IMA Journal of Numerical Analysis (1991) **11**, 299–324

An Algorithm for Large-Scale Quadratic Programming

NICHOLAS I. M. GOULD

Central Computer Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, UK

[Received 27 June 1989 and in revised form 3 September 1990]

We describe a method for solving large-scale general quadratic programming problems. Our method is based upon a compendium of ideas which have their origins in sparse matrix techniques and methods for solving smaller quadratic programs. We include a discussion on resolving degeneracy, on single phase methods and on solving parametric problems. Some numerical results are included.

1. Introduction

In this paper we describe a method for solving the quadratic programming problem

$$\text{minimize } Q(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \quad (\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{H} = \boldsymbol{H}^{\mathsf{T}}), \tag{1.1a}$$

$$\text{subject to } \boldsymbol{a}_i^{\mathsf{T}}\boldsymbol{x} = b_i, \quad 1 \leqslant i \leqslant m_{\mathrm{e}} \tag{1.1b}$$

$$\text{and } \boldsymbol{a}_i^{\mathsf{T}}\boldsymbol{x} \geqslant b_i, \quad m_{\mathrm{e}} + 1 \leqslant i \leqslant m. \tag{1.1c}$$

We are particularly concerned in solving (1.1) when n is large and the vectors $\boldsymbol{a}_i$ and matrix $\boldsymbol{H}$ are sparse. We do not restrict $\boldsymbol{H}$ to being positive (semi-)definite and consequently are content with finding local solutions to (1.1). Of course, for many classes of problem, it is known a priori that any local solution is a global one. Our method is fundamentally related to that proposed by Fletcher (1971), but makes use of sparse matrix technology (in particular, linear programming basis handling techniques) to exploit the nature of the problem.

In Section 2, we describe a general framework for our method. Linear algebraic issues are considered in Section 3 together with a description of how these issues relate to solving more specific quadratic programming problems of the form

$$\text{minimize } Q(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^{\mathsf{T}}\boldsymbol{H}\boldsymbol{x} + \boldsymbol{c}^{\mathsf{T}}\boldsymbol{x} \quad (\boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{H} = \boldsymbol{H}^{\mathsf{T}}), \tag{1.2a}$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1.2b}$$

$$\text{and } \boldsymbol{l} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}, \tag{1.2c}$$

in which $\boldsymbol{A}$ is an m by n matrix, $\boldsymbol{b}$ is an m-vector, and $\boldsymbol{l}$ and $\boldsymbol{u}$ are n-vectors whose components may be infinite. This latter description is important as it forms the basis of the algorithm implemented as subroutine VE09 in the Harwell Subroutine Library. The construction of initial feasible points for (1.2) is described in Section 4 and a discussion of parametric problems follows in Section

5. Finally some numerical results are given in Section 6. An appendix is included to show how a certain set of anti-cycling rules may be applied to problem (1.1).

We let $g(x)$ denote $Hx + c$, the gradient of $Q(x)$, let I be the (appropriately dimensioned) identity matrix and let e_k denote its kth column.

2. The basic algorithm

In this section we describe the algorithm of Fletcher (1971) in a general setting. Our aim is to isolate the important parts of Fletcher's method and to show how each stage is related to solving a particular set of linear equations. Methods for solving such equations will not be considered until Section 3.

In his paper, Fletcher describes an active-set algorithm for solving (1.1). He comments that he hesitates to call his method 'new' as it freely borrows from existing ideas in other branches of mathematical programming. We should add that the method described here is even less 'new' as it borrows quite heavily from Fletcher's method and from the sparse matrix technology that has arisen in the 1970s and 80s. Indeed, most existing algorithms for convex quadratic programming (including Fletcher's method and the method given here when applied to convex problems) are theoretically identical (in exact arithmetic they all generate the same sequence of iterates from a given starting point) provided that they operate under an identical set of pivoting rules (Best, 1984; Djang, 1979). For the general non-convex problem, the method of Gill & Murray (1978) is closely related in philosophy to Fletcher's method; this method and the philosophically different method of Keller (1973) actually generate the same sequence of iterates as Fletcher's approach under identical pivoting rules. The differences between the various methods for solving (1.1) are essentially restricted to the manner in which the set(s) of linear equations that occur at each iteration are solved.

In order to describe the algorithmic framework in a general setting, we need the following definitions and observations.

DEFINITION 2.1 An *equality problem* EP(I) is given by

$$\text{minimize} \quad Q(x) \quad \text{subject to} \quad a_i^T x = b_i, \quad i \in I \tag{2.1}$$

for the set $I = E \cup A$, where $E = \{1,...,m_e\}$ and $A \subseteq \{m_e + 1,...,m\}$.

Given a point x that satisfies $a_i^T x = b_i$ for all $i \in I$, any finite solution $x + p$ to EP(I) must satisfy the first-order optimality conditions

$$Hp + \hat{A}^T \lambda = -g(x), \tag{2.2a}$$

$$\hat{A}p = 0, \tag{2.2b}$$

where $\hat{A}$ is the matrix whose rows are a_i^T ($i \in I$), and the vector λ is a vector of Lagrange multipliers for EP(I).

DEFINITION 2.2 A matrix $K(I)$ of the form

$$K(I) = \begin{bmatrix} H & \hat{A}^T \\ \hat{A} & 0 \end{bmatrix} \tag{2.3}$$

is said to satisfy the *second-order condition* (*SOC*) if $\boldsymbol{K}(\boldsymbol{I})$ is non-singular and has exactly $|\boldsymbol{I}|$ negative eigenvalues.

This property has a number of alternative interpretations (Gould, 1985); in particular, it is equivalent to $\hat{\boldsymbol{A}}$ having full rank and $\boldsymbol{p}^{\mathsf{T}}\boldsymbol{H}\boldsymbol{p}$ being strictly positive for all nonzero vectors $\boldsymbol{p}$ satisfying $\hat{\boldsymbol{A}}\boldsymbol{p} = \boldsymbol{0}$ and thus to the projected or reduced Hessian matrix (see, for example, Gill, Murray, & Wright, 1981) being positive definite. If $\boldsymbol{K}(\boldsymbol{I})$ satisfies SOC and $\boldsymbol{p}$ satisfies equation (2.2), the solution of problem (2.1) is unique and occurs at $x + p$. If $\boldsymbol{H}$ is positive definite and $\hat{\boldsymbol{A}}$ has full rank, $\boldsymbol{K}(\boldsymbol{I})$ always satisfies SOC.

DEFINITION 2.3 A *feasible solution to an equality problem* (FSEP) is a solution $\boldsymbol{x}(\boldsymbol{I})$ to EP($\boldsymbol{I}$) for which $\boldsymbol{a}_i^{\mathsf{T}}\boldsymbol{x} \geq \boldsymbol{b}_i$ holds for all i not in $\boldsymbol{I}$.

The aim of Fletcher's algorithm is to construct a sequence of FSEPs with decreasing objective function values. The algorithm terminates when an FSEP without any positive Lagrange multipliers is obtained. If the multipliers are all strictly negative or if $\boldsymbol{H}$ is positive semi-definite, such a point will be a local solution to (1.1). If $\boldsymbol{H}$ is indefinite and there are some zero multipliers, the point obtained may not be a local solution to (1.1); however, as Fletcher points out, such a situation may be regarded as degenerate in that an arbitrarily small perturbation to the problem data can transform the FSEP into a local solution. The difficulty of verifying optimality in this circumstance is considerable; the optimality conditions for (1.1), when there are zero Lagrange multipliers, are computationally unattractive (see, for example, Mangasarian, 1980) in that they are equivalent to needing to find the *global* solution to a second (closely related) nonconvex quadratic program. To our knowledge, no existing algorithm is able to overcome this difficulty in an efficient manner.

As there are only a finite number of FSEPs and, as we shall indicate, moving from one to the next is a finite process, the overall algorithm is finite. It remains to describe how to proceed from one FSEP to a better one.

The idea is very simple. Suppose $\boldsymbol{x}^{(k)}$ is a feasible point for (1.1) and

$$\boldsymbol{a}_i^{\mathsf{T}}\boldsymbol{x}^{(k)} = \boldsymbol{b}_i \quad \text{for all } i \in \boldsymbol{I}^{(k)} = \boldsymbol{E} \cup \boldsymbol{A}^{(k)}.$$

If $\boldsymbol{x}^{(k)}$ is an FSEP for EP($\boldsymbol{I}^{(k)}$), a constraint must be removed from $\boldsymbol{I}^{(k)}$ to reduce $Q(\boldsymbol{x})$ further—candidates occur in $\boldsymbol{A}^{(k)}$ and are identified by having positive Lagrange multipliers. Any one such constraint, say j, may be removed and we set $\bar{\boldsymbol{I}} = \boldsymbol{I}^{(k)} \backslash \{j\}$. Otherwise, when $\boldsymbol{x}^{(k)}$ is not an FSEP for EP($\boldsymbol{I}^{(k)}$), we set $\bar{\boldsymbol{I}} = \boldsymbol{I}^{(k)}$.

Now we solve EP($\bar{\boldsymbol{I}}$) in the sense that either

(a) we find a vector $\boldsymbol{p}^{(k)}$ for which $\boldsymbol{x}^{(k)} + \boldsymbol{p}^{(k)}$ solves EP($\bar{\boldsymbol{I}}$), or

(b) we determine a direction of infinite descent $\boldsymbol{p}^{(k)}$ so that for all $\alpha \geq 0$

$$\begin{aligned} &\boldsymbol{a}_i^{\mathsf{T}}(\boldsymbol{x}^{(k)} + \alpha\boldsymbol{p}^{(k)}) = \boldsymbol{b}_i \quad \text{for all } i \in \bar{\boldsymbol{I}}, \\ &\boldsymbol{a}_i^{\mathsf{T}}(\boldsymbol{x}^{(k)} + \alpha\boldsymbol{p}^{(k)}) \geq \boldsymbol{b}_j, \end{aligned} \tag{2.4}$$

and so that $Q(\boldsymbol{x}^{(k)} + \alpha\boldsymbol{p}^{(k)})$ is a monotonically decreasing function of α. (This case can only occur when $\boldsymbol{H}$ is not positive definite.)

There are now three possibilities. Firstly, if case (a) above occurs, $\boldsymbol{x}^{(k)} + \boldsymbol{p}^{(k)}$ may be an FSEP. We then set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{p}^{(k)}$ and $\boldsymbol{I}^{(k+1)} = \bar{\boldsymbol{I}}$. Secondly, if case

(b) occurs, $x^{(k)} + \alpha p^{(k)}$ might satisfy (1.1c) for *all* $\alpha > 0$. In this case, the problem is unbounded and we stop. Finally, if neither of these situations occur, there must be a largest value $\bar{\alpha}$ for which $x^{(k)} + \alpha p^{(k)}$ satisfies (1.1c). If the lth constraint is one of the constraints which imposes this upper bound on α, we set $x^{(k+1)} = x^{(k)} + \bar{\alpha} p^{(k)}$ and $I^{(k+1)} = \bar{I} \cup \{l\}$. It is important to observe that in this last case

$$a_l^T p^{(k)} < 0. \tag{2.5}$$

We now repeat the procedure with k incremented by one.

The above process is finite and, provided an unbounded direction is not found, must terminate with an FSEP. This is because every time $x^{(k)} + p^{(k)}$ fails to be an FSEP, we add to $I^{(k)}$ a constraint whose gradient is independent of those already indexed by $I^{(k)}$. If this adding continues we will eventually either exhaust all of the constraints (1.1c) or arrive at an extreme point of the feasible region (which is automatically an FSEP).

Complications arise when H is not positive definite, as it is then not obvious which of the two cases (a) or (b) will occur. However, whenever $K(\bar{I})$ satisfies SOC, only case (a) is possible. The novel part of Fletcher's algorithm is that, at each iteration, the iterate $x^{(k)}$ is associated with an equality problem EP($\bar{I}^{(k)}$) for which $K(\bar{I}^{(k)})$ satisfies SOC. The essence of the algorithm is contained in the following theorems (the proofs are straightforward and are omitted).

THEOREM 2.1 *Suppose* $K(I)$ *satisfies* SOC, x *satisfies the constraints* $a_i^T x = b_i$, *for all* $i \in I$, *and that* p *and* λ *satisfy the equations*

$$\begin{bmatrix} H & \hat{A}^T \\ \hat{A} & 0 \end{bmatrix} \begin{bmatrix} p \\ \lambda \end{bmatrix} = \begin{bmatrix} -g(x) \\ 0 \end{bmatrix}. \tag{2.6}$$

Then $x + p$ *solves* EP(I) *and* λ *is the vector of Lagrange multipliers for* EP(I).

THEOREM 2.2 *Suppose* $K(I)$ *satisfies* SOC, there is a vector p for which $\hat{A}p = 0$ *and* $a_l^T p \neq 0$, *and that* $I_{+l} = I \cup \{l\}$. *Then* $K(I_{+l})$ *satisfies* SOC.

THEOREM 2.3 *Suppose* $K(I)$ *satisfies* SOC, x *solves* EP(I), x *is an* FSEP, *and that the Lagrange multiplier* λ_j *associated with constraint* j *of* EP(I) *is strictly positive. Let* $I_{-j} = I \backslash \{j\}$, *let* $\hat{A}_{-j}$ *denote* $\hat{A}$ *with the row* a_j^T *removed,* λ_{-j} *denote* λ *with* λ_j *removed and let* p, q, *and* μ *satisfy the equation*

$$\begin{bmatrix} H & \hat{A}_{-j}^T & a_j \\ \hat{A}_{-j} & 0 & 0 \\ a_j^T & 0 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ \mu \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{2.7}$$

then

(i) $\mu = -p^T H p$;

(ii) *if* $\mu < 0$ *and* $\hat{\alpha} = \lambda_j / p^T H p$, *then* $K(I_{-j})$ *satisfies* SOC, $x + \hat{\alpha} p$ *solves* EP(I_{-j}) *and* $\lambda_{-j} + \hat{\alpha} q$ *is the corresponding vector of Lagrange multipliers*;

(iii) *if* $\mu \geq 0$, *then* $a_i^T(x + \alpha p) = b_i$, *for all* $i \in I_{-j}$, $a_j^T(x + \alpha p) \geq b_j$, *and* $Q(x + \alpha p)$ *is a monotonically decreasing function of* α *for all* $\alpha \geq 0$. *However* $K(I_{-j})$ *does not satisfy* SOC;

(iv) $K(I_{-j})$ *is singular if and only if* $\mu = 0$.

Now suppose that $x + \alpha p$ *becomes infeasible for* $\alpha > \bar{\alpha}$ *because constraint* l *is*

violated. Let $\boldsymbol{u}$, $\boldsymbol{v}$, *and* θ *satisfy the equation*

$$\begin{bmatrix} \boldsymbol{H} & \hat{\boldsymbol{A}}_{-j}^{\mathsf{T}} & \boldsymbol{a}_j \\ \hat{\boldsymbol{A}}_{-j} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{a}_j^{\mathsf{T}} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \theta \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_l \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tag{2.8}$$

and let $\boldsymbol{I}_{-j+l} = \boldsymbol{I}_{-j} \cup \{l\}$, *then*

(v) $\theta = \boldsymbol{a}_l^{\mathsf{T}}\boldsymbol{p} < 0$;

(vi) *if* $\boldsymbol{u} = 0$ *and if* $\bar{\alpha} < \hat{\alpha}$ *when* $\mu < 0$, *then* $\boldsymbol{K}(\boldsymbol{I}_{-j+l})$ *satisfies* SOC, $\boldsymbol{x} + \bar{\alpha}\boldsymbol{p}$ *solves* EP($\boldsymbol{I}_{-j+l}$) *(and is therefore an* FSEP), λ_l, *the Lagrange multiplier for constraint* l, *is* $(\lambda_j + \bar{\alpha}\mu)/\theta$ *and the remaining Lagrange multipliers are* $\boldsymbol{\lambda}_{-j} + \bar{\alpha}\boldsymbol{q} - \lambda_l\boldsymbol{v}$;

(vii) *if* $\boldsymbol{u} \neq 0$ *and* $\boldsymbol{I}_{+l} = \boldsymbol{I} \cup \{l\}$ *then* $\boldsymbol{K}(\boldsymbol{I}_{+l})$ *satisfies* SOC;

(viii) $\boldsymbol{u} = 0$ *if and only if* $\boldsymbol{a}_l^{\mathsf{T}}\boldsymbol{u} = 0$.

For future reference (Section 5) we also state the following:

THEOREM 2.4 *Suppose that the assumptions made at the start of Theorem* 2.3 *are satisfied excepting that the Lagrange multiplier* λ_j *associated with constraint* j *of* EP($\boldsymbol{I}$) *is zero. Then*

(i) *if* $\mu < 0$ *then* $\boldsymbol{K}(\boldsymbol{I}_{-j})$ *satisfies* SOC, $\boldsymbol{x}$ *solves* EP($\boldsymbol{I}_{-j}$) *(and is therefore an* FSEP), *and* $\boldsymbol{\lambda}_{-j}$ *is the corresponding vector of Lagrange multipliers*;

(ii) *if* $\mu = 0$ *then* $\boldsymbol{x} + \alpha\boldsymbol{p}$ *solves* (2.2) *for all* $\alpha \geq 0$ *for which* $\boldsymbol{x} + \alpha\boldsymbol{p}$ *remains feasible. The remaining Lagrange multipliers are* $\boldsymbol{\lambda}_{-j} + \alpha\boldsymbol{q}$;

(iii) *if* $\mu > 0$ *then* $\boldsymbol{a}_i^{\mathsf{T}}(\boldsymbol{x} + \alpha\boldsymbol{p}) = b_i$ *for all* $i \in \boldsymbol{I}_{-j}$, $\boldsymbol{a}_j^{\mathsf{T}}(\boldsymbol{x} + \alpha\boldsymbol{p}) \geq b_j$, *and* $Q(\boldsymbol{x} + \alpha\boldsymbol{p})$ *is a monotonically decreasing function of* α *for all* $\alpha \geq 0$.

In addition, conclusions (v)–(viii) *of Theorem* 2.3 *remain true under the current hypotheses.*

Theorems 2.1 and 2.2 tell us that once we have encountered a set $\boldsymbol{I}$ for which $\boldsymbol{K}(\boldsymbol{I})$ satisfies SOC, then we will eventually discover an FSEP for which the corresponding matrix $\boldsymbol{K}$ must satisfy SOC. Theorem 2.3 describes the only circumstances under which we may lose the second-order condition; we delete a constraint from $\boldsymbol{I}$ and encounter a direction of infinite descent ($\mu \geq 0$). Once we have lost the second-order condition, it would appear to be more difficult to solve EP($\boldsymbol{I}_{-j}$). However, Fletcher observes that if the circumstances described by (iii) of Theorem 2.3 are satisfied then either the problem is unbounded from below or one or more constraints will restrict the step α for which $\boldsymbol{x} + \alpha\boldsymbol{p}$ satisfies (1.1c). If the lth constraint is one such constraint and if α is restricted by this constraint to be at most $\bar{\alpha}$, then $\boldsymbol{x} + \bar{\alpha}\boldsymbol{p}$ satisfies (1.1c) and equation (2.5) holds. Hence either condition (vi) or condition (vii) of the theorem is satisfied. In the former case, the second-order condition is maintained for the set $\boldsymbol{I}_{-j+l}$. In the latter case, $\boldsymbol{x} + \bar{\alpha}\boldsymbol{p}$ is an FSEP for the equality problem

$$\text{minimize} \quad Q(\boldsymbol{x}) \tag{2.9a}$$

$$\text{subject to} \quad \boldsymbol{a}_k^{\mathsf{T}}\boldsymbol{x} = b_k, \quad k \in \boldsymbol{I}_{+l} \backslash \{j\} \tag{2.9b}$$

$$\text{and} \quad \boldsymbol{a}_j^{\mathsf{T}}\boldsymbol{x} = b_j + \bar{\alpha}, \tag{2.9c}$$

for which the second-order condition is satisfied. Moreover, the Lagrange multiplier corresponding to the constraint (2.9c) for problem (2.9) has the value $\lambda_j + \bar{\alpha}\mu \geq \lambda_j > 0$ and is therefore a candidate for deletion (applying Theorem 2.3 to (2.9); note that the remaining nonzero multipliers for (2.9) are $\lambda_{-j} + \bar{\alpha}q$). But then the resulting equality problem, EP($I_{+l}\backslash\{j\}$) is the *same* problem as the problem EP($I_{-j} \cup \{l\}$) which would have arisen naturally by just removing constraint j from I and then adding constraint l; the *difference* is that, by going by way of the set I_{+l}, we are able to maintain the second-order condition and to calculate the solution to EP($I_{+l}\backslash\{j\}$).

Repeating this argument from $x + \bar{\alpha}p$, we must eventually arrive in the position where one of the alternatives (ii) or (vi) of Theorem 2.3 occurs. (As the sets I_{+l} grow in size, then alternative (vi), which automatically occurs when $|I| = n$, will occur unless (ii) happens first.) In case (vi) we will have found an FSEP which satisfies SOC. In all cases any new FSEP encountered must satisfy SOC.

In order to guarantee that we do not generate a repeating sequence of FSEPs with the same objective function value, we must ensure that we occasionally take a nonzero step. Consideration of Theorem 2.3 indicates that it is sufficient to ensure that an infinite consecutive occurrence of case (vi) (a *cycle*) cannot happen. (All other cases result in either a nonzero step or an increase in the size of I and hence a move to an FSEP which has not been encountered before.) The simplest possible method for ensuring that a cycle cannot happen is to perturb the right-hand sides b_i by small random amounts (such as might be introduced naturally in a finite precision calculation) whenever case (vi) of the theorem occurs. Other possibilities include replacing the true objective function by its linearized approximation $g(x_0)^T x$ at the degenerate point x_0 and using anti-cycling rules from linear programming (see Wolfe, 1963; Fletcher, 1988; Gill *et al.*, 1989) to find a direction in which the approximation, and hence $Q(x)$, may be reduced in a feasible neighbourhood of x_0. This approach is somewhat unsatisfactory in that matrix operators/factorizations for a different problem may be required. Alternatively, anti-cycling rules which take account of the quadratic nature of the objective function have been suggested in conjunction with certain quadratic programming methods (see, for example, Chang & Cottle, 1980; Ritter, 1981). In the appendix, we establish that the least index rule proposed by Bland (1977) for use with the simplex method is equally appropriate within our general framework (this was conjectured, but not proved, in the paper by Chang & Cottle).

Finally, the algorithm requires that we start from an FSEP for which the second-order condition holds. It is, of course, by no means obvious how one can obtain even a feasible point in general. Fletcher suggests using a phase-one linear programming method to find a feasible vertex (at which the second-order condition must hold and which must necessarily be an FSEP). If there are fewer than n linearly independent constraints it will be necessary to introduce some artificial bounds to create a vertex to use this approach. We prefer to use a single-phase method (in which an artificial objective function contrasting constraint infeasibilities and the true objective is minimized), but a description of this is considerably simplified in the context of a reformulation of problem (1.1) and will be left until Section 4.

3. Linear algebra

Considerable ingenuity has been used in finding different ways of solving matrix equations of the type (2.6), (2.7), and (2.8) which arise at each iteration of the algorithm described in Section 2 (see, for instance, Van de Panne & Whinston, 1969; Bartels, Golub, & Saunders, 1970; Fletcher, 1971; Murray, 1971; Best & Ritter, 1976; Gill & Murray, 1978; Bunch & Kaufman, 1980; Powell, 1981; Goldfarb & Idnani, 1983; Gill *et al.*, 1984). Often smaller systems of equations which take account of the structure of the matrix $\boldsymbol{K}(\boldsymbol{I})$ are obtained; the equations are usually solved by means of matrix factorizations or, in earlier methods (including Fletcher's implementation of the algorithm of Section 2), explicit matrix inverses. As consecutive sets $\boldsymbol{I}$ are closely related, it is normal to *update* these factorizations or inverses rather than recomputing them each iteration.

In the large-sparse context, we would normally think of using methods for which the resulting systems of equations have coefficient matrices

(a) which are comparable in sparsity with the problem matrices $\boldsymbol{H}$ and $\boldsymbol{A}$,

(b) for which sparse factorization methods are applicable, and

(c) for which it is possible to update the factorization in an efficient manner when the set $\boldsymbol{I}$ *changes.*

We would not usually think of using matrix inverses in this context as they are invariably dense. Unfortunately, most of the methods which have been developed for small problems (including those mentioned above) are not generally appropriate for larger problems as they violate requirement (a). The only really satisfactory methods for general sparse problems are likely to be related to methods which *do not* try to simplify equations (2.6)–(2.8) but form factorizations of the matrix $\boldsymbol{K}(\boldsymbol{I})$ itself. Such methods are likely to satisfy requirements (a) and (b). However requirement (c) poses more of a problem. The trouble with the matrix $\boldsymbol{K}(\boldsymbol{I})$ is that *both rows and columns* change as $\boldsymbol{I}$ is altered and it is this aspect which causes trouble when an existing sparse factorization is updated.

The Schur-complement update method proposed by Gill *et al.* (1985, 1987a) manages to avoid this problem by maintaining a sparse factorization of the initial matrix $\boldsymbol{K}(\boldsymbol{I}^{(0)})$ and a dense factorization of the Schur complement (see, for example, Golub & Van Loan, 1983) of $\boldsymbol{K}(\boldsymbol{I}^{(0)})$ in a growing matrix of the form

$$\begin{bmatrix} \boldsymbol{K}(\boldsymbol{I}^{(0)}) & \boldsymbol{E}^{\mathrm{T}} \\ \boldsymbol{E} & \boldsymbol{F} \end{bmatrix}.$$

Equations (2.6)–(2.8) and their solutions may be embedded in linear systems whose coefficient matrices are precisely of this form. The matrix $\boldsymbol{E}$ is made up from appropriate rows of the identity matrix and vectors $\boldsymbol{a}_i^{\mathrm{T}}$. $\boldsymbol{F}$ is symmetric and made up of rows with at most one nonzero entry. Both $\boldsymbol{K}(\boldsymbol{I}^{(0)})$ and its Schur complement are therefore symmetric and this may be exploited when forming factorizations. Updating the dense factorization of the Schur complement is not difficult provided that sufficient storage space is available. However, the constant need to access $\boldsymbol{E}$ can be inconvenient; furthermore, it will be necessary to refactorize $\boldsymbol{K}(\boldsymbol{I})$ whenever the Schur complement becomes too large to store economically. This approach seems to hold considerable promise and, indeed, has

already been used successfully to solve very large convex problems arising in the electrical power industry (P. E. Gill, private communication). The adaptation to nonconvex problems using the framework of Section 2 should not prove very difficult. The paper of Gill *et al.* (1987a) should be consulted for further details.

More recently, Gill *et al.* (1987b) have described a method of updating a sparse unsymmetric factorization of a matrix when both rows and/or columns are added, deleted, or replaced. Such a tool is extremely valuable and, again, its use in conjunction with the framework of Section 2 is anticipated. It has the slightly unfortunate feature that, although symmetric changes are made to $\boldsymbol{K}(\boldsymbol{I})$ as $\boldsymbol{I}$ changes, the updating procedures for its factorization ignore this symmetry—the updates are treated as a row change *followed* by a column change (or vice versa). This is almost certainly a consequence of maintaining an unsymmetric factorization of a symmetric matrix.

Ideally, we should like to maintain a symmetric sparse factorization of $\boldsymbol{K}(\boldsymbol{I})$ and to perform symmetric updates as $\boldsymbol{I}$ changes. Although it is possible to find such a factorization (see Duff & Reid, 1983), the possibility of being able to update it appears remote; even in the dense case, when the Duff–Reid factorization reduces to the symmetric indefinite factorization of Bunch & Parlett (1971) and where an updating scheme is known (Sorensen, 1977), the updates are extremely complicated and appear not to adapt to the sparse case.

All of the equations (2.6)–(2.8) are of the generic form

$$\begin{bmatrix} \boldsymbol{H} & \hat{\boldsymbol{A}}^{\mathsf{T}} \\ \hat{\boldsymbol{A}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{q} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{r} \end{bmatrix}. \tag{3.1}$$

In the method that we propose, we suppose that $\bar{\boldsymbol{A}}$ represents the matrix whose rows are the vectors $\boldsymbol{a}_k^{\mathsf{T}}$ for $k \notin \boldsymbol{I}$. Then, on defining $\boldsymbol{s} = -\bar{\boldsymbol{A}}\boldsymbol{p}$, (3.1) can be expanded to give

$$\begin{bmatrix} \boldsymbol{H} & \hat{\boldsymbol{A}}^{\mathsf{T}} & \boldsymbol{0} \\ \hat{\boldsymbol{A}} & \boldsymbol{0} & \boldsymbol{0} \\ \bar{\boldsymbol{A}} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{q} \\ \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{r} \\ \boldsymbol{0} \end{bmatrix} \tag{3.2}$$

and we shall refer to (3.2) as the *expanded* version of (3.1). The advantage of (3.2) over (3.1) is that as $\boldsymbol{I}$ changes, with the exception of simple row permutations, only the *columns* of the coefficient matrix

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{H} & \hat{\boldsymbol{A}}^{\mathsf{T}} & \boldsymbol{0} \\ \hat{\boldsymbol{A}} & \boldsymbol{0} & \boldsymbol{0} \\ \bar{\boldsymbol{A}} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{3.3}$$

change. This is the situation normally associated with linear programming basis matrices and there are extremely powerful methods for dealing with the changes which occur if an LU factorization of $\boldsymbol{B}$ is maintained (see, for example, Saunders, 1976; Reid, 1982). We shall refer to $\boldsymbol{B}$ as the *extended basis matrix*. The principal disadvantages of using (3.2) in comparison with (3.1) are that $\boldsymbol{B}$ is not symmetric and may be significantly larger than $\boldsymbol{K}(\boldsymbol{I})$. The hope is that the advantages of using a proven factorization updating technique will outweigh such

disadvantages. We note, however, that the work involved in finding a sparse LU factorization of $\boldsymbol{B}$ using many common pivotal strategies (such as the Markowitz test) will naturally pick the last diagonal block of (3.3) first. Thereafter the factorization will actually be working upon the remaining block of $\boldsymbol{B}$ which is actually $\boldsymbol{K}(\boldsymbol{I})$. The work will therefore be comparable with schemes which just factorize $\boldsymbol{K}(\boldsymbol{I})$. There is some evidence that ignoring symmetry can actually be advantageous when forming sparse factorizations (see Duff, Erisman, & Reid, 1986: p. 241; George, 1974).

Most real-life problems include simple bound constraints of the form

$$\boldsymbol{l} \leqslant \boldsymbol{x} \leqslant \boldsymbol{u}. \tag{3.4}$$

We do not necessarily assume that all (or any) of the bounds in (3.4) are finite. We now consider how the structure of such simple constraints might be exploited. For simplicity, we shall assume that the first k such constraints are contained in $\boldsymbol{I}$. If we partition the vectors and matrices in the obvious way, (3.2) may be rewritten as

$$\begin{bmatrix} \boldsymbol{H}_{\mathrm{FX}} & \boldsymbol{H}_{\mathrm{OD}} & \hat{\boldsymbol{A}}_{\mathrm{FX}}^{\mathrm{T}} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{H}_{\mathrm{OD}}^{\mathrm{T}} & \boldsymbol{H}_{\mathrm{FR}} & \hat{\boldsymbol{A}}_{\mathrm{FR}}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \hat{\boldsymbol{A}}_{\mathrm{FX}} & \hat{\boldsymbol{A}}_{\mathrm{FR}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \bar{\boldsymbol{A}}_{\mathrm{FX}} & \bar{\boldsymbol{A}}_{\mathrm{FR}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_{\mathrm{FX}} \\ \boldsymbol{p}_{\mathrm{FR}} \\ \boldsymbol{q}_{\mathrm{GC}} \\ \boldsymbol{q}_{\mathrm{BC}} \\ \boldsymbol{s}_{\mathrm{GC}} \\ \boldsymbol{s}_{\mathrm{BC}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{\mathrm{FX}} \\ \boldsymbol{b}_{\mathrm{FR}} \\ \boldsymbol{r}_{\mathrm{GC}} \\ \boldsymbol{r}_{\mathrm{BC}} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}$$

(the suffices FX, FR, OD, GC, and BC are supposed to indicate 'FiXed', 'FRee', 'Off Diagonal', 'General Constraint', and 'Bound Constraint' respectively); this can be simplified to

$$\begin{bmatrix} \boldsymbol{H}_{\mathrm{OD}} & \hat{\boldsymbol{A}}_{\mathrm{FX}}^{\mathrm{T}} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{H}_{\mathrm{FR}} & \hat{\boldsymbol{A}}_{\mathrm{FR}}^{\mathrm{T}} & \boldsymbol{0} & \boldsymbol{0} \\ \hat{\boldsymbol{A}}_{\mathrm{FR}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \bar{\boldsymbol{A}}_{\mathrm{FR}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_{\mathrm{FR}} \\ \boldsymbol{q}_{\mathrm{GC}} \\ \boldsymbol{q}_{\mathrm{BC}} \\ \boldsymbol{s}_{\mathrm{GC}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_{\mathrm{FX}} - \boldsymbol{H}_{\mathrm{FX}}\boldsymbol{r}_{\mathrm{BC}} \\ \boldsymbol{b}_{\mathrm{FR}} - \boldsymbol{H}_{\mathrm{OD}}^{\mathrm{T}}\boldsymbol{r}_{\mathrm{BC}} \\ \boldsymbol{r}_{\mathrm{GC}} - \hat{\boldsymbol{A}}_{\mathrm{FX}}\boldsymbol{r}_{\mathrm{BC}} \\ -\bar{\boldsymbol{A}}_{\mathrm{FX}}\boldsymbol{r}_{\mathrm{BC}} \end{bmatrix}, \tag{3.5}$$

where $\boldsymbol{p}_{\mathrm{FX}} = \boldsymbol{r}_{\mathrm{BC}}$. Once again, only columns of the extended basis matrix associated with (3.5) change as $\boldsymbol{I}$ changes. Moreover the matrix for (3.5) is considerably smaller than that which would result if (3.2) were applied without taking special account of the simple bound constraints. However it is interesting to consider the types of column changes which can occur as $\boldsymbol{I}$ changes. In particular, when a general constraint or a simple bound is added to $\boldsymbol{I}$, a column of the coefficient matrix of (3.5) is replaced by a column of

$$\begin{bmatrix} \bar{\boldsymbol{A}}_{\mathrm{FX}}^{\mathrm{T}} \\ \bar{\boldsymbol{A}}_{\mathrm{FR}}^{\mathrm{T}} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \boldsymbol{H}_{\mathrm{FX}} \\ \boldsymbol{H}_{\mathrm{OD}}^{\mathrm{T}} \\ \hat{\boldsymbol{A}}_{\mathrm{FX}} \\ \bar{\boldsymbol{A}}_{\mathrm{FX}} \end{bmatrix}$$

respectively. Notice that, in the former case, access is needed to a row of A whereas, in the latter case, a column of A is needed. This can be rather awkward for large problems when A is stored in a packed form. If, however, the general constraints are all *equality* constraints and are included in the initial I, the only subsequent changes in I will be due to simple bound constraints and hence only access to the columns of A will be necessary.

It is for this reason that we now consider it convenient to concentrate on the problem

$$\text{minimize } Q(x) = \tfrac{1}{2}x^T H x + c^T x \quad (x \in \mathbb{R}^n, H = H^T), \tag{3.6a}$$

$$\text{subject to } Ax = b \tag{3.6b}$$

$$\text{and} \quad l \leq x \leq u. \tag{3.6c}$$

Indeed, the subroutine VE09 in the Harwell Subroutine Library assumes that problems are supplied in this form. We note that (1.1) may, of course, be converted to this form by adding slack variables to the inequality constraints (1.1c), although this inevitably increases the number of unknowns and may be undesirable if there are many general inequality constraints. For problem (3.6), our generic linear system becomes

$$\begin{bmatrix} A_{FX}^T & H_{OD} & I \\ A_{FR}^T & H_{FR} & 0 \\ 0 & A_{FR} & 0 \end{bmatrix} \begin{bmatrix} q_{GC} \\ p_{FR} \\ q_{BC} \end{bmatrix} = \begin{bmatrix} b_{FX} - H_{FX} r_{BC} \\ b_{FR} - H_{OD}^T r_{BC} \\ r_{GC} - A_{FX} r_{BC} \end{bmatrix} \tag{3.7}$$

and the extended basis matrix is now

$$B = \begin{bmatrix} A_{FX}^T & H_{OD} & I \\ A_{FR}^T & H_{FR} & 0 \\ 0 & A_{FR} & 0 \end{bmatrix}. \tag{3.8}$$

(We have reordered the variables and the rows of the extended basis matrix for convenience. Now only the last n columns of B can change with I.) It is convenient to store the data for the problem in the matrix

$$\begin{bmatrix} H & c \\ A & b \end{bmatrix}$$

stored in packed form by columns. Notice that it is still necessary to access the rows of A when the initial factorization (or any refactorization) of B is formed. However, this overhead will normally prove acceptable in comparison with the overall work needed to form the factorization. (There are efficient algorithms for changing from a column-wise to a row-wise storage scheme for a matrix A, requiring a small multiple of n_A, the number of nonzeros of A, operations—for instance, the code MC46 in the Harwell Subroutine Library performs this task in roughly $2n_A$ operations. Thus, the original matrix B may be formed efficiently in comparison with the number of operations—a linear or frequently worse function of n_B—required to find its factors.)

We now consider the work involved in updating an LU factorization of $\boldsymbol{B}$ as $\boldsymbol{I}$ changes. Each change will involve at most one element leaving $\boldsymbol{I}$ and one element being added (as described in Theorems 2.2 and 2.3). We must be careful with the order in which the computation is performed because, although the extended basis matrix at the end of each iteration is guaranteed to be nonsingular, the intermediate matrices need not be. In fact, this can only happen if possibility (iv) of Theorem 2.3 occurs. Our implementation uses subroutine LA05 (Reid, 1982) from the Harwell Subroutine Library as the mechanism for updating the factorization. Very briefly, if the kth column of $\boldsymbol{B}$, $\boldsymbol{b}_k$, is to be replaced by the vector $\bar{\boldsymbol{b}}$ to form the matrix $\bar{\boldsymbol{B}}$, we have the relationship

$$\bar{\boldsymbol{B}} = \boldsymbol{B} + (\bar{\boldsymbol{b}} - \boldsymbol{b}_k)\boldsymbol{e}_k^{\mathrm{T}} = \boldsymbol{B}(\boldsymbol{I} + (\boldsymbol{d} - \boldsymbol{e}_k)\boldsymbol{e}_k^{\mathrm{T}}), \tag{3.9}$$

where $\boldsymbol{Bd} = \bar{\boldsymbol{b}}$. If a factorization of $\boldsymbol{B}$ is known, the special form of the matrix

$$\boldsymbol{I} + (\boldsymbol{d} - \boldsymbol{e}_k)\boldsymbol{e}_k^{\mathrm{T}}$$

(differing from the identity matrix in just a single column) may be exploited so that the factors of $\boldsymbol{B}$ are easily updated to find those of $\bar{\boldsymbol{B}}$. If a constraint is being removed from $\boldsymbol{I}$, the linear system '$\boldsymbol{Bd} = \bar{\boldsymbol{b}}$' is the expanded version of (2.7); when a constraint is added to $\boldsymbol{I}$, the relevant system is the expanded version of (2.9). Thus the solutions to these systems actually play two roles; in determining the search direction (as in Theorem 2.3) and in updating the factorization of $\boldsymbol{B}$. Notice that when $\boldsymbol{B}$ is nonsingular, $\bar{\boldsymbol{B}}$ is nonsingular if and only if the kth entry of $\boldsymbol{d}$ is nonzero.

In the case of possibility (iv) of Theorem 2.3, it can happen that although the matrices $\boldsymbol{B}$ and

$$\begin{aligned} \breve{\boldsymbol{B}} &= \boldsymbol{B} + (\bar{\boldsymbol{b}}_1 - \boldsymbol{b}_k)\boldsymbol{e}_k^{\mathrm{T}} + (\bar{\boldsymbol{b}}_2 - \boldsymbol{b}_l)\boldsymbol{e}_l^{\mathrm{T}} \\ &= \boldsymbol{B}(\boldsymbol{I} + (\boldsymbol{d}_1 - \boldsymbol{e}_k)\boldsymbol{e}_k^{\mathrm{T}} + (\boldsymbol{d}_2 - \boldsymbol{e}_l)\boldsymbol{e}_l^{\mathrm{T}}) \end{aligned} \tag{3.10}$$

(corresponding to the sets $\boldsymbol{I}$ and $\breve{\boldsymbol{I}}$ respectively) are nonsingular, both of the intermediate matrices

$$\bar{\boldsymbol{B}}_1 = \boldsymbol{B} + (\bar{\boldsymbol{b}}_1 - \boldsymbol{b}_k)\boldsymbol{e}_k^{\mathrm{T}} \quad \text{and} \quad \bar{\boldsymbol{B}}_2 = \boldsymbol{B} + (\bar{\boldsymbol{b}}_2 - \boldsymbol{b}_l)\boldsymbol{e}_l^{\mathrm{T}} \tag{3.11}$$

are singular. Therefore, we must exercise some care in finding the factors of $\breve{\boldsymbol{B}}$; in particular we cannot simply use LA05 twice, putting $\bar{\boldsymbol{b}}_1$ in column k and then $\bar{\boldsymbol{b}}_2$ in column l. The singularity of $\bar{\boldsymbol{B}}_1$ and $\bar{\boldsymbol{B}}_2$ imply that the kth entry of $\boldsymbol{d}_1$ and the lth entry of $\boldsymbol{d}_2$ are zero. But the nonsingularity of $\breve{\boldsymbol{B}}$ implies that the matrix

$$\boldsymbol{I} + (\boldsymbol{d}_1 - \boldsymbol{e}_k)\boldsymbol{e}_k^{\mathrm{T}} + (\boldsymbol{d}_2 - \boldsymbol{e}_l)\boldsymbol{e}_l^{\mathrm{T}}$$

must be nonsingular. Hence the lth entry of $\boldsymbol{d}_1$ and the kth entry of $\boldsymbol{d}_2$ are necessarily nonzero and both of the matrices

$$\boldsymbol{B} + (\bar{\boldsymbol{b}}_2 - \boldsymbol{b}_k)\boldsymbol{e}_k^{\mathrm{T}} \quad \text{and} \quad \boldsymbol{B} + (\bar{\boldsymbol{b}}_1 - \boldsymbol{b}_l)\boldsymbol{e}_l^{\mathrm{T}} \tag{3.12}$$

are nonsingular. Noting that $\breve{\boldsymbol{B}} = \hat{\boldsymbol{B}}\boldsymbol{P}$, where

$$\begin{aligned} \hat{\boldsymbol{B}} &= \boldsymbol{B} + (\bar{\boldsymbol{b}}_2 - \boldsymbol{b}_k)\boldsymbol{e}_k^{\mathrm{T}} + (\bar{\boldsymbol{b}}_1 - \boldsymbol{b}_l)\boldsymbol{e}_l^{\mathrm{T}} \\ &= \boldsymbol{B}(\boldsymbol{I} + (\boldsymbol{d}_2 - \boldsymbol{e}_k)\boldsymbol{e}_k^{\mathrm{T}} + (\boldsymbol{d}_1 - \boldsymbol{e}_l)\boldsymbol{e}_l^{\mathrm{T}}) \end{aligned}$$

and $\boldsymbol{P}$ is the permutation matrix which interchanges columns k and l, it is clear that $\hat{\boldsymbol{B}}$ is nonsingular and that the factorization of $\hat{\boldsymbol{B}}$ can be obtained by way of the factors of one of the matrices in (3.12). That is, we use LA05 twice, placing $\bar{\boldsymbol{b}}_1$ in column l and then $\bar{\boldsymbol{b}}_2$ in column k. Rather than incorporating $\boldsymbol{P}$ in the factorization of $\bar{\boldsymbol{B}}$, the permutation may be stored separately. All of the other updates required by Theorems 2.2 and 2.3 may be accomplished by either a single (nonsingular) rank-1 update of the form (3.9) or a (nonsingular) rank-2 update of the form (3.10) in which at least one of the intermediate matrices (3.11) is nonsingular.

4. Initial point problems

We now turn to the problem of finding an initial FSEP mentioned at the end of Section 2. Let $\boldsymbol{x}_0$ be any point satisfying (3.6c) and let $\boldsymbol{D}$ be the diagonal matrix whose ith diagonal entry is $b_i - \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}_0$. The standard phase-1 problem (see, for example, Chvátal, 1983) is then given by

$$\text{minimize} \quad \boldsymbol{e}^{\mathrm{T}}\boldsymbol{y} \tag{4.1a}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{y} = \boldsymbol{b}, \tag{4.1b}$$

$$\boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \tag{4.1c}$$

$$\text{and} \quad \boldsymbol{0} \leq \boldsymbol{y} \leq \boldsymbol{e}, \tag{4.1d}$$

where $\boldsymbol{e}$ is a vector of ones. The point $\boldsymbol{x} = \boldsymbol{x}_0$, $\boldsymbol{y} = \boldsymbol{e}$ is feasible for (4.1); furthermore $\boldsymbol{x} = \bar{\boldsymbol{x}}$ satisfies (3.6b,c) if and only if $\boldsymbol{x} = \bar{\boldsymbol{x}}$, $\boldsymbol{y} = \boldsymbol{0}$ solves (4.1). It is straightforward to pick a point $\boldsymbol{x}_0$ so that $\boldsymbol{x} = \boldsymbol{x}_0$, $\boldsymbol{y} = \boldsymbol{e}$ is a vertex of (4.1b,c,d) (introducing artificial bounds if necessary). In this case, any optimal solution to (4.1) for which $\boldsymbol{y} = \boldsymbol{0}$ must yield an FSEP for (3.6) provided that we now consider (3.6) to be a problem in the $n+m$ variables $\boldsymbol{x}$ and $\boldsymbol{y}$ and include the additional constraints $0 \leq y_i \leq 0$. (These latter constraints would be removed from the problem if they were not, or if they ceased to be, in $\boldsymbol{l}$.) Possible disadvantages of such an approach are that (4.1) is a linear programming problem (requiring different factorizations from (3.6)) and that the initial feasible point will normally lie at a vertex of (3.6b,c) (whereas the solution to (3.6), especially when $\boldsymbol{H}$ is positive definite, may not and the phase-2 calculation may require many iterations to recover from a poor initial feasible point).

An alternative is to let ρ be a nonnegative scalar parameter and to solve the composite problem

$$\text{minimize} \quad Q(\boldsymbol{x}) + \rho \boldsymbol{e}^{\mathrm{T}}\boldsymbol{y}, \tag{4.2}$$

subject to the constraints (4.1b,c,d). (Such a problem is related to the Big-M method for linear programming, see, for example, Chvátal, 1983.) It is easy to show that, if the Lagrange multipliers for the general constraints (3.6b) at a local minimizer $\bar{\boldsymbol{x}}$ of (3.6) are $\boldsymbol{\lambda}_{\mathrm{GC}}$ and if $\rho \geq -\min(\boldsymbol{D}\boldsymbol{\lambda}_{\mathrm{GC}})_i$ then $\boldsymbol{x} = \bar{\boldsymbol{x}}$ and $\boldsymbol{y} = \boldsymbol{0}$ is a local minimizer of (4.2). This means that, in particular, if $\boldsymbol{H}$ is positive definite and if ρ is large enough, simple convexity arguments show that solving (4.2) is equivalent to solving (3.6). Consequently, such problems can be solved in a single

phase. The difficulty in making an a priori selection of ρ may be avoided if (4.2) is considered as a *parametric* problem (in which ρ is increased as a parameter) and if the quadratic programming method is capable of solving such problems (see Section 5).

When $\boldsymbol{H}$ is not positive definite, it is conceivable that local solutions to (4.2) for which $\boldsymbol{e}^T\boldsymbol{y} \neq \boldsymbol{0}$ might be encountered for all finite ρ and yet a feasible solution to (3.6) still exist. In fact this turns out to be impossible in a number of important cases. We have

THEOREM 4.1 *Suppose that the feasible region* (3.6b,c) *is nonempty. Suppose further that either*

(i) *the feasible region* (3.6b,c) *is bounded, or*

(ii) *for each value of ρ, only bounded local solutions to* (4.2) *are encountered in the solution process and that the matrix $\boldsymbol{K}(\boldsymbol{l})$ corresponding to a given solution satisfies* SOC.

Then, any solution $(\boldsymbol{x}, \boldsymbol{y})$ to (4.2) *yields a solution $\boldsymbol{x}$ to* (3.6) *for all sufficiently large ρ.*

Proof. To see this, let L and U be any two disjoint subsets of the first n integers, let Z be any subset of the first m integers and let

$$A = \{(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{x}_i = \boldsymbol{l}_i,\ i \in L,\ \boldsymbol{x}_i = \boldsymbol{u}_i,\ i \in U \text{ and } \boldsymbol{y}_i = 0,\ i \in Z\}. \tag{4.3}$$

Now let EP(A, ρ) denote the problem

$$\text{minimize} \quad Q(\boldsymbol{x}) + \rho \boldsymbol{e}^T\boldsymbol{y} \tag{4.4a}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{y} = \boldsymbol{b} \tag{4.4b}$$

$$\text{and} \quad (\boldsymbol{x}, \boldsymbol{y}) \in A. \tag{4.4c}$$

For such a problem $\boldsymbol{l}$ is made up of the constraints (4.4b) and (4.4c). It is easy to show that, for a given set A, EP(A, ρ) can only provide a local solution to (4.2) for all ρ in at most one interval $[\underline{\rho}_A, \bar{\rho}_A]$. (The intervals may be closed or open at either end and may be infinite.) Let $S(\rho)$ be the set of sets A which give rise to local solutions to (4.2) for a particular value of ρ. The region (4.1b,c,d) is nonempty. Therefore, (4.2) must have at least one solution for each value of ρ ($S(\rho)$ is nonempty) and any local solution is, by assumption (a) or (b), necessarily bounded. (In case (a), the boundedness and nonemptiness of (3.6b,c) is sufficient to ensure that (4.1b,c,d) is bounded. For suppose otherwise that $\boldsymbol{x} = \bar{\boldsymbol{x}}$ satisfies (3.6b,c) but that (4.1b,c,d) is unbounded. Then there must be a nonzero vector $(\boldsymbol{p}, \boldsymbol{q})$ (a direction of recession in the terminology of convex analysis) such that

$$\boldsymbol{A}\boldsymbol{p} + \boldsymbol{D}\boldsymbol{q} = \boldsymbol{0}, \quad \boldsymbol{l} \leqslant \bar{\boldsymbol{x}} + \alpha\boldsymbol{p} \leqslant \boldsymbol{u} \text{ and } 0 \leqslant \alpha\boldsymbol{q} \leqslant \boldsymbol{e},$$

for all $\alpha \geqslant 0$. This implies that $\mathbf{q} = \mathbf{0}$ and hence that

$$\boldsymbol{A}(\bar{\boldsymbol{x}} + \alpha\boldsymbol{p}) = \boldsymbol{b} \quad \text{and} \quad \boldsymbol{l} \leqslant \bar{\boldsymbol{x}} + \alpha\boldsymbol{p} \leqslant \boldsymbol{u},$$

for all $\alpha \geqslant 0$ which is impossible if (3.1b,c) is bounded. Thus (4.1b,c,d) is bounded.)

The only way in which we can obtain a local solution that does not yield a feasible solution to (3.6) for *any* ρ is for the algorithm to encounter a set $A \in S$ for which $\bar{\rho}_A$ is infinite and for which $e^T y > 0$ for all $\rho \in [\rho_A, \infty)$. Consider a particular value of ρ, $\hat{\rho}$, in this interval and its related solution. The solution, (x, y) and its associated Lagrange multipliers, λ, will satisfy an equation of the form

$$\begin{bmatrix} A_{FX}^T & H_{OD} & 0 & I & 0 \\ D_{FX}^T & 0 & 0 & 0 & I \\ A_{FR}^T & H_{FR} & 0 & 0 & 0 \\ D_{FR}^T & 0 & 0 & 0 & 0 \\ 0 & A_{FR} & D_{FR} & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_{GC} \\ x_{FR} \\ y_{FR} \\ \lambda_{BX} \\ \lambda_{BY} \end{bmatrix} = \begin{bmatrix} -c_{FX} - H_{FX} x_{FX} \\ -\hat{\rho} e \\ -c_{FR} - H_{OD}^T x_{FX} \\ -\hat{\rho} e \\ b - A_{FX} x_{FX} - D_{FX} y_{FX} \end{bmatrix}, \tag{4.5}$$

with x_{FX} and y_{FX} fixed at the bounds defined in (4.4c). The solution for $\rho \geq \hat{\rho}$ and its associated multipliers are then given by

$$(x + (\rho - \hat{\rho})\Delta x, y + (\rho - \hat{\rho})\Delta y) \quad \text{and} \quad \lambda + (\rho - \hat{\rho})\Delta\lambda, \tag{4.6}$$

where $\Delta x_{FX} = 0$, $\Delta y_{FX} = 0$ and

$$\begin{bmatrix} A_{FX}^T & H_{OD} & 0 & I & 0 \\ D_{FX}^T & 0 & 0 & 0 & I \\ A_{FR}^T & H_{FR} & 0 & 0 & 0 \\ D_{FR}^T & 0 & 0 & 0 & 0 \\ 0 & A_{FR} & D_{FR} & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\lambda_{GC} \\ \Delta x_{FR} \\ \Delta y_{FR} \\ \Delta\lambda_{BX} \\ \Delta\lambda_{BY} \end{bmatrix} = \begin{bmatrix} 0 \\ -e \\ 0 \\ -e \\ 0 \end{bmatrix}, \tag{4.7}$$

As (4.6) is a local solution to (4.2) for all $\rho \geq \hat{\rho}$, the signs of the components of $\Delta\lambda_{BX}$ and $\Delta\lambda_{BY}$ are such that the multipliers corresponding to the constraints (4.4) do not change sign as ρ increases. Moreover $\Delta x_{FR} = 0$ and $\Delta y_{FR} = 0$. (This occurs in case (a) because (4.1b,c,d) is bounded. In case (b), (4.7) yields

$$[\Delta x_{FR}^T \; \Delta y_{FR}^T] \begin{bmatrix} H_{FR} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{FR} \\ \Delta y_{FR} \end{bmatrix} = -e^T \Delta y_{FR},$$

$$[A_{FR} D_{FR}] \begin{bmatrix} \Delta x_{FR} \\ \Delta y_{FR} \end{bmatrix} = 0.$$

It then follows from SOC that $e^T \Delta y_{FR} \leq 0$ and that $\Delta x_{FR} = 0$ if and only if $e^T \Delta y_{FR} = 0$. As the variables y_{FR} are bounded from below, Δy_{FR}, and hence $e^T \Delta y_{FR}$ and Δx_{FR}, must be zero.) Hence

$$\begin{bmatrix} A_{FX}^T & I & 0 \\ D_{FX}^T & 0 & I \\ A_{FR}^T & 0 & 0 \\ D_{FR}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\lambda_{GC} \\ \Delta\lambda_{BX} \\ \Delta\lambda_{BY} \end{bmatrix} = \begin{bmatrix} 0 \\ -e \\ 0 \\ -e \end{bmatrix}. \tag{4.8}$$

But then the solution (x, y) and the parameters $\Delta\lambda_{BX}$ and $\Delta\lambda_{BY}$ are optimal primal and dual variables for the phase-1 linear program (4.1); as $e^T y > 0$, (3.6)

has no feasible solution, which contradicts our initial hypothesis and thus establishes the theorem. □

We may also deduce the following corollaries to Theorem 4.1.

COROLLARY 4.2 *Suppose that the feasible region* (3.6b,c) *is nonempty and that the bounds* $\boldsymbol{l}$ *and* $\boldsymbol{u}$ *are finite. Then, any solution* $(\boldsymbol{x}, \boldsymbol{y})$ *to* (4.2) *yields a solution* $\boldsymbol{x}$ *to* (3.6) *for all sufficiently large* ρ.

COROLLARY 4.3 *Suppose that the assumptions of Theorem* 4.1 *are satisfied and further that any solution to* (3.6) *satisfies* SOC. *Then the solutions to* (3.6) *and* (4.2) *coincide for all sufficiently large* ρ.

There are a number of ways of proceeding if (a) and (b) of Theorem 4.1 are not satisfied and we encounter an unbounded solution to (4.2). The simplest is merely to revert to solving (4.1) (This is equivalent to solving (4.2) with infinite ρ.) We might also introduce artificial bounds whenever a step to infinity is indicated. It is worth noting, however, that if a step to infinity occurs with $\mu > 0$ (Theorem 2.3) for (4.2), then a move in the same direction from any feasible point for (3.6) will also result in a step to infinity. This follows as the components of $\boldsymbol{p}_{\mathrm{FR}}$ for the artificial variables $\boldsymbol{y}$ must all be zero by virtue of (4.1d). The remaining components of $\boldsymbol{p}$ then give a feasible direction of negative curvature for $Q(\boldsymbol{x})$ which is not restricted by any of the bounds (3.6c/4.1c). Therefore the presence of such a direction for (4.2) is indicative of an unbounded solution to (3.6) whenever (3.6) has a feasible point. The situation when $\mu = 0$ is not so clear.

We note that it is obviously unnecessary to include an artificial variable for any constraint which is satisfied at $\boldsymbol{x}_0$. Indeed, it is desirable for there to be as few artificial variables as possible as the algorithm will normally perform at least as many iterations as there are artificial variables. This then raises the interesting question of finding an initial $\boldsymbol{x}_0$ at which many constraints are satisfied, a process often known as crashing. We shall not comment further on this problem here but refer the reader to Gould and Reid (1989) for one possible solution.

5. Parametric problems

An important feature of the Harwell code VE09 is that it provides a facility for solving parametric problems. Indeed this option is used extensively in the solution of ordinary problems whenever the single-phase method (4.2) described in Section 4 is attempted—VE09 used just such a single-phase method. The theory of parametric quadratic programming is well understood (at least for convex problems) and has been described by many authors (see, for instance, Ritter, 1967; Best, 1982; Väliaho, 1985). Our implementation is based upon the framework given by Best (1982) with adaptations to cope with nonconvex problems. Briefly, we consider the parametric problem

$$\text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} + (\boldsymbol{c} + t\,\delta\boldsymbol{c})^{\mathrm{T}}\boldsymbol{x} \tag{5.1a}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} + t\,\delta\boldsymbol{b} \tag{5.1b}$$

$$\text{and} \quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u} \tag{5.1c}$$

for all values of t in the interval $[\underline{t}, \bar{t}]$. Although other parametric problems sometimes occur (see, for example, Pang, Kaneko, & Hallman, 1979), problems of the form (5.1) are by far the most common. Classes of such problems occur in, for example, the fields of portfolio analysis (see Markowitz, 1952), structural engineering (see Maier, 1970), and multiple criterion decision processes (see Rhode & Weber, 1984).

Suppose we have found a local solution $\boldsymbol{x}$ for the value $t = \hat{t} \in [\underline{t}, \bar{t})$, that the components $\boldsymbol{x}_{\mathrm{FX}}$ of $\boldsymbol{x}$ are contained in $\boldsymbol{l}$ and fixed at one of the bounds (5.1c), and that $\boldsymbol{K}(\boldsymbol{l})$ satisfies SOC. The remaining variables $\boldsymbol{x}_{\mathrm{FR}}$ and the Lagrange multipliers then satisfy

$$\begin{bmatrix} \boldsymbol{A}_{\mathrm{FX}}^{\mathrm{T}} & \boldsymbol{H}_{\mathrm{OD}} & \boldsymbol{I} \\ \boldsymbol{A}_{\mathrm{FR}}^{\mathrm{T}} & \boldsymbol{H}_{\mathrm{FR}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{\mathrm{FR}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_{\mathrm{GC}} \\ \boldsymbol{x}_{\mathrm{FR}} \\ \boldsymbol{\lambda}_{\mathrm{BC}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{c}_{\mathrm{FX}} - \hat{t}\,\delta\boldsymbol{c}_{\mathrm{FX}} - \boldsymbol{H}_{\mathrm{FX}}\boldsymbol{x}_{\mathrm{FX}} \\ -\boldsymbol{c}_{\mathrm{FR}} - \hat{t}\,\delta\boldsymbol{c}_{\mathrm{FR}} - \boldsymbol{H}_{\mathrm{OD}}^{\mathrm{T}}\boldsymbol{x}_{\mathrm{FX}} \\ \boldsymbol{b} + \hat{t}\,\delta\boldsymbol{b} - \boldsymbol{A}_{\mathrm{FX}}\boldsymbol{x}_{\mathrm{FX}} \end{bmatrix}. \tag{5.2}$$

Let $\delta\boldsymbol{x}_{\mathrm{FX}} = \boldsymbol{0}$ and

$$\begin{bmatrix} \boldsymbol{A}_{\mathrm{FX}}^{\mathrm{T}} & \boldsymbol{H}_{\mathrm{OD}} & \boldsymbol{I} \\ \boldsymbol{A}_{\mathrm{FR}}^{\mathrm{T}} & \boldsymbol{H}_{\mathrm{FR}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{\mathrm{FR}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\lambda}_{\mathrm{GC}} \\ \delta\boldsymbol{x}_{\mathrm{FR}} \\ \delta\boldsymbol{\lambda}_{\mathrm{BC}} \end{bmatrix} = \begin{bmatrix} -\delta\boldsymbol{c}_{\mathrm{FX}} \\ -\delta\boldsymbol{c}_{\mathrm{FR}} \\ \delta\boldsymbol{b} \end{bmatrix}. \tag{5.3}$$

Then $\boldsymbol{x} + (t - \hat{t})\,\delta\boldsymbol{x}$ gives a local solution to (5.1) so long as

$$\boldsymbol{l}_{\mathrm{FR}} \leq \boldsymbol{x}_{\mathrm{FR}} + (t - \hat{t})\,\delta\boldsymbol{x}_{\mathrm{FR}} \leq \boldsymbol{u}_{\mathrm{FR}}, \tag{5.4a}$$

$$(\boldsymbol{\lambda}_{\mathrm{BC}})_i + (t - \hat{t})(\delta\boldsymbol{\lambda}_{\mathrm{BC}})_i < 0 \quad \text{if} \quad (\boldsymbol{x}_{\mathrm{FR}})_i < (\boldsymbol{u}_{\mathrm{FR}})_i, \tag{5.4b}$$

$$(\boldsymbol{\lambda}_{\mathrm{BC}})_i + (t - \hat{t})(\delta\boldsymbol{\lambda}_{\mathrm{BC}})_i > 0 \quad \text{if} \quad (\boldsymbol{x}_{\mathrm{FR}})_i > (\boldsymbol{l}_{\mathrm{FR}})_i. \tag{5.4c}$$

Let $\acute{t}$ be the supremum of all values of t for which (5.4) is satisfied. To simplify the discussion, suppose that only one of the inequalities (5.4) is critical—that is, would be violated if t were further increased—at this supremum. (This amounts to a nondegeneracy assumption; Ritter (1981) shows how this assumption may be relaxed.) We may then extend the parametric solution from $\hat{t}$ to $\acute{t}$. The solution at and beyond $\acute{t}$ depends upon which of the inequalities (5.4) becomes critical at $\acute{t}$.

If one of (5.4a) is violated beyond $\acute{t}$, the relevant variable is introduced into $\boldsymbol{l}$. Under these circumstances the analysis given by Best remains true even for nonconvex problems. Two possibilities occur; either the new $\boldsymbol{K}(\boldsymbol{l})$ is nonsingular (in which case it satisfies SOC and t may be extended beyond $\acute{t}$ using the new (5.3)) or it is singular. In the latter case, either we can conclude that the solution path we are following ends at $\acute{t}$ or there is another variable whose removal gives rise to a set $\boldsymbol{l}$ for which $\boldsymbol{K}(\boldsymbol{l})$ again satisfies SOC and which also allows us to extend t beyond $\acute{t}$.

If one of (5.4b) or (5.4c) is violated at $\acute{t}$, possible outcomes may be deduced from Theorem 2.4 when applied to (5.1). If case (i) of the theorem applies, the solution may be extended beyond $\acute{t}$ by freeing the relevant variable from its bound and using the new (5.3). In case (iii), $\boldsymbol{x} + (t - \hat{t})\,\delta\boldsymbol{x}$ does not solve (5.1) at $t = \acute{t}$ and a new solution to the problem must be sought. This gives a discontinuity in the parametric solution but cannot occur when $\boldsymbol{H}$ is positive (semi-)definite. In

case (ii) the position is not so clear. Although $x + (t - \hat{t})\,\delta x$ is an FSEP for (5.1) at $t = \hat{t}$ with nonpositive Lagrange multipliers, the presence of a zero multiplier makes it difficult to say whether $x + (\hat{t} - \hat{t})\,\delta x$ solves (5.1) or not. (This is the same problem as that discussed in Section 2.) Our remedy is to increase t very slightly beyond $\hat{t}$ to $\grave{t}$ and to resolve the problem at $t = \grave{t}$ using $x + (\grave{t} - \hat{t})\,\delta x$ as the starting point. This will mean that we will miss a small portion of the parametric interval but can simply backtrack from $\grave{t}$ to recover the missing part. Under our simplifying nondegeneracy assumption, we can always choose $\grave{t} > \hat{t}$ so that the starting point is an FSEP for (5.1); the existing factorization of $\boldsymbol{B}$ may be used when the new minimization is attempted.

We note that special forms of equations (5.2)–(5.4) are the basis of the method proposed by Markowitz & Perold (1981) for the solution of portfolio optimization problems; such problems can be stated and solved as parametric quadratic programs.

6. Numerical experience

In this section we give details of some of the problems that we have solved using VE09. The list is intended to provide a bench-mark for the development of future codes. We have tried to find problems which are either nonconvex or large. We have found the former to be extremely useful in debugging our code. The latter are included as an indication as to how the algorithm performs on larger problems; we purposefully did not solve any huge problems because of the expense involved but would be grateful to receive any real-life problems (i.e. problems for which the answer is of interest to someone) for future testing.

Our code is written in Double Precision Fortran 77. All of our computation was performed on the IBM 3084Q computer at Harwell; the code was compiled using the VS Fortran compiler with the optimization option OPT = 2. All timings reported are in seconds for time spent in the CPU and appear to be correct to about one hundredth of a second.

Our purpose in reporting some numerical results is merely to indicate how an algorithm like VE09 should perform on reasonably large examples. We do not regard the results as indicating the best performance that may be achieved by such algorithms. Indeed, as the number of iterations required will depend crucially upon the initial point and the associated set $\boldsymbol{I}$, the performance will undoubtedly be enhanced by a more sophisticated crash procedure for automatically generating the starting point. This issue has recently been investigated by Gould & Reid (1989).

Problem 1. For our first problem we took problem 118 of Hock & Schittkowski (1981), converted the inequality constraints to equalities by adding slack variables, and perturbed some of the diagonal entries for the objective function to make the problem nonconvex. If we denote the diagonal Hessian matrix for the problem by $\boldsymbol{D}$, our problem differs from that given by Hock & Schittkowski in the following diagonal elements: $d_{1,1} = -1.0$, $d_{4,4} = -0.0001$, $d_{6,6} = 10.0$, $d_{7,7} = -0.0001$, $d_{9,9} = 25.0$, $d_{10,10} = -2.5$, and $d_{13,13} = -0.0001$. The performance of the algorithm on this problem is summarized in Table 6.1.

TABLE 6.1
Results for test problem 1.

n	Iterations	$Q(x^*)$	Time (s)
32	22	−3.485 333E + 03	0.10

The solution x^* obtained for this problem is defined by the variables x_i $(i \in L)$ being fixed at their lower bounds and x_i $(i \in U)$ at their upper bounds (the remaining variables lying between their bounds); the sets L and U are as follows:

$$L = \{2, 3, 6, 9, 20, 30, 31, 32\}, \qquad U = \{1, 16, 17, 18, 22, 23, 27\}.$$

In addition we solve a parametric version (5.1) of the same problem for all t in the interval $[0, 5]$ and where the elements of the vectors $\delta \boldsymbol{b}$ and $\delta \boldsymbol{c}$ are all zero excepting $\delta c_1 = 1$, $\delta c_5 = -1$, $\delta c_{16} = 1$, $\delta c_{24} = -2$, $\delta b_1 = 6$, and $\delta b_9 = -6$. The solution obtained was defined over 10 parametric sub-intervals and for reference the intervals and the changes which occur at the end of each interval are described in Table 6.2. The problem has no feasible solution for $t > 4.5$.

Problem class 2. Here we solve

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{k-2} (x_{k+i+1} - x_{k+i})^2$$

$$\text{subject to} \quad x_{k+i} - x_{i+1} + x_i = 0, \quad i = 1, \ldots, k-1,$$

$$\alpha_i \leq x_i \leq \alpha_{i+1}, \quad i = 1, \ldots, k,$$

$$0.4(\alpha_{i+2} - \alpha_i) \leq x_{k+i} \leq 0.6(\alpha_{i+2} - \alpha_i), \quad i = 1, \ldots, k-1,$$

where the constants α_i, $i = 1, \ldots, k+1$, are given. These problems arise in the optimal placement of nodes in a scheme for solving ordinary differential equations with given boundary values (J. R. Kightley, private communication).

In Table 6.3 we give details of the performance of VE09 on this problem for various values of k and n $(= 2k - 1)$, and where we chose $\alpha_i = 1.0 + 1.01^{i-1}$,

TABLE 6.2
Details of the parametric solution

Interval end	Variable leaving I	Variable joining I
0.6667	6 (L)	24 (L)
1.0000	20 (L)	21 (U)
1.7080	21 (U)	
1.7100		20 (U)
1.8333	9 (L)	25 (L)
3.4186	2 (L)	
3.4206		21 (L)
4.1423	30 (L)	
4.1566		26 (L)
4.5000	4 (L)	

TABLE 6.3
Results for test problem class 2

k	n	Iterations	$Q(x^*)$	Time (s)
50	99	62	1.309070E − 07	0.63
100	199	122	9.395663E − 07	1.77
150	299	169	3.123420E − 06	3.16
200	399	222	9.002067E − 06	6.37
250	499	265	2.489882E − 05	7.01
300	599	308	6.789742E − 05	8.86
350	699	350	1.841940E − 04	10.70

$1 \leq i \leq k+1$. The algorithm was started from the feasible point at which the first k variables are at their lower bounds and the remaining variables are determined by the constraints—this is the point chosen by the crash procedure in VE09. For reference, the solutions for these problems are defined by the sets L and U as follows:

$$k = 50, \quad L = \{1, 50\}, \quad U = \{25, 26\};$$

$$k = 100, \quad L = \{1, 100\}, \quad U = \{24, 25, 75, 76\};$$

$$k = 150, \quad L = \{1, 150\}, \quad U = \{22, 126\};$$

$$k = 200, \quad L = \{1, 90, 200\}, \quad U = \{20, 21, 177\};$$

$$k = 250, \quad L = \{1, 107, 115, 123, 130, 250\}, \quad U = \{21, 226, 227\};$$

$$k = 300, \quad L = \{1, 107, 115, 123, 130, 136, 139, 147, 155, 162, 168, 179, 181, 300\},$$
$$U = \{21, 277\};$$

$$k = 350,$$
$$L = \{1, 107, 115, 123, 130, 136, 139, 147, 155, 162, 168, 171, 179, 187, 195, 200, 203,$$
$$211, 219, 228, 230, 232, 350\},$$
$$U = \{21, 326, 327\}.$$

Problem class 3. Here we solve a variant of the problem in class 2,

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{k-2}(x_{k+i+1} - x_{k+i})^2 + \frac{1}{2}\sum_{i=1}^{k-1}(x_{k-i} + x_{k+i} - \alpha_{k-i+1})^2$$

$$\text{subject to} \quad x_{k+i} - x_{i+1} + x_i = 0, \quad i = 1, \ldots, k-1,$$

$$\alpha_i \leq x_i \leq \alpha_{i+1}, \quad i = 1, \ldots, k,$$

$$0.4(\alpha_{i+2} - \alpha_i) \leq x_{k+i} \leq 0.6(\alpha_{i+2} - \alpha_i), \quad i = 1, \ldots, k-1,$$

where the constants α_i, $i = 1, \ldots, k-1$, are given.

In Table 6.4 we give details of the performance of VE09 on this problem for various values of k; we chose α_i and started from the same point as described for the previous class. For reference, the solutions for these problems are defined by

TABLE 6.4
Results for test problem class 3

k	n	Iterations	$Q(x^*)$	Time (s)
50	99	26	1.307696E + 02	0.28
100	199	72	3.755771E + 02	1.01
150	299	142	8.851590E + 02	2.44
200	399	208	2.036007E + 03	4.54
250	499	285	4.801736E + 03	7.05
300	599	370	1.174076E + 04	10.60
350	699	459	2.964436E + 04	14.65

the following sets L, there being no variables at their upper bounds:

$$k = 50, \quad L = \{1,..., 24, 50, 99\};$$

$$k = 100, \quad L = \{1,..., 49, 199\};$$

$$k = 150, \quad L = \{1,..., 74, 298, 299\};$$

$$k = 200, \quad L = \{1,..., 99, 398, 399\};$$

$$k = 250, \quad L = \{1,..., 124, 498, 499\};$$

$$k = 300, \quad L = \{1,..., 149, 598, 599\};$$

$$k = 350, \quad L = \{1,..., 174, 698, 699\}.$$

Problem class 4. Here we take any (large-scale) linear programming problem

$$\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x}$$

$$\text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \quad \text{and} \quad \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}$$

and replace the objective by

$$\text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{D}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x},$$

where $\boldsymbol{D}$ is a prespecified diagonal matrix. In Table 6.5 we summarize the performance of VE09 on such problems for various (well-known) linear programming test examples. The problems were supplied by John Reid of Harwell and further details are available from the author. The matrix $\boldsymbol{D}$ for the problems was chosen to have diagonal elements d_{ii} satisfying

$$d_{ii} = d_{11} + (i-1)/(n-1)(d_{nn} - d_{11}), \quad i = 1,..., n,$$

where the elements d_{11} and d_{nn} were either (A) 1.0 and 10.0 or (B) -1.0 and 10.0.

The initial point for each problem was chosen by the crash routine within VE09; the sets which define the solutions obtained are not given here but are available on request.

TABLE 6.5
Results for test problem class 4

Test problem	n	m	Iterations	$Q(x^*)$	Time (s)
Blend (A)	114	74	91	2.492275E + 03	1.55
Blend (B)	114	74	79	1.210728E + 03	1.41
Boeing 1 (A)	726	351	859	1.815191E + 09	56.00
Boeing 1 (B)	726	351	1001	1.248909E + 09	63.85
Boeing 2 (A)	304	165	222	8.918297E + 07	8.33
Boeing 2 (B)	304	165	178	5.701263E + 07	7.47
Stair (A)	532	356	443	3.568584E + 06	82.82
Stair (B)	532	356	465	3.246777E + 06	88.05

7. Conclusions

We have presented the basis for a general large-scale quadratic programming algorithm. The algorithm proceeds in a single phase and is equally capable of solving parametric problems. It has been implemented as subroutine VE09 in the Harwell Subroutine Library. In the future we intend to compare this method with the promising Schur complement method of Gill *et al.* (1987a).

Acknowledgement

The author would like to thank Iain Duff, John Reid and two referees for their useful comments on this paper. This work was carried out when the author was employed by the Computer Science and Systems Division, Harwell Laboratory, Oxfordshire, OX11 1RA, England.

REFERENCES

BARTELS, R. H., GOLUB, G. H., & SAUNDERS, M. A. 1970 Numerical techniques in mathematical programming. In: *Nonlinear Programming* (J. B. Rosen, O. L. Mangasarian, & K. Ritter, Eds). London and New York: Academic Press. Pp. 123–176.

BEST, M. J. 1982 An algorithm for the solution of the parametric quadratic programming problem. CORR 82–14, Dept. of Combinatorics and Optimization, University of Waterloo, Ontario, Canada.

BEST, M. J. 1984 Equivalence of some quadratic programming algorithms. *Mathematical Programming* **30,** 71–87.

BEST, M. J., & RITTER, K. 1976 An effective algorithm for quadratic minimization problems. MRC Technical report 1691, University of Wisconsin at Madison, Wisconsin, USA.

BLAND, R. G. 1977 New finite pivoting rules for the simplex method. *Mathematics of Operations Research* **2,** 103–107.

BUNCH, J. R., & KAUFMAN, L. C. 1980 A computational method for the indefinite quadratic programming problem. *Linear Algebra and its Applications* **34,** 341–370.

BUNCH, J. R., & PARLETT, B. N. 1971 Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Num. Anal.* **8,** 639–655.

CHANG, Y.-Y., & COTTLE, R. W. 1980 Least-index resolution of degeneracy in quadratic programming. *Mathematical Programming* **18,** 127–137.

CHVÁTAL, V. 1983 *Linear Programming.* New York and San Francisco: W. H. Freeman.

DANTZIG, G. B. 1963 *Linear Programming and Extensions.* Princeton: Princeton University Press.

DJANG, A. 1979 Algorithmic equivalence in quadratic programming. Ph.D. Thesis, Stanford University, California, USA.

DUFF, I. S., & REID, J. K. 1983 A multifrontal solution of indefinite sparse symmetric linear systems *ACM Transactions on Mathematical Software* **9,** 302–325.

DUFF, I. S., ERISMAN, A. M., & REID, J. K. 1986 *Direct Methods for Sparse Matrices.* Oxford: Oxford University Press.

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

FLETCHER, R. 1988 Degeneracy in the presence of round-off errors. *Linear Algebra and its Applications* **106,** 149–183.

GEORGE, A. 1974 On block elimination for sparse linear systems. *SIAM J. Num. Anal.* **11,** 585–603.

GILL, P. E., & MURRAY, W. 1978 Numerically stable methods for quadratic programming. *Mathematical Programming* **14,** 349–372.

GILL, P. E., MURRAY, W., & WRIGHT, M. H. 1981 *Practical Optimization.* London and New York: Academic Press.

GILL, P. E., MURRAY, W., SAUNDERS, M. A., & WRIGHT, M. H. 1985 Sparse matrix methods in optimization. *SIAM Journal on Scientific and Statistical Computing* **5,** 562–589.

GILL, P. E., MURRAY, W., SAUNDERS, M. A., & WRIGHT, M. H. 1987a A Schur-complement method for sparse quadratic programming. Technical report SOL 87-, Department of Operations Research, Stanford University, California, USA.

GILL, P. E., MURRAY, W., SAUNDERS, M. A., & WRIGHT, M. H. 1987b Maintaining LU factors of a general sparse matrix. *Linear Algebra and its Applications* **88/89,** 239–270.

GILL, P. E., MURRAY, W., SAUNDERS, M. A., & WRIGHT, M. H. 1989 A practical anti-cycling procedure for linear and nonlinear programming. *Mathematical Programming* **45,** 437–474.

GILL, P. E., GOULD, N. I. M., MURRAY, W., SAUNDERS, M. A., & WRIGHT, M. H. 1984 A weighted Gram–Schmidt method for convex quadratic programming. *Mathematical Programming* **30,** 176–196.

GOLDFARB, D., & IDNANI, A. 1983 A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming* **27,** 1–33.

GOLUB, G. H., & VAN LOAN, C. 1983 *Matrix Computations.* Baltimore: Johns Hopkins University Press.

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

GOULD, N. I. M., & REID, J. K. 1989 New crash procedures for large systems of linear constraints. *Mathematical Programming* **45,** 475–502.

HOCK, W., & SCHITTKOWSKI, K. 1981 *Test Examples for Nonlinear Programming Codes.* Lecture Notes in Economics and Mathematical Systems 187. Berlin: Springer-Verlag.

KELLER, E. L. 1973 The general quadratic programming problem. *Mathematical Programming* **5,** 311–337.

MAIER, G. 1970 A matrix structural theory of piecewise-elastoplasticity with interacting yield planes. *Meccanica* **5,** 54–66.

MANGASARIAN, O. L. 1980 Locally unique solutions of quadratic programs, linear and non-linear complementarity problems. *Mathematical Programming* **19,** 200–212.

MARKOWITZ, H. M. 1952 Portfolio selection. *The Journal of Finance* **12,** 77–91.

MARKOWITZ, H. M., & PEROLD, A. F. 1981 Sparsity and piecewise linearity in large portfolio optimization problems. In: *Sparse matrices and their uses* (I. S. Duff, Ed.). London and New York: Academic Press. Pp. 89–108.

MURRAY, W. 1971 An algorithm for finding a local minimum of an indefinite quadratic program. NAC 1, National Physical Laboratory, UK.

PANG, J. S., KANEKO, I., & HALLMAN, W. P. 1979 On the solution of some (parametric) linear complementarity problems with applications to portfolio selection, structural engineering and actuarial graduation. *Mathematical Programming* **16,** 325–347.

POWELL, M. J. D. 1981 An upper triangular matrix method for quadratic programming. In: *Nonlinear Programming 4* (O. L. Mangasarian, R. R. Meyer, & S. M. Robinson, Eds). London and New York: Academic Press. Pp. 1–24.

REID, J. K. 1982 A sparsity exploiting variant of the Bartels–Golub decomposition for linear programming bases. *Mathematical Programming* **24,** 55–69.

RHODE, R., & WEBER, R. 1984 The range of the efficient frontier in multiple objective linear programming. *Mathematical Programming* **28,** 84–95.

RITTER, K. 1967 A method for solving nonlinear maximum problems depending on parameters. *Naval Research Logistics Quarterly* **14,** 147–162.

RITTER, K. 1981 On parametric linear and quadratic programming. MRC Technical report 2197, University of Wisconsin at Madison, Wisconsin, USA.

SAUNDERS, M. A. 1976 A fast stable implementation of the simplex method using Bartels–Golub updating. In: *Sparse Matrix Computations* (J. R. Bunch & D. J. Rose, Eds). London and New York: Academic Press. Pp. 213–226.

SORENSEN, D. C. 1977 Updating the symmetric indefinite factorization with applications in a modified Newton method. ANL 77–49, Argonne National Laboratory, Illinois, USA.

VÄLIAHO, H. 1985 A unified approach to one-parametric general quadratic programming. *Mathematical Programming* **33,** 318–338.

VAN DE PANNE, C., & WHINSTON, A. 1969 The symmetric formulation of the simplex method for quadratic programming. *Econometrica* **37,** 507–527.

WOLFE, P. 1963 A technique for resolving degeneracy in linear programming. *SIAM Journal on Applied Mathematics* **11,** 205–211.

Appendix

In this appendix we show that the anti-cycling rules for linear programming proposed by Bland (1977) may be extended to cover the quadratic programming case. The rules are extremely simple to state:

(i) To remove a constraint from $\boldsymbol{l}$, pick the candidate with the smallest index.

(ii) When more than one constraint is encountered as the largest feasible step is taken, choose the candidate with the smallest index.

We model our proof of the finiteness of our algorithm under such rules on those given by Chang & Cottle (1980) and Chvátal (1983). The proof is by contradiction.

We have indicated in Section 2 that the algorithm can only fail to be finite if we encounter a cycle; that is we encounter a sequence of FSEPs corresponding to the sets $\boldsymbol{l}_j$ for which $\boldsymbol{l}_1 = \boldsymbol{l}_k$ for some $k > 1$. This can only happen in our algorithm if case (vi) of Theorem 2.3 repeatedly occurs and if $\bar{\alpha}$ is persistently 0. (Each matrix $\boldsymbol{K}(\boldsymbol{l}_j)$ will then satisfy SOC.) Notice that a cycle does not necessarily occur at a vertex of the feasible region. To obtain a contradiction we suppose that we apply rules (i) and (ii) when solving (1.1) and that a cycle is encountered.

Let $\boldsymbol{A}$ be the $m \times n$ matrix whose rows are the vectors $\boldsymbol{a}_i^{\mathrm{T}}$ and let $\boldsymbol{b}$ be the corresponding vector whose elements are b_i. Consider the underdetermined

system of linear equations

$$\begin{bmatrix} H & A^T & 0 \\ A & 0 & I \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ r \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}. \tag{A.1}$$

The objective function value $Q(x)$ for any point x obeying equation (A.1) then satisfies

$$2Q(x) = c^T x - b^T \lambda + r^T \lambda.$$

Now let I be a subset of $\{1,...,m\}$, let $\hat{A}_I$ and $\bar{A}_I$ be matrices whose rows are made up from the vectors a_i^T for $i \in I$ and $i \notin I$ respectively and let $\hat{b}_I$ and $\bar{b}_I$ be the corresponding vectors made up from elements b_i. Then, provided that $K(I)$ is nonsingular, a solution to (A.1) is given by

$$\begin{bmatrix} H & \hat{A}_I^T & 0 \\ \hat{A}_I & 0 & 0 \\ \bar{A}_I & 0 & I \end{bmatrix} \begin{bmatrix} x(\bar{\lambda}_I, \hat{r}_I, I) \\ \hat{\lambda}(\bar{\lambda}_I, \hat{r}_I, I) \\ \bar{r}(\bar{\lambda}_I, \hat{r}_I, I) \end{bmatrix} = \begin{bmatrix} -c \\ \hat{b} \\ \bar{b} \end{bmatrix} - \begin{bmatrix} \bar{A}_I^T & 0 \\ 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{\lambda}_I \\ \hat{r}_I \end{bmatrix} \tag{A.2}$$

for any choice of $\bar{\lambda}_I$ and $\hat{r}_I$. We may thus express the objective function $Q(x)$ in terms of the independent variables $\bar{\lambda}_I$ and $\hat{r}_I$. If we denote the value of $Q(x)$ by $Q(\bar{\lambda}_I, \hat{r}_I, I)$, Q satisfies

$$2Q(\bar{\lambda}_I, \hat{r}_I, I) = v_I + 2\hat{\lambda}(0, 0, I)^T \hat{r}_I + \hat{r}_I^T L_I \hat{r}_I + \bar{\lambda}_I^T N_I \bar{\lambda}_I, \tag{A.3}$$

where the block skew symmetric matrix

$$\begin{bmatrix} L_I & -M_I^T \\ M_I & N_I \end{bmatrix} = \begin{bmatrix} 0 & -I \\ \bar{A}_I & 0 \end{bmatrix} \begin{bmatrix} H & \hat{A}_I^T \\ \hat{A}_I & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & \bar{A}_I^T \\ I & 0 \end{bmatrix} \tag{A.4}$$

and where v_I is a constant. For convenience, define the vector λ_I to have a nonzero entry in the ith position $i \in I$ with the value of the appropriate entry from $\hat{\lambda}(0, 0, I)$.

The solution of each problem EP(I_j) is a special instance of (A.2) and the matrix $K(I_j)$ is invertible for these problems by virtue of SOC. The relationships (A.2) and (A.3) are quadratic programming equivalents of the dictionaries used by Chvátal in his proof of the finiteness of the simplex method under Bland's rules. They are also the basis for many of the linear programming-based methods for solving quadratic programs (see, for instance, Dantzig, 1963).

Following Chvátal, we say that a constraint is *fickle* if it is present in some set I_i but absent from I_{i+1} with $1 \leq i < k$. We shall say that a constraint is *faithful* if it is present in all the sets I_i. Any constraint which is neither faithful nor fickle is *disinterested*. We note that the residual r_i for any fickle or faithful constraint must be zero. Let constraint w be the fickle constraint with the largest subscript, let I_i be such that $w \in I_{i+1} \backslash I_i$, let $v \in I_i \backslash I_{i+1}$ and let I_j be such that $w \in I_j \backslash I_{j+1}$. We note that the least index rules imply

(a) $(\lambda_{I_i})_v > 0$ and $(\lambda_{I_i})_u \leq 0$ for all $u < v$, $u \in I_i$;

(b) constraint w is the only candidate for inclusion in $\mathcal{I}_{i+1}$ (because it is the fickle constraint with the largest index); and
(c) $(\lambda_{\mathcal{I}_i})_w > 0$ and $(\lambda_{\mathcal{I}_i})_u \leq 0$ for all $u < w$, $u \in \mathcal{I}_i$.
We shall need the following lemma.

LEMMA A.1 *Suppose $\mathcal{I}$ is any of the index sets encountered during the cycle. Then the matrix $N_{\mathcal{I}}$ defined by (A.4) is independent of $\mathcal{I}$. Furthermore, each row and column of $N_{\mathcal{I}}$ corresponding to a fickle constraint is null.*

Proof. We consider the consequences of changing from the index set $\mathcal{I}$ to the next set $\bar{\mathcal{I}}$ in the cycle. Suppose that the constraint with gradient $\hat{\boldsymbol{a}}$ moves out of $\mathcal{I}$ and is replaced with that with gradient $\bar{\boldsymbol{a}}$. Let

$$\begin{bmatrix} N_1 & N_2^T \\ N_2 & N_3 \end{bmatrix} = \begin{bmatrix} H & \hat{A}_{\mathcal{I}}^T \\ \hat{A}_{\mathcal{I}} & 0 \end{bmatrix}^{-1} \quad \text{and} \quad \begin{bmatrix} \bar{N}_1 & \bar{N}_2^T \\ \bar{N}_2 & \bar{N}_3 \end{bmatrix} = \begin{bmatrix} H & \hat{A}_{\bar{\mathcal{I}}}^T \\ \hat{A}_{\bar{\mathcal{I}}} & 0 \end{bmatrix}^{-1}. \tag{A.5}$$

It follows from (A.4) that $N_{\mathcal{I}} = \bar{A}_{\mathcal{I}} N_1 \bar{A}_{\mathcal{I}}^T$. By definition $N_1 \hat{\boldsymbol{a}} = \boldsymbol{0}$. Furthermore, as the cycle can only occur if equation (2.8) is satisfied with $\boldsymbol{u} = 0$, $N_1 \bar{\boldsymbol{a}} = \boldsymbol{u} = 0$. Referring to equation (15b) in Fletcher (1971) (with the suitable change in notation), we have $\bar{N}_1 = N_1$. But then, as $N_{\bar{\mathcal{I}}}$ only differs from $N_{\mathcal{I}}$ by a number of rank-1 terms, each of which contains one of the products $N_1 \bar{\boldsymbol{a}}$ or $N_1 \hat{\boldsymbol{a}}$, we have $N_{\bar{\mathcal{I}}} = N_{\mathcal{I}}$. Thus $N_{\mathcal{I}}$ is independent of $\mathcal{I}$. Finally, the column of $N_{\mathcal{I}}$ corresponding to the fickle constraint with gradient $\bar{\boldsymbol{a}}$ is given by $\bar{A}_{\mathcal{I}} N_1 \bar{\boldsymbol{a}}$ which is zero as $N_1 \bar{\boldsymbol{a}} = \boldsymbol{0}$. Since $N_{\mathcal{I}}$ is symmetric, the lemma is proved. □

Although the form of (A.2) is different for $\mathcal{I}_i$ and $\mathcal{I}_j$, the solution sets must be identical as both sets of equations are derived algebraically from (A.1). The objective function values (A.3) are therefore identical for $\mathcal{I}_i$ and $\mathcal{I}_j$ so long as the same choices of $\boldsymbol{\lambda}$ and $\boldsymbol{r}$ are made.

Let $\boldsymbol{p}$, $\boldsymbol{q}$, and $\boldsymbol{s}$ satisfy the equation

$$\begin{bmatrix} H & \hat{A}_{\mathcal{I}_i}^T & 0 \\ \hat{A}_{\mathcal{I}_i} & 0 & 0 \\ \bar{A}_{\mathcal{I}_i} & 0 & I \end{bmatrix} \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{q} \\ \boldsymbol{s} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{e}_{v_i} \\ \boldsymbol{0} \end{bmatrix},$$

where $\boldsymbol{a}_v^T$ is the v_ith row of $\hat{A}_{\mathcal{I}_i}$. (The solution to this equation is exactly that used to compute the search direction in Theorem 2.3, c.f. equation (2.7).) Furthermore, let $\boldsymbol{x}(\alpha)$, $\boldsymbol{\lambda}(\alpha)$, and $\boldsymbol{r}(\alpha)$ be such that $\boldsymbol{x}(\alpha) = \boldsymbol{x}(0, 0, \mathcal{I}_i) - \alpha \boldsymbol{p}$, the nonzero entries of $\boldsymbol{\lambda}(\alpha)$ occur in positions $k \in \mathcal{I}_i$ with the values being the entries of $\hat{\boldsymbol{\lambda}}(0, 0, \mathcal{I}_i) - \alpha \boldsymbol{q}$ in the appropriate order, and the nonzero entries of $\boldsymbol{r}(\alpha)$ occur in position v where the entry has the value α and in positions $k \notin \mathcal{I}_i$ with the values of the entries of $\bar{\boldsymbol{r}}(0, 0, \mathcal{I}_i) - \alpha \boldsymbol{s}$ in the appropriate order. Such choices of $\boldsymbol{x}$, $\boldsymbol{\lambda}$, and $\boldsymbol{r}$ satisfy (A.1) for all α as they satisfy (A.2) when $\mathcal{I} = \mathcal{I}_i$. Therefore the objective function values

$$v_{\mathcal{I}_i} + 2\sum_{l \in \mathcal{I}_i} (\lambda_{\mathcal{I}_i})_l r(\alpha)_l + \hat{\boldsymbol{r}}(\alpha)_{\mathcal{I}_i}^T L_{\mathcal{I}_i} \hat{\boldsymbol{r}}(\alpha)_{\mathcal{I}_i} + \bar{\boldsymbol{\lambda}}(\alpha)_{\mathcal{I}_i}^T N_{\mathcal{I}_i} \bar{\boldsymbol{\lambda}}(\alpha)_{\mathcal{I}_i} \tag{A.6a}$$

and

$$v_{\mathbf{I}_j} + 2\sum_{l=1}^{m} (\lambda_{\mathbf{I}_j})_l r(\alpha)_l + \hat{r}(\alpha)_{\mathbf{I}_j}^{\mathrm{T}} \boldsymbol{L}_{\mathbf{I}_j} \hat{r}(\alpha)_{\mathbf{I}_j} + \bar{\lambda}(\alpha)_{\mathbf{I}_j}^{\mathrm{T}} \boldsymbol{N}_{\mathbf{I}_j} \bar{\lambda}(\alpha)_{\mathbf{I}_j} \qquad \text{(A.6b)}$$

are identical for all α. We may thus equate the coefficients of similar powers of α in the expressions (A.6a) and (A.6b). In particular, we shall consider the coefficient of α in the two expressions.

Taking (A.6a) first, the only coefficient of α is the term

$$2(\lambda_{\mathbf{I}_i})_v, \qquad \text{(A.7)}$$

the only other nonzero terms in the expression are the constant $v_{\mathbf{I}_i}$ and a single coefficient of α^2, $(\boldsymbol{L}_{\mathbf{I}_i})_{v_i v_i}$. Now referring to (A.6b), the components of the vector $\hat{r}(\alpha)_{\mathbf{I}_j}$ are linear functions of α. However, the constant term in each component is zero; for the components correspond to either faithful or fickle constraints and the residuals for such constraints are necessarily zero. Thus the term $\hat{r}(\alpha)_{\mathbf{I}_j}^{\mathrm{T}} \boldsymbol{L}_{\mathbf{I}_j} \hat{r}(\alpha)_{\mathbf{I}_j}$ gives only α^2 (or possibly constant) terms. The components of $\bar{\lambda}(\alpha)_{\mathbf{I}_j}$ correspond to fickle or disinterested constraints. Lemma A.1 ensures that only the disinterested components make a contribution to $\bar{\lambda}(\alpha)_{\mathbf{I}_j}^{\mathrm{T}} \boldsymbol{N}_{\mathbf{I}_j} \bar{\lambda}(\alpha)_{\mathbf{I}_j}$; the constant terms for these components are zero by definition. Therefore, once again, the term $\bar{\lambda}(\alpha)_{\mathbf{I}_j}^{\mathrm{T}} \boldsymbol{N}_{\mathbf{I}_j} \bar{\lambda}(\alpha)_{\mathbf{I}_j}$ gives rise to only α^2 (or possibly constant) terms. The remaining terms in (A.6b) have the value

$$v_{\mathbf{I}_j} + 2\alpha(\lambda_{\mathbf{I}_j})_v + 2\sum_{l \notin \mathbf{I}_j} (\lambda_{\mathbf{I}_j})_l r(\alpha)_l. \qquad \text{(A.8)}$$

The coefficient of the α term in (A.6b) is therefore

$$2(\lambda_{\mathbf{I}_j})_v - 2\sum_{l \notin \mathbf{I}_j} (\lambda_{\mathbf{I}_j})_l (s)_{l_i}, \qquad \text{(A.9)}$$

where the index l_i is such that the l_ith row of $\bar{\boldsymbol{A}}_{\mathbf{I}_i}$ is $\boldsymbol{a}_l^{\mathrm{T}}$ for $l \notin \mathbf{I}_i$. Equating (A.7) and (A.9) and rearranging terms we have

$$(\lambda_{\mathbf{I}_i})_v - (\lambda_{\mathbf{I}_j})_v + \sum_{l \notin \mathbf{I}_j} (\lambda_{\mathbf{I}_j})_l s_{l_i} = \mathbf{0}. \qquad \text{(A.10)}$$

The remainder of the proof then follows Chvátal's. From observation (a) above, $(\lambda_{\mathbf{I}_i})_v > 0$. Furthermore $(\lambda_{\mathbf{I}_j})_v \leqslant 0$. (Constraint v is fickle and therefore $v < w$. If $v \in \mathbf{I}_j$, $(\lambda_{\mathbf{I}_j})_v \leqslant 0$ from observation (c). If $v \notin \mathbf{I}_j$, $(\lambda_{\mathbf{I}_j})_v = 0$, by definition.) Hence from (A.10)

$$(\lambda_{\mathbf{I}_j})_u s_{u_i} < 0 \quad \text{for some } u \notin \mathbf{I}_i. \qquad \text{(A.11)}$$

Since $u \notin \mathbf{I}_i$ and as $u \in \mathbf{I}_j$ (because $(\lambda_{\mathbf{I}_j})_u \neq 0$) constraint u is fickle. Thus $u < w$. In fact $u \neq w$ as w does not satisfy (A.11). For $(\lambda_{\mathbf{I}_j})_w > 0$ from observation (a), $s_{w_i} > 0$ as $w \in \mathbf{I}_{i+1} \backslash \mathbf{I}_i$ and hence $(\lambda_{\mathbf{I}_j})_w s_{w_i} > 0$. Thus as $u < w$ and $u \in \mathbf{I}_j$, $(\lambda_{\mathbf{I}_j})_u \leqslant 0$ from observation (c) and therefore $s_{u_i} > 0$ from (A.11). But this means that constraint u is a candidate for inclusion in $\mathbf{I}_{i+1}$ which contradicts observation (b) as $u \neq w$. Thus the algorithm cannot cycle and is therefore finite.