IMA Journal of Numerical Analysis (1986) **6**, 137–152

A Method for Solving Certain Quadratic Programming Problems Arising in Nonsmooth Optimization

KRZYSZTOF C. KIWIEL

Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland

[Received 27 July 1984 and in revised form 21 May 1985]

We present a finite algorithm for minimizing a piecewise linear convex function augmented with a simple quadratic term. To solve the dual problem, which is of least-squares form with an additional linear term, we include in a standard active-set quadratic programming algorithm a new column-exchange strategy for treating positive semidefinite problems. Numerical results are given for an implementation using the Cholesky factorization.

1. Introduction

CONSIDER the problem of finding an m-vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^{\mathrm{T}}$ to

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\,|P\boldsymbol{x}|^2 + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} = \tfrac{1}{2}\boldsymbol{x}^{\mathrm{T}}P^{\mathrm{T}}P\boldsymbol{x} + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x}, \\ &\text{subject to} \quad \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} = 1, \boldsymbol{x} \geqslant \boldsymbol{0}, \end{aligned} \tag{1.1}$$

where P is an $n \times m$ matrix, $\boldsymbol{a}$ is an m-vector, the vector $\boldsymbol{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$ has m components and the superscript T denotes transposition. The column rank of P may be less than m. Here and in the sequel, $\boldsymbol{x} \geqslant \boldsymbol{0}$ means $x_i \geqslant 0$ for $i = 1, \ldots, m$. Similarly $\boldsymbol{x} > \boldsymbol{0}$ means $x_i > 0$ for $i = 1, \ldots, m$.

Attention has recently been drawn to the need for efficient and robust methods for solving (1.1) by the development of descent methods for nonsmooth (nondifferentiable) optimization; see, for instance, Kiwiel (1983), Lemarechal (1977), Mifflin (1982). These methods find at each iteration a search direction $\boldsymbol{d}$ to

$$\text{minimize} \quad \hat{v}(\boldsymbol{d}) + \tfrac{1}{2}\,|\boldsymbol{d}|^2 \quad \text{over all } \boldsymbol{d} \in \mathbb{R}^n \tag{1.2a}$$

where

$$\hat{v}(\boldsymbol{d}) = \max\,\{-a_i + \boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{d} : i = 1, \ldots, m\} \tag{1.2b}$$

is a piecewise linear convex approximation to a nondifferentiable function f of n variables to be minimized, and $\boldsymbol{p}_i$ denotes the ith column of the matrix P. In this application the n-vectors $\boldsymbol{p}_i$ are subgradients of f, and the solution method employed must not require the $(n+1)$-vectors $\begin{bmatrix} 1 \\ \boldsymbol{p}_i \end{bmatrix}$ to be linearly independent. Problem (1.1) is dual to the following equivalent of (1.2) (see Wierzbicki (1982))

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\,|\boldsymbol{d}|^2 + v \quad \text{over all } (\boldsymbol{d}, v) \in \mathbb{R}^{n+1} \\ &\text{satisfying} \quad -a_i + \boldsymbol{p}_i^{\mathrm{T}}\boldsymbol{d} \leqslant v \quad \text{for } i = 1, \ldots, m, \end{aligned} \tag{1.3}$$

and their solutions are related by

$$d = -Px, \qquad v = -(|Px|^2 + a^{\mathrm{T}}x). \tag{1.4a,b}$$

Here and throughout the paper we write

$$(A, B, \ldots) = [A^{\mathrm{T}} \quad B^{\mathrm{T}} \quad \ldots]^{\mathrm{T}} = \begin{bmatrix} A \\ B \\ \vdots \end{bmatrix}$$

where A, B, etc. have common row-dimension, for example, when they are vectors or scalars.

Problem (1.1) is, of course, a problem of positive semi-definite quadratic programming, for which there exist several algorithms, e.g. Best & Ritter (1976), Fletcher (1971), Gill & Murray (1978), van de Panne & Whinston (1969), and Dantzig (1963) (see Pang (1981)). When applied to problem (1.1), these algorithms are *equivalent* to the general active-set method described by Best (1984). This means that under exact arithmetic they generate an identical sequence of points provided a common initial point is used, arbitrary decisions concerning ties are made using common rules and a simple assumption concerning the problem data at the initial point is satisfied. The various algorithms differ only in the manner in which they solve the linear equations expressing the Kuhn–Tucker system for the associated equality-constrained subproblems.

In this paper we give a highly specialised version of the active-set algorithm for solving problem (1.1). Under exact arithmetic, our algorithm is equivalent to those mentioned above. However, all these algorithms require the determination of potential singularity of certain matrices. In practice this determination may pose problems due to rounding errors. We present, therefore, a new column-exchange strategy which increases the stability of our method. The modified method is no longer equivalent to those mentioned above.

We may add that problem (1.1) may be solved by the method of Stoer (1971) (see also Schittkowski and Stoer (1979) and Schittkowski (1983)). For problem (1.1) Stoer's method is not equivalent to ours, solves its subproblems in a different way and does not use column exchanges to deal with (approximate) singularity of certain matrices. Also for the special case of problem (1.1) with $a = 0$ the algorithms of Mifflin (1979) and Wolfe (1976) are mathematically, but not computationally, equivalent to the simpler version of our method.

The equality-constrained subproblems of our method are solved by updating the Cholesky factorization of a sub-matrix of $P^{\mathrm{T}}P + ee^{\mathrm{T}}$ as in Wolfe (1976). We shall report elsewhere a (hopefully) more stable technique that recurs the QR factorization of a certain matrix. Here we note that the Cholesky decomposition requires only one third of the storage needed by the *QR* factorization, but may yield lower accuracy, since it solves least-squares problems via normal equations. Nevertheless, this accuracy seems to suffice for applications in nonsmooth optimization algorithms (see Kiwiel (1985)).

The algorithm is derived in Section 2. Section 3 discusses an implementation using the Cholesky factorization. Preliminary computational experience is reported in Section 4. Finally, we have a conclusion section.

2. The method

For $\boldsymbol{x}$ to solve problem (1.1) it is both necessary and sufficient (because of objective convexity and constraint linearity) that $\boldsymbol{x}$ and some number v satisfy the optimality conditions

$$\left.\begin{array}{l} \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x} = 1, \qquad \boldsymbol{x} \geqslant \boldsymbol{0}, \\ \boldsymbol{e}v + P^{\mathrm{T}}P\boldsymbol{x} + \boldsymbol{a} \geqslant \boldsymbol{0}, \\ v + \boldsymbol{p}_j^{\mathrm{T}}P\boldsymbol{x} + a_j = 0 \quad \text{or} \quad x_j = 0 \text{ for each } j = 1, \ldots, m. \end{array}\right\} \tag{2.1}$$

Note that v and $\boldsymbol{e}v + P^{\mathrm{T}}P\boldsymbol{x} + \boldsymbol{a}$ equal the Lagrange multipliers of (1.1).

The algorithm follows the active-set strategy (see Best (1984), Fletcher (1981), Gill, Murray & Wright (1981)) by solving a sequence of equality-constrained subproblems of the form

$$\begin{array}{ll} \text{minimize} & \tfrac{1}{2}|\hat{P}\boldsymbol{y}|^2 + \boldsymbol{a}^{\mathrm{T}}\boldsymbol{y}, \\ \text{subject to} & \boldsymbol{e}^{\mathrm{T}}\boldsymbol{y} = 1. \end{array} \tag{2.2}$$

Each subproblem is obtained by choosing an ordered set of some k different indices, $J = \{j_1, \ldots, j_k\}$, identifying $\boldsymbol{y}$ with "unconstrained" components of $\boldsymbol{x}$ via

$$\begin{array}{ll} x_{j_i} = y_i & \text{for } i = 1, \ldots, k, \\ x_j = 0 & \text{for } j \notin J, \end{array} \tag{2.3}$$

and setting $\hat{P} = [\boldsymbol{p}_{j_1}, \ldots, \boldsymbol{p}_{j_k}]$ and $\hat{\boldsymbol{a}} = [a_{j_1}, \ldots, a_{j_k}]^{\mathrm{T}}$. (Throughout, the number of components of vectors such as $\boldsymbol{e}$ and $\boldsymbol{y}$ is to be inferred from the context. In this case, it is k for both.) The algorithm revises J until the solution $\boldsymbol{y}$ of (2.2) produces a solution $\boldsymbol{x}$ of (1.1) via (2.3). Each J is such that the columns of $\hat{P}$ are affinely independent, i.e. the $(n+1) \times k$ matrix

$$\hat{P}_{\boldsymbol{e}} = \begin{bmatrix} \boldsymbol{e}^{\mathrm{T}} \\ \hat{P} \end{bmatrix} \quad \text{has full column rank.} \tag{2.4}$$

The importance of this property is explained in the following Lemma, whose proof is left to the reader.

LEMMA 2.1 *Condition* (2.4) *is equivalent to each of the following*:
(i) *The matrix* $\hat{P}_{\boldsymbol{e}}^{\mathrm{T}}\hat{P}_{\boldsymbol{e}} = \hat{P}^{\mathrm{T}}\hat{P} + \boldsymbol{e}\boldsymbol{e}^{\mathrm{T}}$ *is positive definite*;
(ii) *The following system has a unique solution* $(v, \boldsymbol{y})$

$$\boldsymbol{e}^{\mathrm{T}}\boldsymbol{y} = 1, \tag{2.5a}$$

$$\boldsymbol{e}v + \hat{P}^{\mathrm{T}}\hat{P}\boldsymbol{y} = -\hat{\boldsymbol{a}}. \tag{2.5b}$$

Moreover, $(v, \boldsymbol{y})$ *solves* (2.5) *if and only if* $\boldsymbol{y}$ *solves* (2.2) *and* v *is the associated Lagrange multiplier.*

The positive definiteness of $\hat{P}_{\boldsymbol{e}}^{\mathrm{T}}\hat{P}_{\boldsymbol{e}}$ will be used in Section 3 for solving (2.5).

A stopping criterion for the method is suggested by the following standard result (see, e.g. Fletcher (1981), Section 10.1).

LEMMA 2.2 *Suppose J is such that (v, y) solves* (2.5) *and*

$$y_i \geqslant 0 \quad \text{for each } i = 1, \ldots, k,$$
$$v + p_j^{\mathrm{T}}\hat{P}y + a_j \geqslant 0 \quad \textit{for each } j \in \{1, \ldots, m\}\backslash J$$

and define x by (2.3). *Then x and v satisfy* (2.1) *and, hence, x solves* (1.1).

We now briefly describe an iteration of the method. The current iterate x is identified with a k-vector $\hat{y}$ such that

$$e^{\mathrm{T}}\hat{y} = 1, \qquad \hat{y} \geqslant \mathbf{0}$$

and $x_{j_i} = \hat{y}_i$ for $i = 1, \ldots, k$, with $x_j = 0$ for $j \notin J$. If y, the solution of (2.2), is feasible ($y \geqslant \mathbf{0}$), then it becomes the new $\hat{y}$. If not then a step towards the nearest constraint is made from $\hat{y}$ in the direction of $s = y - \hat{y}$ to find the best feasible point. This can be expressed by choosing the step-size

$$t = \min\{1, \min\{\hat{y}_i/(\hat{y}_i - y_i) : y_i < 0,\ 1 \leqslant i \leqslant k\}\} \tag{2.6}$$

and taking the new $\hat{y}$ as $\hat{y} + ts$. (Note that the new $\hat{y}$ satisfies $e^{\mathrm{T}}y = 1$, since so do $\hat{y}$ and y). If the new $\hat{y}$ does not satisfy $\hat{y} > \mathbf{0}$ then a new constraint $x_{j_c} = 0$ becomes active by deleting j_c from J, where c is an index such that $\hat{y}_c = 0$. This reduction of J is called *column deletion*, cf. (2.2). Of course, there can be only finitely many successive column deletions, since $y = (y_1) = (1) > 0$ when J is a singleton. Hence, eventually the method computes $t = 1$ and $\hat{y} = y > \mathbf{0}$.

When $\hat{y} = y > \mathbf{0}$ and x does not solve (1.1), i.e.

$$v + p_l^{\mathrm{T}}\hat{P}y + a_l < 0 \tag{2.7}$$

for some $l \notin J$ (the Lagrange mulitplier for the constraint $x_l \geqslant 0$ is negative), it is possible to reduce the objective value by deactivating the constraint $x_l = 0$. This is done by including l in J.

Special care must be taken to ensure that the linear independence condition (2.4) holds for the new J. We start by analysing the case of *column augmentation* when the new $(n+1) \times (k+1)$ matrix

$$\hat{P}_e^+ = \begin{bmatrix} e^{\mathrm{T}} & 1 \\ \hat{P} & p_l \end{bmatrix} \tag{2.8}$$

has full column rank. Lemmas 2.3 and 2.4 given below show how to move from y to another feasible point z with a lower objective value. They are well-known results specialized to our case (see, e.g. Fletcher (1981), Section 9.1; Best (1984)). Their proofs are omitted, since they are similar to ones in Mifflin (1979).

LEMMA 2.3 *Suppose that (v, y) solves* (2.5) *and* (2.7) *holds for some $l \notin J$ such that the matrix $\hat{P}_e^+$ given by* (2.8) *has full column rank. Then the augmented linear system*

$$\begin{aligned} &e^{\mathrm{T}}y^+ + y_l^+ = 1, \\ &ev^+ + \hat{P}^{\mathrm{T}}(\hat{P}y^+ + p_l y_l^+) = -\hat{a}, \\ &v^+ + p_l^{\mathrm{T}}(\hat{P}y^+ + p_l y_l^+) = -a_l, \end{aligned}$$

has a solution (v, y^+, y_l^+) *such that* $y_l^+ > 0$. *Moreover, denoting the objective values of* y *and* (y^+, y_l^+) *by*

$$w(y, 0) = \tfrac{1}{2}\,|\hat{P}y|^2 + \hat{a}^{\mathrm{T}}y,$$
$$w(y^+, y_l^+) = \tfrac{1}{2}\,|\hat{P}y^+ + p_l y_l^+|^2 + \hat{a}^{\mathrm{T}}y^+ + a_l y_l^+,$$

respectively, one has

$$w(y^+, y_l^+) - w(y, 0) = \tfrac{1}{2}(v + p_l^{\mathrm{T}}\hat{P}y + a_l)y_l^+ < 0.$$

LEMMA 2.4 *Suppose that the assumptions of Lemma* 2.3 *hold and that* $y > \mathbf{0}$. *Let* t *have the largest value in* $[0, 1]$ *such that the* $(k+1)$*-vector* $z = (\bar{z}, \bar{z}_l) = t(y^+, y_l^+) + (1-t)(y, 0)$ *satisfies* $z \geq \mathbf{0}$. *Then* $\bar{z}_l > 0$, $e^{\mathrm{T}}z = 1$ *and* $w(\bar{z}, \bar{z}_l) < w(y, 0)$. *Furthermore, if* $y^+ > \mathbf{0}$ *then* $t = 1$ *and* $z = (y^+, y_l^+) > \mathbf{0}$, *while if* $y^+ > \mathbf{0}$ *does not hold, then* $z_c = 0$ *for some* $c \in \{1, \dots, k\}$.

Column augmentation cannot occur if

$$\operatorname{rank}\begin{bmatrix} e^{\mathrm{T}} \\ \hat{P} \end{bmatrix} = \operatorname{rank}\begin{bmatrix} e^{\mathrm{T}} & 1 \\ \hat{P} & p_l \end{bmatrix} = k. \tag{2.9}$$

Then a *column exchange* must be performed by appending l to J and deleting another index c from J. The following lemma suggests how to modify $\hat{y} = y$ so that the objective value is reduced.

LEMMA 2.5 *Suppose that* (v, y) *solves* (2.5) *and* (2.7) *holds for some* $l \notin J$ *such that* (2.9) *is satisfied. Let* $\tilde{y}$ *solve the system*

$$e^{\mathrm{T}}\tilde{y} = 1, \qquad \hat{P}\tilde{y} = p_l, \tag{2.10a,b}$$

and let

$$\begin{bmatrix} \bar{y} \\ \bar{y}_l \end{bmatrix} = \begin{bmatrix} -\tilde{y} \\ 1 \end{bmatrix}, \tag{2.11}$$

$$t = \max\{\bar{t} \geq 0 : y + \bar{t}\bar{y} \geq \mathbf{0}\}, \tag{2.12}$$

$$z = \begin{bmatrix} \bar{z} \\ \bar{z}_l \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} + t\begin{bmatrix} \bar{y} \\ \bar{y}_l \end{bmatrix}. \tag{2.13}$$

Then $t > 0$, $\bar{z}_l > 0$, $z \geq \mathbf{0}$, $e^{\mathrm{T}}z = 1$, *and*

$$w(\bar{z}, \bar{z}_l) - w(y, 0) = t(v + p_l^{\mathrm{T}}\hat{P}y + a_l) < 0. \tag{2.14}$$

Furthermore, $z_c = 0$ *for some* $c \in \{1, \dots, k\}$, *and for any such* c

$$\operatorname{rank}\begin{bmatrix} \hat{P}_e^- & 1 \\ & p_l \end{bmatrix} = \operatorname{rank}\hat{P}_e = k, \tag{2.15}$$

where $\hat{P}_e^-$ *is formed by deleting the* cth *column from* $\hat{P}_e$.

Proof. (2.9) ensures that $\tilde{y}$ is well-defined. By construction,

$$e^{\mathrm{T}}\bar{y} + \bar{y}_l = 0, \tag{2.16}$$

$$\hat{P}\bar{y} + p_l\bar{y}_l = \mathbf{0}, \tag{2.17}$$

and $\bar{y}_l = 1$, so $t > 0$ is well-defined by

$$t = \min \{-y_i/\bar{y}_i : \bar{y}_i < 0,\ 1 \leqslant i \leqslant k\}$$

because $y > \mathbf{0}$, and we have $\bar{z}_l = t\bar{y}_l > 0$, $z \geqslant \mathbf{0}$, and $e^{\mathrm{T}}z = 1$ from (2.5a) and (2.16). If $t = -y_c/\bar{y}_c$ for some $\bar{y}_c < 0$, then $\bar{z}_c = 0$ and, by (2.16)–(2.17),

$$\sum_{i=1, i \neq c}^{k} \begin{bmatrix} 1 \\ p_{j_i} \end{bmatrix} (-\bar{y}_i/\bar{y}_c) + \begin{bmatrix} 1 \\ p_l \end{bmatrix} (-1/\bar{y}_c) = \begin{bmatrix} 1 \\ p_{j_c} \end{bmatrix},$$

which combined with (2.9) yields (2.15). Multiplying the transpose of (2.5b) on the right by $\bar{y}$ and using (2.16)–(2.17), we get

$$-\hat{a}^{\mathrm{T}}\bar{y} = ve^{\mathrm{T}}\bar{y} + y^{\mathrm{T}}\hat{P}^{\mathrm{T}}\hat{P}\bar{y} = -v\bar{y}_l - y^{\mathrm{T}}\hat{P}^{\mathrm{T}}P_l\bar{y}_l$$

and by transposition and negation

$$\bar{y}_l(v + p_l^{\mathrm{T}}\hat{P}y + a_l) = \hat{a}^{\mathrm{T}}\bar{y} + a_l\bar{y}_l.$$

Combining this equality with (2.17) and the fact that $\bar{y}_l = 1$, we get

$$\begin{aligned} w(\bar{z}, \bar{z}_l) &= \tfrac{1}{2}\, |\hat{P}(y + t\bar{y}) + p_l t\bar{y}_l|^2 + \hat{a}^{\mathrm{T}}(y + t\bar{y}) + a_l t\bar{y}_l \\ &= \tfrac{1}{2}\, |\hat{P}y + t(\hat{P}\bar{y} + p_l\bar{y}_l)|^2 + \hat{a}^{\mathrm{T}}y + t(\hat{a}^{\mathrm{T}}\bar{y} + a_l\bar{y}_l) \\ &= \tfrac{1}{2}\, |\hat{P}y|^2 + \hat{a}^{\mathrm{T}}y + t\bar{y}_l(v + p_l^{\mathrm{T}}\hat{p}y + a_l) \\ &= w(y, 0) + t(v + p_l^{\mathrm{T}}\hat{P}y + a_l) \end{aligned}$$

and (2.14) follows from (2.7) and the positivity of t.

The test for deciding whether column augmentation or exchange should be performed is given in the following elementary result.

LEMMA 2.6 *Suppose that* (2.4) *holds. Then the following least squares version of* (2.13)

$$\hat{P}_e^{\mathrm{T}}\hat{P}_e\bar{y} = \hat{P}_e^{\mathrm{T}}\begin{bmatrix} 1 \\ p_l \end{bmatrix}, \qquad (2.18)$$

has a unique solution $\bar{y}$, which satisfies (2.10) *if and only if* (2.9) *holds.*

We may now state the method.

ALGORITHM 2.7
Step 0 (*Initialization*). Defining l by

$$\tfrac{1}{2}\, |p_l|^2 + a_l = \min \{\tfrac{1}{2}\, |p_j|^2 + a_j : 1 \leqslant j \leqslant m\},$$

set $k = 1$, $J = \{l\}$, $P = [p_l]$, $\hat{a} = (a_l)$, $\hat{y} = y = (1)$, and $v = -(|p_l|^2 + a_l)$.
Step 1 (*Stopping criterion*). If $v + p_j^{\mathrm{T}}\hat{P}\hat{y} + a_j \geqslant 0$ for each $j \notin J$, stop. Otherwise, choose $l \notin J$ satisfying $v + p_l^{\mathrm{T}}\hat{P}\hat{y} + a_l < 0$.
Step 2 (*Linear independence test*). Solve for $\bar{y}$ the system (2.18). If (2.10) holds, go to step 4; otherwise, continue.
Step 3 (*Column augmentation*). Set $\hat{y}_{k+1} = 0$, append l to J, p_l to $\hat{P}$, a_l to $\hat{a}$, increase k by 1, and go to Step 5.

Step 4 (*Column exchange*). Defining t by

$$t = \min\{\hat{y}_i/\bar{y}_i : \bar{y}_i > 0,\ 1 \leqslant i \leqslant k\},$$

replace $\hat{y}$ by $\hat{y} - t\bar{y}$. Find some $c \in \{1, \ldots, k\}$ such that $\hat{y}_c = 0$. Delete the cth index from J, the cth components from $\hat{y}$ and $\hat{a}$, and the cth column from $\hat{P}$. Set $\hat{y}_k = t$, append l to J, a_l to $\hat{a}$, and p_l to $\hat{P}$.
Step 5 (*Subproblem solution*). Solve for (v, y) the system (2.5). If $y > 0$, set $\hat{y} = y$ and go to Step 1. Otherwise, continue.
Step 6 (*Column deletion*). Defining t by (2.6), replace $\hat{y}$ by $ty + (1-t)\hat{y}$ and find $c \in \{1, \ldots, k\}$ such that $\hat{y}_c = 0$. Delete the cth index from J, the cth components from $\hat{y}$ and $\hat{a}$, the cth column from $\hat{P}$ and decrease k by 1. Go to Step 5.

We may now establish finite convergence of the algorithm.

THEOREM 2.8 *After a finite number of executions of Steps* 1 *through* 6, *Algorithm* 2.7 *terminates with a solution to problem* (1.1).

Proof. Collecting the preceding results as in Mifflin (1979), one can show that to each J considered at Step 1 there corresponds a unique solution $(v, y) = (v, \hat{y})$ of (2.5), and that at Step 1 the consecutive objective functions decrease strictly monotonically. Thus no J can enter Step 1 more than once. Since there is but a finite number of possible index sets J the algorithm must terminate, and it can only do so when the problem is solved.

An alternative method of establishing finite convergence consists in showing that our algorithm is equivalent to the general method of Best (1984) (when that method is applied to problem (1.1)). This task is left to the interested readers.

3. Solving the equations via Cholesky factorizations

Following Wolfe (1976), we can maintain the Cholesky factorization of the $k \times k$ matrix $\hat{P}_e^{\mathrm{T}}\hat{P}_e$ in the form

$$R^{\mathrm{T}}R = \hat{P}_e^{\mathrm{T}}\hat{P}_e = \hat{P}^{\mathrm{T}}\hat{P} + ee^{\mathrm{T}}, \tag{3.1}$$

where R is a $k \times k$ upper triangular nonsingular matrix. It is easy to check that if the k-vectors y_e and y_a satisfy

$$R^{\mathrm{T}}y_e = e, \qquad R^{\mathrm{T}}y_a = -\hat{a}, \tag{3.2}$$

then the solution (v, y) of system (2.5) can be computed from the relations

$$v = (|y_e|^2 + y_e^{\mathrm{T}}y_a - 1)/|y_e|^2, \qquad Ry = y_a + (1-v)y_e, \tag{3.3a,b}$$

where $y_e = R^{-\mathrm{T}}e \neq 0$. Also the solution of (2.18) is the result of solving the two systems

$$R^{\mathrm{T}}r = e + \hat{P}^{\mathrm{T}}p_l, \tag{3.4}$$

$$R\bar{y} = r. \tag{3.5}$$

At Step 0 we have $R = [(1 + |p_l|^2)^{\frac{1}{2}}]$. R must be altered whenever $\hat{P}_e$ is, and this can be done as follows (see, e.g. Gill, Golub, Murray and Saunders (1974), Seber (1977), Chapter 11 and Wolfe (1976)).

In order to append column $(1, \boldsymbol{p}_l)$ to $\hat{P}_e$ on the right, solve (3.4) for r and adjoin to R, on the right, the column (r, ρ), where $\rho = \sqrt{\rho^2}$ for

$$\rho^2 = 1 + |\boldsymbol{p}_l|^2 - |r|^2. \tag{3.6}$$

In general, ρ is the distance from $(1, \boldsymbol{p}_l)$ to the range of $\hat{P}_e$, i.e. $\bar{y}$ given by (3.5) is the unique least-squares solution of (2.10) with the residuals

$$\delta = \boldsymbol{e}^{\mathrm{T}}\bar{\boldsymbol{y}} - 1, \tag{3.7a}$$

$$\boldsymbol{\Delta} = \hat{P}\bar{\boldsymbol{y}} - \boldsymbol{p}_l \tag{3.7b}$$

satisfying

$$\rho^2 = \delta^2 + |\boldsymbol{\Delta}|^2. \tag{3.8}$$

Thus, by Lemma 2.6, the new $\hat{P}_e$ has full column rank if and only if ρ is positive and, hence, the new R is nonsingular.

In practice computer rounding error makes the linear independence test of Step 2 meaningless, since the computed residuals (3.7) do not, in general, vanish even when they should. Fortunately, we can enhance the numerical stability of the method by modifying Steps 2 and 4 so that even for nonzero but "small" residuals we still have column exchanges with objective decreases. To this end, we shall need the following extension of Lemma 2.5.

LEMMA 3.1 *Suppose that* $(v, \boldsymbol{y})$ *solves* (2.5), $\boldsymbol{y} > \boldsymbol{0}$, $\bar{\boldsymbol{y}}$ *solves* (2.18) *and there are* $\varepsilon_1 \in [0, 1)$, $\varepsilon_s \geq 0$, *and* $\varepsilon \geq 0$ *such that the residuals* (3.7) *satisfy*

$$\delta \geq -\varepsilon_1, \tag{3.9}$$

$$\delta^2 + |\boldsymbol{\Delta}|^2 \leq \varepsilon(1 + |\boldsymbol{p}_l|^2), \tag{3.10}$$

$$v + \boldsymbol{p}_l^{\mathrm{T}}\hat{P}\boldsymbol{y} + a_l \leq -\varepsilon_s(1 + |\boldsymbol{p}_l|^2) \tag{3.11}$$

$$\varepsilon \leq \varepsilon_s(1 - \varepsilon_1)^2/(1 + |\boldsymbol{p}_l|)^2 \tag{3.12}$$

for some $l \notin J$. *Let*

$$\begin{bmatrix} \tilde{y} \\ \tilde{y}_l \end{bmatrix} = \begin{bmatrix} -\bar{y} \\ \boldsymbol{e}^{\mathrm{T}}\bar{y} \end{bmatrix} \tag{3.13}$$

and define t *and* $\boldsymbol{z} = (\bar{z}, \bar{z}_l)$ *by* (2.12) *and* (2.13), *respectively. Then* $t > 0$, $\boldsymbol{z} \geq \boldsymbol{0}$, $\boldsymbol{e}^{\mathrm{T}}\boldsymbol{z} = 1$ *and* $z_c = 0$ *for some* $c \in \{1, \ldots, k\}$. *Moreover,*

$$\Delta_w := w(\bar{z}, \bar{z}_l) - w(y, 0) < 0.$$

Proof. Since $\boldsymbol{e}^{\mathrm{T}}\tilde{y} + \tilde{y}_l = 0$ and $\tilde{y}_l = \boldsymbol{e}^{\mathrm{T}}\bar{y} = 1 + \delta \geq 1 - \varepsilon_1 > 0$, at least one $\tilde{y}_i = -\bar{y}_i$ is negative, so $t > 0$ is well-defined by

$$t = \min\{y_i/\bar{y}_i : \bar{y}_i > 0,\ 1 \leq i \leq k\} \tag{3.14}$$

because $\boldsymbol{y} > \boldsymbol{0}$, and we have $\bar{z}_l = t\tilde{y}_l > 0$, $\boldsymbol{z} \geq \boldsymbol{0}$, and $\boldsymbol{e}^{\mathrm{T}}\boldsymbol{z} = \boldsymbol{e}^{\mathrm{T}}\boldsymbol{y} + t(\boldsymbol{e}^{\mathrm{T}}\tilde{y} + \tilde{y}_l) = \boldsymbol{e}^{\mathrm{T}}\boldsymbol{y} = 1$ from (2.5a). Next, by construction,

$$\begin{aligned}\Delta_w &= \tfrac{1}{2}\,|\hat{P}(y + t\tilde{y}) + \boldsymbol{p}_l t\tilde{y}_l|^2 + \hat{\boldsymbol{a}}^{\mathrm{T}}(y + t\tilde{y}) + a_l t\tilde{y}_l - \tfrac{1}{2}\,|\hat{P}\boldsymbol{y}|^2 - \hat{\boldsymbol{a}}^{\mathrm{T}}\boldsymbol{y} \\ &= \tfrac{1}{2}t^2\,|\hat{P}\tilde{y} + \boldsymbol{p}_l\tilde{y}_l|^2 + t[\hat{\boldsymbol{a}}^{\mathrm{T}}\tilde{y} + a_l\tilde{y}_l + \boldsymbol{y}^{\mathrm{T}}\hat{P}^{\mathrm{T}}(\hat{P}\tilde{y} + \boldsymbol{p}_l\tilde{y}_l)].\end{aligned}$$

Multiplying the transpose of (2.5b) on the right by $\bar{y}$ and using the fact that $\bar{y}_l = -\boldsymbol{e}^{\mathrm{T}}\bar{\boldsymbol{y}}$, we get

$$-\hat{\boldsymbol{a}}^{\mathrm{T}}\bar{\boldsymbol{y}} = v\boldsymbol{e}^{\mathrm{T}}\bar{\boldsymbol{y}} + \boldsymbol{y}^{\mathrm{T}}\hat{P}^{\mathrm{T}}\hat{P}\bar{\boldsymbol{y}} = -v\bar{y}_l + \boldsymbol{y}^{\mathrm{T}}\hat{P}^{\mathrm{T}}\hat{P}\bar{\boldsymbol{y}},$$

whereas, by (3.13) and (3.7), $\bar{y}_l = 1 + \delta$ and

$$\hat{P}\bar{\boldsymbol{y}} + \boldsymbol{p}_l\bar{y}_l = -\hat{P}\bar{\boldsymbol{y}} + \boldsymbol{p}_l(1+\delta) = -\boldsymbol{\Delta} + \boldsymbol{p}_l\delta,$$

so we may express Δ_w as

$$\Delta_w = \tfrac{1}{2}t^2\,|\boldsymbol{\Delta} - \boldsymbol{p}_l\delta|^2 + t(1+\delta)(v + \boldsymbol{p}_l^{\mathrm{T}}\hat{P}\boldsymbol{y} + a_l). \tag{3.15}$$

Hence we only need to prove that

$$\tfrac{1}{2}t\,|\boldsymbol{\Delta} - \boldsymbol{p}_l\delta|^2 < -(1+\delta)(v + \boldsymbol{p}_l^{\mathrm{T}}\hat{P}\boldsymbol{y} + a_l). \tag{3.16}$$

Since $\boldsymbol{e}^{\mathrm{T}}\boldsymbol{z} = 1$ and $\boldsymbol{z} \geq \boldsymbol{0}$, we have $1 \geq \bar{z}_l = t\bar{y}_l = t(1+\delta) \geq t(1-\varepsilon_1)$, so

$$t \leq 1/(1-\varepsilon_1). \tag{3.17}$$

Next, by (3.10),

$$|\boldsymbol{\Delta} - \boldsymbol{p}_l\delta|^2 \leq (|\boldsymbol{\Delta}| + |\delta|\,|\boldsymbol{p}_l|)^2 \leq \varepsilon(1 + |\boldsymbol{p}_l|^2)(1 + |\boldsymbol{p}_l|)^2. \tag{3.18}$$

Since $1 + \delta \geq 1 - \varepsilon_1 > 0$, it is easy to check that (3.17), (3.18), (3.11), and (3.12) imply (3.16), as desired.

We use Lemma 3.1 for modifying the algorithm as follows. The preceding analysis indicates that the augmented $\hat{P}_e$ should have full column rank if ρ^2 computed by (3.6) satisfies

$$\rho^2 > \varepsilon_c(1 + |\boldsymbol{p}_l|^2) \tag{3.19}$$

for some ε_c larger than the relative machine accuracy ε_{M}. If (3.19) is violated then, by (3.8), inequalities (3.9) and (3.10) should hold for some small positive ε_1 and ε_s; moreover, we should have (3.11) and (3.12) because $v + \boldsymbol{p}_l^{\mathrm{T}}\hat{P}\boldsymbol{y} + a_l < 0$ at Step 2. Then Lemma 3.1 indicates that we need only modify Step 4 by setting

$$\hat{y}_k = t\boldsymbol{e}^{\mathrm{T}}\bar{\boldsymbol{y}}$$

(cf. (2.11)–(2.13) and (3.13)) to ensure the necessary objective decrease $\Delta_w < 0$.

In fact, Lemma 3.1 shows that column exchanges may occur even when $\boldsymbol{p}_l$ is affinely independent from $\hat{P}$. To reduce the number of such exchanges, one may use the test

$$\Delta_w < \varepsilon_2 t(v + \boldsymbol{p}_l^{\mathrm{T}}\hat{P}\boldsymbol{y} + a_l) \tag{3.20}$$

for some small $\varepsilon_2 > 0$, where Δ_w is the objective decrease given by (3.15), whereas $t(v + \boldsymbol{p}_l^{\mathrm{T}}\hat{P}\boldsymbol{y} + a_l)$ is the "ideal" decrease (see (2.14)) for zero residuals.

Thus we arrive at the following modification of Step 2.

Step 2′ (*Linear independence test*).

(i) Solve (3.4) for r and compute $\tilde{\rho}^2 = \rho^2$ from (3.6).

(ii) If (3.19) holds, compute $\rho = \sqrt{\rho^2}$ and go to Step 3. Otherwise, continue.

(iii) Calculate $\bar{\boldsymbol{y}}$, δ, and $\boldsymbol{\Delta}$ from (3.5) and (3.7). If (3.9) does not hold, set

$$\rho = (\max\{\tilde{\rho}^2, \delta^2 + |\boldsymbol{\Delta}|^2\})^{\frac{1}{2}} \tag{3.21}$$

and go to Step 3. Otherwise, calculate t and Δ_w by (3.14)–(3.15).

(iv) If (3.20) holds, go to Step 4. Otherwise, compute ρ by (3.21) and go to Step 3.

Clearly, Theorem 2.9 remains valid for the modified method with any $\varepsilon_c \geq 0$, $\varepsilon_1 \in [0, 1)$, and $\varepsilon_2 \geq 0$; and we recover the original method if $\varepsilon_c = 0$. We note that when these tolerances are positive, the modified column exchange strategy causes our algorithm to be no longer equivalent to the methods analysed by Best (1984), Mifflin (1979), and Wolfe (1976).

The accuracy tolerances should allow for rounding error. Following the analysis of Wolfe (1976), at Step 1 we choose l to

$$\text{minimize} \quad v + p_j^{\mathrm{T}}\hat{P}y + a_j \quad \text{over all } j \in \{1, \ldots, m\} \tag{3.22}$$

and terminate if

$$v + p_l^{\mathrm{T}}\hat{P} + a_l \geq -\varepsilon_s(1 + |p_l|^2). \tag{3.23}$$

If $l \in J$, $y = \hat{y}$ is so inaccurate that it fails to satisfy (2.5b) to more than the degree measured by ε_s, then the algorithm stops. In our numerical experiments the algorithm always terminated with $l \notin J$ for $\varepsilon_s = 100\varepsilon_{\mathrm{M}}$ (see Section 4).

The test (3.19) is only used to reduce the effort involved in Step 2′ (iii) and the possible subsequent column exchange. When this test is passed, relation (3.19) indicates that the new R will not be approximately singular. At the same time, we expect $1 + |p_l|^2 - |r|^2$ to be of order $\varepsilon_{\mathrm{M}}(1 + |p_l|^2)$ even when it should vanish. Therefore, we suggest using $\varepsilon_c \geq 100\varepsilon_{\mathrm{M}}$.

We have found in practice that the use of $\varepsilon_c = 1$ and $\varepsilon_2 = 1$ in the test (3.20) leads to many unnecessary column exchanges at earlier iterations. Hence we use $\varepsilon_2 = 10^{-2}$.

To sum up, the modified column exchange strategy uses a rather sophisticated mechanism for controlling the values of the diagonal elements of R. This mechanism seems to work well in practice, although we cannot give guarantees on the condition number of $\hat{P}_e^{\mathrm{T}}\hat{P}_e$.

Observe that when column (r, ρ) is appended to R, (3.2) implies that the first k components of the new y_e and y_a do not change, whereas

$$\begin{aligned} y_{e,k+1} &= (1 - r^{\mathrm{T}}y_e)/\rho, \\ y_{a,k+1} &= (-a_l - r^{\mathrm{T}}y_a)/\rho. \end{aligned} \tag{3.24}$$

This updating of y_e and y_a saves the effort involved in computing them directly from (3.2).

Suppose the cth column of the $(n + 1) \times k$ matrix $\hat{P}_e$ is deleted at Step 4 (or Step 6). Then one may replace R by $\bar{R}$, y_e by $\bar{y}_e$ and y_a by $\bar{y}_a$ as follows. Delete the cth column of R to get a $k \times (k - 1)$ matrix $\tilde{R}$ and form an appropriate product of plane rotations (Givens) matrices $\bar{Q} = Q_k^{k-1}Q_k^{k-2} \cdots Q_{i+1}^{i}$ in order to reduce $\tilde{R}$ to a $(k - 1) \times (k - 1)$ upper triangular matrix $\bar{R}$ over a $(k - 1)$-dimensional zero vector. Form

$$\begin{bmatrix} \bar{R} & \bar{y}_e & \bar{y}_a \\ \mathbf{0}^{\mathrm{T}} & & \end{bmatrix} = \bar{Q}[\tilde{R} \quad y_e \quad y_a] \tag{3.25}$$

and drop the last components of $\bar{y}_e$ and $\bar{y}_a$ (see, e.g. Gill *et al.* (1974), Daniel *et al.* (1976), Mifflin (1979) for details).

We shall now give the amount of work required by each step of the algorithm as the highest-order terms in the expressions for the number of multiplications. We assume the three-multiplication form of a plane rotation (see Gill *et al.* (1974)).

Step 1 takes $n \times k$ multiplications to form $\hat{P}\hat{y}$ and $n \times m$ to compute l (see (3.22)). Step 2 takes $n \times k$ to form $\hat{P}^{\mathrm{T}}\boldsymbol{p}_l$, then $\frac{1}{2}k^2$ to solve (3.4), $\frac{1}{2}k^2$ to solve (3.5), and $k(n+1)$ to compute (3.7). At Step 3, equations (3.24) require $2k$ multiplications. At Step 4 the rotations of (3.25) take $\frac{3}{2}(k-c)^2$, and the solution of (3.4) takes $\frac{1}{2}k^2$ ($\hat{P}^{\mathrm{T}}\boldsymbol{p}_l$ is available from Step 2). Step 5 takes $\frac{1}{2}k^2$ (see (3.3b)). The rotations (3.25) of Step 6 take $\frac{3}{2}(k-c)^2$, which has the expected value $\frac{1}{2}k^2$ if c is randomly, uniformly chosen.

The algorithm was programmed in ANSI (1966) Standard Fortran as a set of subroutines called QPDF1 (Quadratic Programming for Direction Finding, Version 1). In addition to the data P and $\boldsymbol{a}$, the user of QPDF1 provides three real arrays for storing $\boldsymbol{x}$, $P\boldsymbol{x}$ and $|\boldsymbol{p}_j|^2$ for $j = 1, \ldots, m$, an integer array of length k_m for storing J and a real array of length

$$l_w = \tfrac{1}{2}k_m^2 + 5k_m + 10$$

for maintaining information for the solution of the equations, where $k_m \geqslant \min\{m, n+1\}$. Such storage organization is convenient when QPDF1 is called repetitively to solve a sequence of related problems.

Thus suppose that the algorithm has solved problem (1.1), but we now want to solve a modified version of (1.1) in which certain columns of P and $\boldsymbol{a}^{\mathrm{T}}$ corresponding to zero components of $\boldsymbol{x}$ have been deleted and some new columns have been added. If the information gathered so far is retained and the terminal value of J is modified consistently with the new problem description, the algorithm may start at Step 2 (or Step 5 if the remaining components of $\boldsymbol{a}$ have been changed). This may save computational effort (see Wolfe (1976)).

The method of calculation discussed so far uses the Cholesky factorization $R^{\mathrm{T}}R$ of the product $\hat{P}_e^{\mathrm{T}}\hat{P}_e$. Daniel *et al.* (1976), among others, advocate using the QR factorization

$$\hat{P}_e = QR$$

of the $(n+1) \times (n+1)$ matrix $\hat{P}_e$, where Q is an $(n+1) \times k$ matrix with orthonormal columns and R is a nonsingular upper triangular $k \times k$ matrix. Then relations (3.4) and (3.6) may be replaced by

$$\boldsymbol{r} = Q^{\mathrm{T}}\begin{bmatrix} 1 \\ \boldsymbol{p}_l \end{bmatrix},$$

$$\boldsymbol{q} = \begin{bmatrix} 1 \\ \boldsymbol{p}_l \end{bmatrix} - Q\boldsymbol{r}, \tag{3.26}$$

$$\rho^2 = \boldsymbol{q}^{\mathrm{T}}\boldsymbol{q}, \tag{3.27}$$

and Q is augmented with $\bar{q} = q/\rho$ at Step 3, whereas (3.25) is complemented by

$$\begin{bmatrix} \bar{Q}^{\mathrm{T}} \\ \bar{q}^{\mathrm{T}} \end{bmatrix} = \bar{Q}Q^{\mathrm{T}}, \tag{3.28}$$

in which $\bar{Q}$ is the new Q. We refer the reader to Daniel *et al.* (1976) for the possible use of reorthogonalization to ensure sufficient accuracy in Q and R. Compared with the previous method, this method requires additional storage of Q (about n^2 locations if $m \geqslant n$), saves the $\frac{1}{2}k^2$ multiplications required by the solution of (3.4), but takes additional $n \times k$ for (3.26) and $3n(k-c)$ for (3.28).

The second method of calculation may give more accurate Cholesky factors. In particular, ρ^2 computed from (3.27) cannot be negative. On the other hand, when R is corrupted by rounding errors at column deletions, (3.6) may yield $\rho^2 \leqslant 0$ at Step 4. In this case we have to refactorize $\hat{P}_e^{\mathrm{T}}\hat{P}_e$, i.e. we recompute R *ab initio* using (3.4)–(3.6) recursively. This happens very infrequently, see Section 4.

4. Numerical Experience

We shall now report on computational testing of the algorithm QPDF1 using double precision on a PDP 11/70 computer with relative accuracy $\varepsilon_{\mathrm{M}} = 2^{-56} = 1.4 \times 10^{-17}$.

Our main goal was to test the accuracy and stability of the method on some rather ill-conditioned problems. In most applications, not much accuracy is required of the solution x of (1.1). For instance, problems of the form (1.3) arising in nonsmooth optimization frequently have (practically) nonunique Lagrange mulipliers x, and it is the accuracy of (d, v) given by (1.4) rather than that of x itself that matters. Hence we also compute for the final x the estimates

$$\begin{aligned} \hat{d} &= -Px, \\ \hat{v} &= -(|Px|^2 + a^{\mathrm{T}}x), \\ \tilde{v} &= \max\{-a_j + p_j^{\mathrm{T}}\hat{d} : j = 1, \ldots, m\} \end{aligned}$$

of the optimal solution (d, v) of problem (1.3).

To test the stability of the method, we solved sequences of related problems in the way described in Section 3, using refactorizations only when they were necessary at Step 4.

Our test examples are generated as follows. We choose $n \geqslant 2$, $m = 2n + 2$, $p_{ij} = j/(i + j)$ for $i = 1, \ldots, n$, $j = 1, \ldots, m$. The sequence of related problems is generated by letting j_a run through the positive integers. Let $\hat{\jmath} = 1 + \mathrm{mod}\,(j_a - 1, m)$, and let

$$\hat{J} = \begin{cases} \{\hat{\jmath}, \hat{\jmath} + 1, \ldots, \hat{\jmath} + n\} & \text{if } \hat{\jmath} \leqslant n + 2, \\ \{1, \ldots, \hat{\jmath} - n - 2\} \cup \{\hat{\jmath}, \ldots, m\} & \text{if } \hat{\jmath} > n + 2. \end{cases}$$

Given $b \geq 0$, let

$$\bar{x}_j = \begin{cases} 1/(n+1) & \text{if } j \in \hat{J}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\bar{v} = \min\{-p_j^{\mathrm{T}} P\bar{x} : j = 1, \ldots, m\},$$

$$a_j = \begin{cases} -\bar{v} - p_j^{\mathrm{T}} P\bar{x} & \text{if } j \in \hat{J}, \\ -v - p_j^{\mathrm{T}} P\bar{x} + b & \text{otherwise,} \end{cases}$$

so that we know the solutions $(\bar{d}, \bar{v}) = (-P\bar{x}, \bar{v})$ of (1.3) and $\bar{x}$ of (1.1) with its objective value $\bar{w} = \frac{1}{2}|P\bar{x}|^2 + a^{\mathrm{T}}\bar{x}$. Note that for $b > 0$ precisely $n+1$ constraints indexed by $j \in \hat{J}$ are active at the solution of (1.3). The columns of $\hat{P}_e$ are multiples of the columns of the Hilbert section, with entries $h_{ij} = 1/(i+j-1)$ $(i = 1, \ldots, n+1, j = 1, \ldots, m)$, and we must have considerable loss of accuracy in maintaining the Cholesky factorization (3.1). Also the numercial rank of $\hat{P}_e$ (Stewart, 1984) may be less than its theoretical value k.

The integer $\hat{j}$ cycles through $\{1, \ldots, m\}$ with increasing j_a. Cycle 0 begins at $j_a = \hat{j} = 1$, and cycle $\bar{m}$ begins at $j_a = \bar{m}m + 1$ and $\hat{j} = 1$, so that the same problem is solved again.

In describing the results, we denote by ε_v, ε_w, and ε_d and ε_x the relative accuracies of the computed estimates of $\bar{v}$, $\bar{w}$, and all components of $\bar{d}$ and $\bar{x}$, respectively. For example, we let i maximize $|\bar{d}_i - \hat{d}_i|/(1 + |\bar{d}_i|)$ and set ε_d to the corresponding value, whereas $\varepsilon_v = |\bar{v} - v|/(1 + |\bar{v}|)$. We also compute

$$\bar{\varepsilon}_v = \max\{|v - \hat{v}|, |v - \bar{v}|, |\hat{v} - \bar{v}|\}/(1 + |v|)$$

to indicate the accuracy in the solution of the final Kuhn–Tucker system.

For a given run of the algorithm, we denote by NAUG, NEXC and NDEL the total numbers of column augmentations, exchanges and deletions performed at Steps 3, 4, and 6, respectively, when solving the problems corresponding to cycles 0 through $\bar{m}$. ITER is the total number of executions of Step 5, IDEL is the total number of restorations of R to upper triangular form through plane rotations at Steps 4 and 6, and NREF is the total number of refactorizations at Step 4.

Table 4.1 gives results for $b = 10^{10}$ and "standard" values of $\varepsilon_s = \varepsilon_c = 10^{-15}$ (=1 E-15). The total loss of accuracy in x for $n \geq 10$ is not surprising, since the final matrix $\hat{P}_e^{\mathrm{T}}\hat{P}_e$ is very ill-conditioned. Table 4.2 shows that there is little accumulation of rounding errors, occurring mainly at the first cycle.

Table 4.3 shows that a significant loss of efficiency may occur if the final accuracy tolerance ε_s is smaller than the error in the solution of the Kuhn–Tucker systems. Then the usual stopping criterion (3.23) is too stringent, and later iterations have to deal with matrices $\hat{P}_e$ with almost linearly dependent columns by employing column exchanges. Note that the use of larger tolerances ε_c may help reduce the number of refactorizations.

We may add that essentially the same results have been obtained for problems with $b = 0$.

The overall accuracy of the method seems to be limited by the error arising in

TABLE 1
Results for $\varepsilon_s = \varepsilon_c = 10^{-15}$, $b = 10^{10}$

n	Cycle	ITER	NAUG	NEXC	NDEL	IDEL	$\bar{\varepsilon}_v$	ε_v	ε_d	ε_w	ε_x
2	0	2	2	0	0	0	2E-17	3E-16	5E-16	9E-18	1E-13
	10	282	112	0	110	90	3E-17	3E-16	2E-15	2E-17	6E-13
3	0	3	3	0	0	0	4E-17	6E-16	3E-14	1E-17	1E-10
	10	403	163	0	160	158	1E-17	3E-15	1E-13	7E-18	5E-10
4	0	4	4	0	0	0	4E-17	5E-15	1E-12	4E-17	1E-07
	10	644	274	0	270	200	0	5E-15	1E-12	1E-17	1E-07
5	0	5	5	0	0	0	2E-17	1E-14	1E-11	0	4E-05
	10	845	365	0	360	290	0	5E-15	8E-12	6E-17	2E-05
10	0	6	6	0	0	0	1E-16	1E-13	5E-10	1E-16	2E-01
	10	1308	547	0	541	345	2E-16	3E-13	2E-09	6E-17	2E-01
20	0	9	8	0	1	1	1E-16	7E-13	5E-09	5E-16	3E-01
	10	2269	928	0	921	581	1E-16	4E-13	2E-09	2E-16	2E-01
30	0	10	9	0	1	1	1E-16	2E-14	3E-10	2E-17	2E-01
	10	2954	1171	0	1163	672	1E-16	9E-14	8E-10	0	2E-01

the solution of the Kuhn–Tucker system (2.5). This error is of order 10^{-16} for the above-described examples. For instance, we always had $|e^T x - 1| < 2 \times 10^{-16}$ for the final x.

We have also solved several other test examples (with $n \leq 30$ and $m \leq 80$) related more directly to applications in nondifferentiable optimization. They are not described here due to space limitation (see Kiwiel (1985)). We have found that the algorithm's performance on the other problems is comparable to that reported above, and that the accuracy was high enough for these applications.

TABLE 2
Error growth for $n = 30$, $b = 10^{10}$, $\varepsilon_s = \varepsilon_c = 10^{-15}$

Cycle	ITER	NAUG	NDEL	ε_v	ε_d	ε_w
0	10	9	1	2E-14	3E-10	2E-17
1	348	147	139	5E-13	3E-10	6E-16
3	950	386	378	8E-14	8E-10	5E-16
5	1526	612	604	1E-13	7E-10	1E-15
8	2386	949	941	1E-13	7E-10	5E-16
10	2954	1171	1163	9E-14	8E-10	0

TABLE 3
Influence of tolerances ε_s and ε_c ($n = 30$, $b = 10^{10}$)

ε_s	ε_c	Cycle	ITER	NAUG	NEXC	NDEL	IDEL	NREF	$\bar{\varepsilon}_v$	ε_v	ε_d
1E-15	1E-15	0	10	9	0	1	1	0	2E-16	2E-14	3E-10
		10	2954	1171	0	1163	672	0	9E-16	9E-14	8E-10
1E-14	1E-15	0	10	9	0	1	1	0	1E-16	2E-14	3E-10
		10	2926	1157	0	1149	585	0	6E-17	7E-14	6E-10
1E-16	1E-15	0	10	9	0	1	1	0	1E-16	2E-14	3E-10
		10	3574	1427	109	1418	959	0	1E-16	1E-13	1E-09
1E-15	1	0	11	8	3	0	3	0	5E-16	3E-14	3E-10
		10	2943	1031	269	1023	738	0	3E-16	7E-14	8E10
0	1E-15	0	12	9	2	1	3	0	5E-16	2E-13	2E-09
		10	4789	1520	1145	1511	—	8	4E-16	1E-13	1E-09
0	1E-10	0	12	9	2	1	3	0	5E-16	2E-13	2E-09
		10	6534	2287	2335	2279	—	5	2E-16	8E-15	3E-10
0	1	0	16	9	7	0	7	0	2E-16	1E-14	2E-10
		10	4621	1306	1398	1298	—	2	2E-16	2E-13	1E-09

5. Conclusions

We have presented a dual method for solving search direction finding subproblems of certain recently developed nonsmooth optimization algorithms. The method is a highly specialized version of the active-set algorithm for positive semi-definite quadratic programming.

Our implementation with the Cholesky factorization seems to be quite efficient and requires relatively little storage. To gain more numerical stability, we are at present working on implementations that recur the QR-factorization of a certain matrix. They will, however, require three times as much storage. These implementations, as well as the possible use of the Powell (1981) approach, will be reported elsewhere.

Acknowledgements

I would like to thank Andrzej Stachurski, Michael A. Saunders and Andrzej Ruszczynski for comments on an earlier draft of the paper. Also the suggestions of the Associate Editor and two referees are gratefully acknowledged.

References

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

BEST, M. J., & RITTER, K. 1976 An effective algorithm for quadratic minimization problems. *MRC Technical Summary Report* No. 1961, University of Wisconsin.

DANIEL, J. W., GRAGG, W. B., KAUFMAN, L., & STEWART, G. W. 1976 Re-orthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp.* **30,** 772–792.

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

FLETCHER, R. 1971 A general quadratic programming algorithm. *J. Inst. Math. Appl.* **7,** 76–91.

FLETCHER, R. 1981 *Practical Methods of Optimization* II: *Constrained Optimization.* New York: Wiley.

GILL, P. E., GOLUB, G. H., MURRAY, W., & SAUNDERS, M. A. 1974 Methods for modifying matrix factorizations. *Math. Comp.* **28,** 505–535.

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

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

KIWIEL, K. C. 1983 An aggregate subgradient method for nonsmooth convex minimization. *Math. Programming* **27,** 320–341.

KIWIEL, K. C. 1985 *Methods of Descent for Nondifferentiable Optimization.* Lecture Notes in Mathematics 1133. Berlin: Springer-Verlag.

LEMARECHAL, C. 1977 Nonsmooth optimization and descent methods. RR. 78-4, International Institute for Applied Systems Analysis, Laxenburg, Austria.

MIFFLIN, R. 1979 A stable method for solving certain constrained least squares problems. *Math. Programming* **16,** 141–158.

MIFFLIN, R. 1982 A modification and an extension of Lemaréchal's algorithm for nonsmooth minimization. In *Nondifferential and Variational Techniques in Optimization* (Sorensen, D. C. & Wets, R. J. B. Eds), Mathematical Programming Study 17, pp. 77–90. Amsterdam: North-Holland.

PANG, J. C. 1981 An equivalence between two algorithms for quadratic programming. *Math. Programming* **20,** 152–165.

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

SEBER, G. A. F. 1977 *Linear Regression Analysis.* New York: Wiley.

STOER, J. 1971 On the numerical solution of constrained least squares problems. *SIAM J. numer. Anal.* **8,** 382–411.

SCHITTKOWSKI, K. 1983 The numerical solution of constrained least squares problems. *IMA J. numer. Anal.* **3,** 11–36.

SCHITTKOWSKI, K., & STOER, J. 1979 A factorization method for constrained least squares problems with data changes. *Numer. Math.* **31,** 431–463.

STEWART, G. W. 1984 Rank degeneracy. *SIAM J. sci. stat. Comput.* **5,** 403–413.

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

WIERZBICKI, A. P. 1982 Lagrangian functions in nondifferentiable optimization. In *Progress in Nondifferentiable Optimization* (Nurminski, E. A., Ed.), pp. 173–213. CP-82-S8, International Institute for Applied Systems Analysis, Laxenburg, Austria.

WOLFE, P. 1976 Finding the nearest point in a polytope. *Math. Programming* **11,** 128–149.