIDE. Technical Report SOL83-20R, Department of Operations Research, Stanford University, Stanford, USA, 1987.

S. G. Nash. Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, **21**(4), 1984.

R. Polyak. Modified barrier functions (theory and methods). *Mathematical Programming*, **54**(2), 177–222, 1992.

M. J. D. Powell. A method for nonlinear constraints in minimization problems. *In* R. Fletcher, ed., 'Optimization', pp. 283–298, Academic Press, London and New York, 1969.

T. Schlick. Modified Cholesky factorizations for sparse preconditioners. *SIAM Journal on Scientific and Statistical Computing*, **14**(2), 424–445, 1993.

R. B. Schnabel and E. Eskow. A new modified Cholesky factorization. *SIAM Journal on Scientific and Statistical Computing*, **11**, 1136–1158, 1991.

D. C. Sorensen. Newton's method with a model trust-region modification. Technical Report ANL-80-106, Argonne National Laboratory, Argonne, USA, 1980.

J. M. Stern and S. A. Vavasis. Nested dissection for sparse nullspace bases. *SIAM Journal on Matrix Analysis and Applications*, **14**, 766–775, 1993.

G. W. Stewart. Modifying pivot elements in gaussian elimination. *Mathematics of Computation*, **28**(126), 537–542, 1967.

R. J. Vanderbei and T. J. Carpenter. Indefinite systems for interior point methods. *Mathematical Programming*, **58**(1), 1–32, 1993.