Mathematical Programming 27 (1983) 1–33
North-Holland

A NUMERICALLY STABLE DUAL METHOD FOR SOLVING STRICTLY CONVEX QUADRATIC PROGRAMS*

D. GOLDFARB

Columbia University, Department of Industrial Engineering and Operations Research, New York, 10027, U.S.A.

A. IDNANI

Bell Laboratories, Murray Hill, N.J. 07974, U.S.A.

Received 20 October 1981
Revised manuscript received 6 August 1982

An efficient and numerically stable dual algorithm for positive definite quadratic programming is described which takes advantage of the fact that the unconstrained minimum of the objective function can be used as a starting point. Its implementation utilizes the Cholesky and QR factorizations and procedures for updating them. The performance of the dual algorithm is compared against that of primal algorithms when used to solve randomly generated test problems and quadratic programs generated in the course of solving nonlinear programming problems by a successive quadratic programming code (the principal motivation for the development of the algorithm). These computational results indicate that the dual algorithm is superior to primal algorithms when a primal feasible point is not readily available. The algorithm is also compared theoretically to the modified-simplex type dual methods of Lemke and Van de Panne and Whinston and it is illustrated by a numerical example.

Key words: Positive Definite Quadratic Programming, Matrix Factorizations, Dual Algorithms, Successive Quadratic Programming Methods.

1. Introduction

We are concerned in this paper with the strictly convex (positive definite) quadratic programming problem

$$\underset{x}{\text{minimize}} \quad f(x) = a^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Gx, \tag{1.1a}$$

$$\text{subject to} \quad s(x) \equiv C^{\mathrm{T}}x - b \geq 0 \tag{1.1b}$$

where x and a are n-vectors, G is an $n \times n$ symmetric positive definite matrix, C is an $n \times m$ matrix, b is an m-vector, and superscript T denotes the transpose. Although the vector of variables x may also be subject to equality constraints

$$\hat{C}^{\mathrm{T}}x - \hat{b} = 0 \tag{1.2}$$

*This research was supported in part by the Army Research Office under Grant No. DAAG 29-77-G-0114 and in part by the National Science Foundation under Grant No. MCS-6006065.

we shall ignore such constraints for the moment in order to simplify our presentation.

Attention has recently been drawn to the need for efficient and robust algorithms to solve (1.1) by the development of successive (recursive) quadratic programming methods for solving general nonlinear programming problems [4, 21, 30, 40]. These methods, which have been highly successful (e.g. see [35]), require the solution of a strictly convex quadratic program to determine the direction of search at each iteration.

Several approaches and numerous algorithms have been proposed for solving quadratic programming problems. These include the primal methods of Beale [2, 3], Dantzig [10], Fletcher [13], Goldfarb [16], Bunch and Kaufman [5], Gill and Murray [15], Murray [28] and Wolfe [41], the dual methods of Lemke [26] and Van de Panne and Whinston [39], the principal pivoting methods of Cottle and Dantzig [7], the parametric methods of Grigoriadis and Ritter [20] and Ritter [32], the primal–dual method of Goncalves [19], the methods for constrained least squares of Stoer [37], Schittkowski and Stoer [36], Lawson and Hanson [25], and Mifflin [27], the exact penalty function methods of Conn and Sinclair [6] and Han [22], and the subproblem optimization method of Theil and Van de Panne [38].

The above methods can for the most part be categorized as either modified simplex type methods or projection methods. The former perform simplex type pivots on basis matrices or tableaux of row size $(m + n)$ that are derived from the Kuhn–Tucker optimality conditions for the QPP (1.1). The latter are based upon projections onto active sets of constraints and employ operators of size no larger than $(n \times n)$. Consequently projection methods are usually more efficient and require less storage than methods of the modified simplex type. In this paper we present a projection type dual algorithm for solving the QPP (1.1), which was first given in Idnani [24], and outline an implementation that is both efficient and numerically stable.

Most of the work in quadratic programming has been, in general, a two-phase approach; in phase 1 a feasible point is obtained and then in phase 2 optimality is achieved while maintaining feasibility. Our computational experience indicates that, unless a feasible point is available, on the average between one-third to one-half of the total effort required to solve a QPP using typical primal algorithms is expended in phase 1. (See Tables 1, 2 and 3, for example.) One way to reduce the total work is to use a phase 1 approach that is likely to result in a near-optimal feasible point. This was suggested by Idnani [23] who used the unconstrained minimum of (1.1a), $x = -G^{-1}a$, as the starting point for the variant of Rosen's feasible point routine [33] proposed by Goldfarb [16]. The same suggestion of a starting point was made by Dax in [9]. Computational testing indicated that this approach usually found a feasible point that was also optimal. When it did not only very few additional iterations in phase 2 were required to obtain optimality.

Although these results were encouraging, they suggested that a dual approach would be even better. In a dual method for the strictly convex QPP one must first provide a dual feasible point, that is, a primal optimal point for some subproblem of the original problem. By relaxing all of the constraints (1.1b), the unconstrained minimum of (1.1a) is such a point. A dual algorithm then iterates until primal feasibility (i.e. dual optimality) is achieved, all the while maintaining the primal optimality of intermediate subproblems (i.e. dual feasibility). This is equivalent to solving the dual by a primal method (which can handle the positive semi-definite case). The important observation to make is that the origin in the space of dual variables is always dual feasible, so that no phase 1 is needed.

In the next section we present our basic approach for solving problem (1.1). Our discussion is in terms of this problem rather than in terms of the problem dual to it (cf. [26, 39]) as we believe this to be more instructive. We also introduce some notation in this section along with some other preliminaries. The dual algorithm is given in Section 3 where its validity and finite termination are proved. A particular numerically stable and efficient implementation of the algorithm is described in Section 4. In Section 5 we give the results of computational tests performed on randomly generated quadratic programs while in Section 6 the performance of our algorithm when used in a successive quadratic programming code is described. In both of these sections the computational performance of the dual algorithm is compared against that of primal algorithms. Comparisons between our dual algorithm and other modified simplex type dual algorithms for quadratic programming are given in Section 7. In Section 8 we make some remarks on alternative primal approaches and implementations of our dual algorithm and comment upon its superior behavior in degenerate and 'near' degenerate situations. We also provide an appendix in which we work through an example to illustrate the various parts of the dual algorithm.

2. Basic approach and preliminaries

The dual algorithm that is described in the next section is of the active set type. By an *active set* we mean a subset of the m constraints in (1.1b) that are satisfied as equalities by the current estimate x of the solution to the QPP (1.1). We shall use K to denote the set $\{1, 2, \dots, m\}$ of indices of the constraints (1.1b) and $A \subseteq K$ to denote the indices of the active set.

We define a *subproblem* P(J) to be the QPP with the objective function (1.1a) subject only to the subset of the constraints (1.1b) indexed by $J \subset K$. For example P($\emptyset$), where $\emptyset$ denotes the empty set, is the problem of finding the unconstrained minimum of (1.1a).

If the solution x of a subproblem P(J) lies on some linearly independent active set of constraints indexed by $A \subseteq J$ we call (x, A) a *solution*-(S) *pair*. Clearly, if (x, A) is an S-pair for subproblem P(J) it is also an S-pair for the subproblem

P(A). By *linear independence* of a set of constraints we mean that the normals corresponding to these constraints are linearly independent. We shall denote the normal vector of the ith constraint in (1.1b), i.e., the ith column of C, by n_i. We can now outline our basic approach for solving the strictly convex QPP (1.1).

Basic approach

Step 0: Assume that some S-pair (x, A) is given.
Step 1: Repeat until all constraints are satisfied:
(a) Choose a violated constraint $p \in K \setminus A$.
(b) If P($A \cup \{p\}$) is infeasible, STOP—the QPP is infeasible.
(c) Else, obtain a new S-pair $(\bar{x}, \bar{A} \cup \{p\})$ where $\bar{A} \subseteq A$ and $f(\bar{x}) > f(x)$ and set $(x, A) \leftarrow (\bar{x}, \bar{A} \cup \{p\})$.
Step 2: STOP—x is the optimal solution to the QPP.

Since the unconstrained minimum of (1.1),

$$x^0 = -G^{-1}a$$

is readily available, one can always start the above procedure with the S-pair $(x^0, \emptyset)$. The most important and interesting part of the above approach is Step 1(c). This step is realizable since it is always possible to implement it as

Step 1(c′): Determine an S-pair $(\hat{x}, \hat{A})$ by solving P($A \cup \{p\}$) and set $(x, A) \leftarrow (\hat{x}, \hat{A})$. (It is easy to show that $f(\hat{x}) > f(x)$ and that $\hat{A}$ must contain p.)

This implementation is more restrictive than Step 1(c) since Step 1(c′) requires that $\hat{x}$ satisfies all constraints indexed by A while Step 1(c) does not. In the next section we give a *dual* algorithm to implement the above basic approach which effects Step 1(c) so that dual feasibility is maintained at every point along the solution path. A *primal–dual* algorithm which incorporates Step 1(c′) is described and compared to our dual algorithm in [18, 24].

In order to describe our algorithm, it is necessary to introduce some notation. The matrix of normal vectors of the constraints in the active set indexed by A will be denoted by N and the cardinality of A will be denoted by q. A^+ will denote the set $A \cup \{p\}$ where p is in $K \setminus A$ and A^- will denote a proper subset of A containing one fewer element than A. N^+ and N^- will represent the matrices of normals corresponding to A^+ and A^-, respectively. Also, we shall use I (or I_k) to denote the $k \times k$ identity matrix and e_j to denote the jth column of I. n^+ will indicate the normal vector n_p added to N to give N^+ and n^- will indicate the column deleted from N to give N^-.

When the columns of N are linearly independent one can define the operators

$$N^* = (N^TG^{-1}N)^{-1}N^TG^{-1} \tag{2.1}$$

and

$$H = G^{-1}(I - NN^*) = G^{-1} - G^{-1}N(N^TG^{-1}N)^{-1}N^TG^{-1}. \tag{2.2}$$

N^* is the so-called pseudo-inverse or Moore–Penrose generalized inverse of N in the space of variables obtained under the transformation $y = G^{1/2}x$. H is a 'reduced' inverse Hessian operator for the quadratic $f(x)$ in (1.1a) subject to the active set of constraints. In particular, if $\hat{x}$ is a point in the $(n-q)$ dimensional manifold $\mathcal{M} = \{x \in \mathbb{R}^n \mid n_i^Tx = b_i, i \in A\}$ and $g(\hat{x}) \equiv \nabla f(\hat{x}) = G\hat{x} + a$ is the gradient of $f(x)$ at $\hat{x}$ then the minimum of $f(x)$ over $\mathcal{M}$ is attained at $\bar{x} = \hat{x} - Hg(\hat{x})$. For $\bar{x}$ to be the optimal solution for the subproblem P(A) we must have that

$$g(\bar{x}) = Nu(\bar{x})$$

where the vector of Lagrange multipliers $u(\bar{x}) \geq 0$. It follows from the definitions of N^* and H that at such a point

$$u(\bar{x}) \equiv N^*g(\bar{x}) \geq 0 \tag{2.3}$$

and

$$Hg(\bar{x}) = 0. \tag{2.4}$$

It is well known that these conditions are sufficient as well as necessary for x to be the optimal solution to P(A).

We note that the Lagrange multipliers defined by (2.3) are independent of x on the manifold $\mathcal{M}$. This is not the case for the first-order multipliers $(N^TN)^{-1}N^Tg(x)$ which equal $N^*g(x)$ only at the optimal point in $\mathcal{M}$. In the next section we shall make use of another set of multipliers

$$r = N^*n^+ \tag{2.5}$$

which we call infeasibility multipliers.

As the operators N^* and H are fundamental to the algorithm described in the next section we present some of their properties which will be useful later.

Properties.

$$Hw = 0 \quad \text{iff} \quad w = N\alpha, \tag{2.6}$$

H is positive semi-definite, (2.7)

$$HGH = H, \tag{2.8}$$

$$N^*GH = 0, \tag{2.9}$$

$$HH^+ = H^+. \tag{2.10}$$

In (2.10) above H^+ denotes the operator (2.2) with N replaced by N^+. Similar notation will be used for N^* and $u = N^*g$.

3. The dual algorithm

The algorithm given below follows the dual approach described in the previous section and makes use of the operators H and N^* defined there. In our implementation we do not explicitly compute or store these operators, rather we store and update the matrices $J = Q^{\mathrm{T}}L^{-1}$ and R obtained from the Cholesky and QR factorizations $G = LL^{\mathrm{T}}$ and $L^{-1}N = Q\left[\begin{smallmatrix}R\\0\end{smallmatrix}\right]$. This is described in Section 4.

Dual algorithm:

Step 0: *Find the unconstrained minimum*:
Set $x \leftarrow -G^{-1}a$, $f \leftarrow \frac{1}{2}a^{\mathrm{T}}x$, $H \leftarrow G^{-1}$, $A \leftarrow \emptyset$, $q \leftarrow 0$.
Step 1: *Choose a violated constraint, if any*:
Compute $s_j(x)$, for all $j \in K \setminus A$.
If $V = \{j \in K \setminus A \mid s_j(x) < 0\} = \emptyset$, STOP, the current solution x is both feasible and optimal;
otherwise, choose $p \in V$ and set $n^+ \leftarrow n_p$ and $u^+ \leftarrow \binom{u}{0}$.
If $q = 0$, set $u \leftarrow 0$. ($A^+ = A \cup \{p\}$.)
Step 2: *Check for feasibility and determine a new S-pair*:
(a) *Determine step direction*
Compute $z = Hn^+$ (the step direction in the primal space) and if $q > 0$, $r = N^*n^+$ (the negative of the step direction in the dual space).
(b) *Compute step length*
(i) *Partial step length*, t_1: (maximum step in dual space without violating dual feasibility).
If $r \le 0$ or $q = 0$, set $t_1 \leftarrow \infty$;
otherwise, set

$$t_1 \leftarrow \min_{\substack{r_j > 0\\ j=1,\dots,q}} \left\{\frac{u_j^+(x)}{r_j}\right\} = \frac{u_l^+(x)}{r_l}.$$

In Step 2(c) below, element $k \in K$ corresponds to the lth element in A.
(ii) *Full step length* t_2: (minimum step in primal space such that the pth constraint becomes feasible).
If $|z| = 0$, set $t_2 \leftarrow \infty$;
otherwise, set $t_2 \leftarrow -s_p(x)/z^{\mathrm{T}}n^+$.
(iii) *Step length*, t:
Set $t \leftarrow \min(t_1, t_2)$.

(c) *Determine new S-pair and take step*:

(i) *No step in primal or dual space*:

If $t = \infty$, STOP, subproblem $\mathrm{P}(A^+)$ and hence the QPP are infeasible.

(ii) *Step in dual space*:

If $t_2 = \infty$, then set $u^+ \leftarrow u^+ + t\binom{-r}{1}$, and drop constraint k; i.e. set $A \leftarrow A \setminus \{k\}$, $q \leftarrow q - 1$, update H and N^*, and go to Step 2(a).

(iii) *Step in primal and dual space*:

Set $x \leftarrow x + tz$,
$f \leftarrow f + tz^{\mathrm{T}} n^+ (\frac{1}{2}t + u^+_{q+1})$,
$u^+ \leftarrow u^+ + t\binom{-r}{1}$.
If $t = t_2$ (full step) set $u \leftarrow u^+$ and add constraint p; i.e. set $A \leftarrow A \cup \{p\}$, $q \leftarrow q + 1$, update H and N^* and go to Step 1.
If $t = t_1$ (partial step) drop constraint k; i.e., set $A \leftarrow A \setminus \{k\}$, $q \leftarrow q - 1$, update H and N^*, and go to Step 2(a).

The heart of the above dual algorithm is the method used to determine a new S-pair $(\bar{x}, \bar{A} \cup \{p\})$ in Step 2 given an S-pair (x, A) and a violated constraint p, i.e.

$$s_p(x) < 0. \tag{3.1}$$

Since (x, A) is an S-pair,

$$s_i(x) = 0 \quad \text{for all} \quad i \in A, \tag{3.2}$$

the columns of N are linearly independent and (2.3) and (2.4) hold.

Let us first consider the case where the columns of N^+ (i.e., those of N and $n^+ = n_p$) are linearly independent. It follows from (2.3), (2.4) and (2.6) that

$$H^+ g(x) = 0 \tag{3.3}$$

and

$$u^+(x) \equiv (N^+)^* g(x) \geq 0. \tag{3.4}$$

As we shall see shortly the following definition is useful.

Definition 1. A triple (x, A, p) consisting of a point x and a set of indices $A^+ = A \cup \{p\}$ where $p \in K \setminus A$ is said to be a V (*violated*)-*triple* if the columns of N^+ are linearly independent and (3.1)–(3.4) hold.

The point $\hat{x}$ corresponding to the V-triple $(\hat{x}, A, p)$ is the optimal solution to the subproblem obtained by replacing the constraint $s_p(x) \geq 0$ in $\mathrm{P}(A^+)$ by

$$s_p(x) \geq s_p(\hat{x});$$

i.e. by a parallel constraint which passes through $\hat{x}$. Let x^* be the point on the manifold $\mathcal{M}^+ = \{x \in \mathbb{R}^n \mid n_i^{\mathrm{T}} x = b_i,\ i \in A^+\}$ at which $f(x)$ is minimized. We now prove a lemma which shows how to move to such a point from a point x corresponding to a V-triple (x, A, p).

Lemma 1. *Let* (x, A, p) *be a V-triple and consider points of the form*

$$\bar{x} = x + tz \tag{3.5}$$

where

$$z = Hn^{+}. \tag{3.6}$$

Then

$$H^{+}g(\bar{x}) = 0, \tag{3.7}$$

$$s_i(\bar{x}) = 0, \quad \textit{for all} \quad i \in A, \tag{3.8}$$

$$u^{+}(\bar{x}) \equiv (N^{+})^{*}g(\bar{x}) = u^{+}(x) + t\binom{-r}{1}, \tag{3.9}$$

where

$$r = N^{*}n^{+}, \tag{3.10}$$

and

$$s_p(\bar{x}) = s_p(x) + tz^{\mathrm{T}}n^{+} \tag{3.11}$$

Proof. All of the above are consequences of (x, A, p) being a V-triple and the facts that

$$g(\bar{x}) = g(x) + tGz,$$

$$Gz = GHn^{+} = (I - NN^{*})n^{+} = n^{+} - Nr = N^{+}\binom{-r}{1},$$

$$H^{+}N^{+} = 0,$$

$$N^{+*}N^{+} = I,$$

and

$$n_i^{\mathrm{T}}Hn^{+} = 0 \quad \text{for all} \quad i \in A.$$

It follows immediately from this lemma that the point $\bar{x} = x + t_2 z$, where $t_2 = -s_p(x)/z^{\mathrm{T}}n^{+}$, minimizes the quadratic (1.1a) over $\mathcal{M}^{+}$. If $u^{+}(\bar{x}) \geq 0$ as well, then $\bar{x}$ is an optimal solution to $\mathrm{P}(A^{+})$ and $(\bar{x}, A^{+})$ is an S-pair. If not, then because of (3.9), there is a smallest value t_1 of t, $t_1 < t_2$, such that some component of $u^{+}(\bar{x}(t)) < 0$ for $t > t_1$. If the constraint, say $k \in A$, corresponding to this component is dropped from the active set, then $(\bar{x}(t_1), A^{-}, p)$, where $A^{-} = A \setminus \{k\}$, is again a V-triple. These remarks are formalized in the following theorems.

Theorem 1. *Given a V-triple* (x, A, p) *if* $\bar{x}$ *is defined by* (3.5) *and* (3.6) *with*

$$t = \min\{t_1, t_2\}, \tag{3.12}$$

where

$$t_1 = \min\left\{ \min_{\substack{r_j > 0 \\ 1 \leq j \leq q}} \left\{ \frac{u_j^{+}(x)}{r_j^{+}(x)} \right\}, \infty \right\}, \tag{3.13}$$

$$t_2 = -s_p(x)/z^{\mathrm{T}}n^{+}, \tag{3.14}$$

and $u^+(x)$ and r are given by (3.9) *and* (3.10), *then*

$$s_p(\bar{x}) \geq s_p(x) \tag{3.15}$$

and

$$f(\bar{x}) - f(x) = tz^{\mathrm{T}}n^+(\tfrac{1}{2}t + u_{j+1}^+(x)) \geq 0. \tag{3.16}$$

Moreover, if $t = t_1 = u_l^+(x)/r_l$, then $(\bar{x}, A \setminus \{k\}, p)$ is a V-triple, where element k in K corresponds to the lth element in A and if $t = t_2$, then $(\bar{x}, A \cup \{p\})$ is an S-pair.

Proof. Since (x, A, p) is a V-triple, we have from (2.6)–(2.8) that

$$z^{\mathrm{T}}n^+ = n^{+\mathrm{T}}Hn^+ = n^{+\mathrm{T}}HGHn^+ = z^{\mathrm{T}}Gz > 0 \tag{3.17}$$

and $t \geq 0$. Hence (3.15) follows from Lemma 1. Also from Taylor's theorem we have that

$$f(\bar{x}) - f(x) = tz^{\mathrm{T}}g(x) + \tfrac{1}{2}t^2z^{\mathrm{T}}Gz. \tag{3.18}$$

Since $H^+g(x) = 0$ implies that $g(x) = N^+u^+(x)$, it follows that $Hg(x) = Hn^+u_{q+1}^+(x)$ and

$$z^{\mathrm{T}}g(x) = n^{+\mathrm{T}}Hg(x) = z^{\mathrm{T}}n^+u_{q+1}^+(x) \geq 0. \tag{3.19}$$

Substituting (3.17) and (3.19) into (3.18) gives (3.16). Moreover as long as $t > 0$, $f(\bar{x}) > f(x)$.

From the definition of t ((3.12)–(3.14)) and Lemma 1 it is evident that $H^+g(\bar{x}) = 0$, $s_i(\bar{x}) = 0$, $i \in A$, and $u^+(\bar{x}) \geq 0$. If $t = t_2$, then $s_p(\bar{x}) = 0$ and $(x, A \cup \{p\})$ is an S-pair. If $t = t_1 < t_2$, then $u_l^+(\bar{x}) = 0$ and $s_p(\bar{x}) < 0$. Since $H^+g(\bar{x}) = 0$ and $u_l^+(\bar{x}) = 0$ we can write

$$g(\bar{x}) = N^+u^+(\bar{x}) = \sum_{i \in A \cup \{p\} \setminus \{k\}} u_{j(i)}^+ n_i,$$

where i is the $j(i)$-th index in A^+. As the set of normals $\{n_i \mid i \in A \cup \{p\} \setminus \{k\}\}$ is clearly linearly independent

$$(\bar{x}, A \setminus \{k\}, p) \quad \text{is a V-triple.}$$

It follows from the above theorem that starting from a V-triple (x, A, p) one can obtain an S-pair $(\bar{x}, \bar{A} \cup \{p\})$ with $\bar{A} \subseteq A$ and $f(\bar{x}) > f(x)$ after $|A| - |\bar{A}| \leq q$ partial steps and one full step.

If n^+ is a linear combination of the columns of N at the start of Step 2, then (x, A, p) is not a V-triple. In this case, either the subproblem $\mathrm{P}(A \cup \{p\})$ is infeasible or a constraint can be dropped from the active set A so that (x, A^-, p) is a V-triple. In the former instance, the original QPP must also be infeasible, while in the latter instance, one may proceed according to Theorem 1 to obtain a new S-pair with a higher function value.

Theorem 2. *Let (x, A) be an S-pair and p be an index of a constraint in $K \setminus A$ such that*

$$n^+ \equiv n_p = Nr \tag{3.20}$$

and

$$s_p(x) < 0. \tag{3.21}$$

If $r \le 0$, then $\mathrm{P}(A \cup \{p\})$ is infeasible; otherwise the kth *constraint can be dropped from the active set, where k is determined by*

$$\frac{u_l(x)}{r_l} = \min_{\substack{r_j > 0 \\ j=1,\dots,q}} \left\{ \frac{u_j(x)}{r_j} \right\}, \quad l = j(k) \tag{3.22}$$

to give $A^- = A \setminus \{k\}$ and the V-triple (x, A^-, p).

Proof. From (3.20) and (3.21) if there is a feasible solution $\bar{x} = x + z$ to $\mathrm{P}(A \cup \{p\})$, it is necessary that

$$n^{+\mathrm{T}} z = r^{\mathrm{T}} N^{\mathrm{T}} z > 0$$

and

$$N^{\mathrm{T}} z \ge 0$$

since $s_i(x) = 0$ for all $i \in A$. But if $r \le 0$, these two requirements cannot be simultaneously satisfied; hence in this case $\mathrm{P}(A \cup \{p\})$ is infeasible.

If some component of r is positive, it follows from (3.22) that $r_l > 0$ and from (3.20) that

$$n_k = \frac{1}{r_l} \left[- \sum_{i \in A^-} r_{j(i)} n_i + n^+ \right]$$

Since (x, A) is an S-pair, we therefore have that

$$\begin{aligned} g(x) &= \sum_{i \in A^-} u_{j(i)} n_i + u_l n_k \\ &= \sum_{i \in A^-} \left(u_{j(i)} - \frac{u_l}{r_l} r_{j(i)} \right) n_i + \frac{u_l}{r_l} n^+ \end{aligned}$$

If we define $\hat{A} = A^- \cup \{p\}$, then it is clear from the above and (3.22) that $\hat{N}$ has full column rank, $\hat{H} g(x) = 0$ and

$$\hat{u}(x) = \hat{N}^* g(x) = \begin{cases} u_{j(i)} - \dfrac{u_l}{r_l} r_{j(i)} \ge 0, & i \in A^-, \\[2ex] \dfrac{u_l}{r_l} \ge 0 \end{cases} \tag{3.23}$$

and hence that (x, A^-, p) is a V-triple.

We note that (3.20) implies that

$$r = N^*n^+.$$

Also if in Step 2 of the dual algorithm the above theorem is applicable, then $u^+ = \binom{u}{0}$ so that $u_j^+ = u_j$, $j = 1, \ldots, q$, since this can only occur before any partial steps have been taken. The change that occurs to the active set A and to the dual variables in Step 2(c(ii)) when $r \not\leq 0$ is the same as the one that occurs in a partial step in Step 2(c(iii)). The only difference is that x is not changed in Step 2(c(ii)) while it usually is changed in Step 2(c(iii)). From the dual point of view, therefore, such a step may thought of as a 'partial' step. Since the dual objective function (of u) is only positive semidefinite when the column rank of C is less than m, it is possible to take nontrivial steps in the space of dual variables without changing x and $f(x)$.

Every time step (1) of the algorithm is executed the current point x solves the subproblem P(A); i.e. (x, A) is an S-pair. If x satisfies all of the constraints in the QPP (1.1), x is an optimal solution. Otherwise after a sequence of at most $q \leq \min(m, n)$ 'partial' steps—the first of which may occur in Step 2(c(ii))—and one full step a new S-pair $(\bar{x}, \bar{A})$ is obtained and we return to Step 1, or infeasibility is detected according to Theorems 1 and 2. Moreover, since $f(\bar{x}) > f(x)$ an S-pair can never reoccur. Since the number of possible S-pairs is finite we have proved:

Theorem 3. *The dual algorithm will solve the* QPP (1.1) *or indicate that it has no feasible solution in a finite number of steps.*

4. Numerically stable implementation

There are many ways to implement the dual algorithm given in Section 3 in a numerically stable manner. Our implementation is based upon the Cholesky factorization

$$G = LL^{\mathrm{T}} \tag{4.1}$$

of the positive definite symmetric Hessian matrix G and the QR factorization

$$B = Q\begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1 \,\vdots\, Q_2]\begin{bmatrix} R \\ 0 \end{bmatrix} \tag{4.2}$$

of the $n \times q$ matrix

$$B = L^{-1}N. \tag{4.3}$$

L is an $(n \times n)$ lower triangular matrix, R is a $(q \times q)$ upper triangular matrix and $Q = [Q_1 \,\vdots\, Q_2]$ is an $(n \times n)$ orthogonal matrix partitioned so that Q_1 has q columns. By substituting (4.1)–(4.3) into (2.1) and (2.2) the operators H and N^*

can be expressed as

$$H = J_2 J_2^{\mathrm{T}} \tag{4.4}$$

and

$$N^* = R^{-1} J_1^{\mathrm{T}} \tag{4.5}$$

where

$$[J_1 \mid J_2] = [L^{-\mathrm{T}} Q_1 \mid L^{-\mathrm{T}} Q_2] = L^{-\mathrm{T}} Q = J. \tag{4.6}$$

Although our implementation is based upon the QR factorization (4.2), we do not store the orthogonal matrix Q; rather we store and update the matrix $J = L^{-\mathrm{T}}Q$ (in addition to L and R) since whenever Q is called for in our algorithm it appears in conjunction with $L^{-\mathrm{T}}$.

In the dual algorithm described in Section 3 the vectors $z = Hn^+$ and $r = N^*n^+$ are required. If we compute

$$d = J^{\mathrm{T}} n^+ = \begin{bmatrix} J_1^{\mathrm{T}} \\ J_2^{\mathrm{T}} \end{bmatrix} n^+ = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}, \tag{4.7}$$

then it follows immediately from (4.4) and (4.5) that

$$z = J_2 d_2 \quad \text{and} \quad r = R^{-1} d_1. \tag{4.8}$$

Updating the factors J and R

Whenever a constraint is added to or deleted from the set of active constraints we update the factors J and R, as computing them ab initio would be much too expensive. Numerically stable methods for updating the QR factorization of a matrix when it is modified by the addition or deletion of a single column are well known (e.g. see Daniel et al. [8], Gill et al. [14] or Goldfarb [17]). Hence our description of the methods used to update J and R will be brief.

In our updating algorithms we make use of Givens matrices Q_{ij} which are equal to the identity matrix with elements (i, i), (j, j), (i, j) and (j, i) replaced respectively, by c, $-c$, s, and s where $c = \cos\theta$ and $s = \sin\theta$ for some value of θ. Because of their special form, computations involving Givens matrices can be illustrated by considering the (2×2) matrix

$$\hat{Q} = \begin{bmatrix} c & s \\ s & -c \end{bmatrix}$$

Since $c^2 + s^2 = 1$, $\hat{Q}$ is orthogonal. In all of our applications $\hat{Q}$ is chosen so as to transform a vector $w = (w_1, w_2)^{\mathrm{T}}$ into the vector $(\omega, 0)^{\mathrm{T}}$ where $\omega = \pm(w_1^2 + w_2^2)^{1/2}$. To accomplish this, we compute

$$\mu = \max\{|w_1|, |w_2|\}, \quad \omega = \operatorname{sign}(w_1)\mu\left[\left(\frac{w_1}{\mu}\right)^2 + \left(\frac{w_2}{\mu}\right)^2\right]^{1/2}$$

$$c = w_1/\omega \quad \text{and} \quad s = w_2/\omega.$$

This scheme, given in Daniel et al. [8], is designed to avoid artificial problems of underflow and overflow caused by squaring very large or very small numbers.

To compute $\hat{y} = \hat{Q}y = \hat{Q}^{\mathrm{T}}y$, $y = (y_1, y_2)^{\mathrm{T}}$ we follow Gill et al. [14] and Daniel et al. [8] and compute

$$\begin{aligned} \nu &= w_2/(\omega + w_1) = s/(1+c), \\ \hat{y}_1 &= cy_1 + sy_2 \end{aligned} \tag{4.9}$$

and

$$\hat{y}_2 = \nu(y_1 + \hat{y}_1) - y_2.$$

When $\hat{Q}$ (or $\hat{Q}^{\mathrm{T}}$) is applied to a $(2\times n)$ matrix, this scheme saves $n-1$ multiplications at the expense of an extra $n+1$ additions over the usual procedure for matrix multiplication. Also, observe that the sign of ω is chosen so that there is no cancellation in the computation of ν.

Adding a constraint

When constraint p, with normal n^+, is added to the active set A, the factorization (4.2) is replaced by

$$B^+ = [B \mid L^{-1}n^+] = Q^+\begin{bmatrix} R^+ \\ 0 \end{bmatrix}. \tag{4.10}$$

From (4.2), (4.6) and (4.7) it follows that

$$Q^{\mathrm{T}}B^+ = \left[\begin{array}{c|c} R & d_1 \\ \hline 0 & d_2 \end{array}\right]. \tag{4.11}$$

Thus the factorization (4.10) can be obtained as

$$Q^+ = Q\left[\begin{array}{c|c} I_q & 0 \\ \hline 0 & \bar{Q}^{\mathrm{T}} \end{array}\right] \quad \text{and} \quad R^+ = \left[\begin{array}{c|c} R & d_1 \\ \hline 0 & \delta \end{array}\right]$$

where $\delta = \|d_2\|$ and $\bar{Q} = Q_{1,2}\cdot Q_{2,3}\cdot \cdots \cdot Q_{n-q-1,n-q}$ is the product of Givens matrices chosen so that

$$\bar{Q}d_2 = \pm\,\delta e_1. \tag{4.12}$$

Moreover,

$$J^+ = L^{-\mathrm{T}}Q^+ = [\underbrace{J_1}_{q} \mid \underbrace{J_2\bar{Q}^{\mathrm{T}}}_{n-q}] = [\underbrace{J_1^+}_{q+1} \mid \underbrace{J_2^+}_{n-q-1}]. \tag{4.13}$$

Note, that for the matrix $\bar{Q}$, one could also choose a single Householder matrix.

Dropping a constraint

Deleting the lth column of N to form N^- when a constraint is dropped from the basis, results in

$$Q_1^{\mathrm{T}}B^- = Q_1^{\mathrm{T}}L^{-1}N^- = \begin{bmatrix} R_1 & S \\ 0 & T \end{bmatrix},$$

where the partitioned matrix on the right equals R with its lth column deleted. If $l \neq q$, T is a $(q-l+1)\times(q-l)$ upper-Hessenberg matrix. Again a sequence of $q-l$ Givens matrices can be chosen so that the product of these matrices $\bar{Q} = Q_{q-l,q-l+1}\cdots Q_{2,3}Q_{1,2}$ reduces T to an upper triangular matrix R_2; i.e.,

$$\bar{Q}T = R_2. \tag{4.14}$$

Thus, we obtain the new factors

$$R^- = \begin{bmatrix} R_1 & S \\ 0 & R_2 \end{bmatrix} \quad\text{and}\quad J^- = J\begin{bmatrix} I_{l-1} & 0 & 0 \\ 0 & \bar{Q}^{\mathrm{T}} & 0 \\ 0 & 0 & I_{n-q} \end{bmatrix}. \tag{4.15}$$

The matrices denoted by $\bar{Q}$ in (4.12) and (4.14) are not computed in our implementation; rather the Givens matrices from which they are formed, which successively introduce zeros into d_2 and T, are applied directly to the rows of J^{T} using the computational scheme (4.9). In [14], Gill et al. show that because of their special structure the matrices $\bar{Q}$ in (4.12) and (4.14) can be generated from two vectors which can be obtained and applied to J^{T} using suitable recurrences. This method, however, is less efficient than the one described above.

The vector $d = J^{\mathrm{T}}n^+$ is required for the step directions z for the primal variables and $-r$ for the dual variables as indicated in (4.8). It is also needed as shown above when the pth constraint, with normal n^+, is added to the active set A. When a constraint is dropped from A, the same orthogonal transformations that are applied to update J^{T} can be applied to d. If this is done, then the updated d can be used to compute the new z and r after the basis change and to determine the appropriate orthogonal transformation $\bar{Q}$ in (4.12) needed when the pth constraint is finally added to the active set. This saves the n^2 operations required to compute d directly.

5. Computational results: Solving randomly generated quadratic programs

In this section we compare the performance of our dual algorithm against that of two different primal algorithms on 168 randomly generated strictly convex quadratic programming problems. Twenty-four different types of problems were generated with known optimal solutions using the technique of Rosen and Suzuki [34]. Each problem type was determined by specifying the number of variables n (9, 27, or 81), the number of constraints m (n or $3n$), the number of constraints q^* in the active set A at the solution ($\frac{1}{9}m$ or $\frac{1}{3}m$), and the conditioning (well or ill) of the Hessian matrix G. For each problem the off-diagonal elements of G were set to $r(-1, 1)$ and $G_{11} = S_1 + r(0, 1) + 1$ was computed, where $r(a, b)$ denotes a freshly computed (pseudo-) random number uniformly distributed between a and b and S_i denotes the sum of the absolute values of the off-diagonal elements in the ith row of G. In the well-conditioned case we set $G_{ii} = S_i + r(0, 1) + 1$, for $i = 2, \dots, n$ while in the ill-conditioned case we set $G_{ii} = G_{i-1} + S_i + S_{i-1} + r(0, 1)$, for $i = 2, \dots, n$. In the latter case the condition members generated were on the order of 50 for $n = 9$, 500 for $n = 27$, and 5000 for $n = 81$. Further, our experiments were subdivided into three runs. In runs 1 and 2 five replications of each of the 16 problem types with n equal to 9 and 27 were generated, and the optimal dual variables u_j, $j \in A$ were set to $r(0, 30)$ and $r(0, 30m)$, respectively. In run 3, one replicate of each of the eight problem types with $n = 81$ was generated, and all u_j, $j \in A$ were set to $(0, 81m)$. To complete the generation of each problem we set the components of the optimal solution x^* to $r(-5, 5)$ and the elements of C to $r(-1, 1)$. The columns of C were then normalized to unit length. For $j \in A$ we set $s_j = 0$ and for $j \not\in A$ we set $s_j = r(0, 1)$ and $u_j = 0$. Then we set $b = s - C^T x^*$ and $a = Cu - Gx^*$.

The results of our computational tests are reported below in Tables 1, 2 and 3. These results differ somewhat from the preliminary results reported in [18] and [24] because of improvements made to the dual and primal codes to make them more efficient. The random number seed used to generate the run 3 problems was changed from the one that was used in [18, 24]. These results are therefore not directly comparable.

The dual algorithm that we used always chose the most violated constraint to add to the active set. The primal algorithms that we used were algorithms 1 and 2 described in Goldfarb [16], the former being essentially identical to the algorithm given by Fletcher [13] in the strictly convex case. For finding a feasible point for the primal algorithms we used the variant of Rosen's [33] procedures suggested in [16], where the operators $N^+ = (N^T N)^{-1}$ and $P = I - NN^+$ used by Rosen are replaced by N^* and H, defined by (2.1) and (2.2).

As already mentioned in the introduction, when this routine is started from the unconstrained minimum the feasible point found by it is often optimal as well; in a certain sense it behaves like a poor version of our dual method which sometimes needs some phase-two primal iteration to 'restore optimality'. For computational results, see Idnani [23] and Dax [9]. Although this choice of starting point is clearly preferred, it would not have enabled us to effectively compare our dual algorithm against the primal algorithms (in phase-two). Therefore, we started the feasible point routine from a randomly generated point (with components set to $r(-5, 5)$) so as to obtain a random feasible point for the start

Table 1
Comparison of primal and dual algorithms—Run 1

Problem type	Problem characteristics					Number of basis changes			No. of operations relative to dual			No. of operations (in 1,000s)
	n	m	q^*	$\mu(G)$	Dual	Feas. pt.	Alg. 1	Alg. 2	Feas. pt.	Alg. 1	Alg. 2	Dual
1	9	9	1	well	5	19	32	24	1.76	3.53	1.85	6
2	9	9	1	ill	5	21	36	30	1.88	3.85	2.16	6
3	9	9	3	well	15	24	17	15	1.09	1.19	0.81	12
4	9	9	3	ill	17(1)	24	29	29	1.01	1.68	1.26	12
5	9	27	3	well	15	52	61	45	1.85	2.88	1.79	16
6	9	27	3	ill	17(1)	49	72	56	1.70	3.75	1.77	17
7	9	27	9	well	61(8)	65	86	78	0.94	1.31	1.27	35
8	9	27	9	ill	55(5)	56	83	89	0.90	1.41	1.56	33
9	27	27	3	well	15	73	98	86	2.34	3.86	1.90	118
10	27	27	3	ill	15	67	90	94	2.22	3.69	2.03	118
11	27	27	9	well	45	67	70	64	1.11	1.46	0.91	237
12	27	27	9	ill	45	74	109	103	1.17	2.16	1.39	237
13	27	81	9	well	47(1)	164	215	145	2.06	3.74	1.54	323
14	27	81	9	ill	47(1)	247	224	178	2.68	4.02	1.86	323
15	27	81	27	well	141(3)	209	292	214	1.15	1.72	1.41	674
16	27	81	27	ill	144(5)	287	202	256	1.40	1.39	1.66	681
Totals/averages:					689(25)	1,498	1,716	1,546	1.58	2.60	1.57	

Table 2
Comparison of primal and dual algorithms—Run 2

Problem type	Problem characteristics					Number of basis changes			No. of operations relative to dual			No. of operations (in 1,000s)
	n	m	q^*	$\mu(G)$	Dual	Feas. pt.	Alg. 1	Alg. 2	Feas. pt.	Alg. 1	Alg. 2	Dual
1	9	9	1	well	5	19	32	28	1.76	3.48	2.03	6
2	9	9	1	ill	5	21	42	38	1.88	4.28	2.70	6
3	9	9	3	well	15	24	17	15	1.09	1.19	0.82	12
4	9	9	3	ill	19(2)	24	33	35	0.96	1.74	1.42	13
5	9	27	3	well	17(1)	52	57	45	1.76	2.55	1.72	17
6	9	27	3	ill	19(2)	49	84	60	1.63	3.55	1.87	17
7	9	27	9	well	113(34)	65	88	78	0.66	0.93	0.90	50
8	9	27	9	ill	77(16)	56	87	85	0.76	1.18	1.23	40
9	27	27	3	well	15	73	106	88	2.34	4.12	2.01	118
10	27	27	3	ill	17(1)	67	98	78	2.12	3.76	1.68	123
11	27	27	9	well	45	67	80	82	1.11	1.62	1.19	237
12	27	27	9	ill	47(1)	74	107	83	1.14	2.03	1.15	242
13	27	81	9	well	73(14)	164	251	189	1.68	3,43	1.74	395
14	27	81	9	ill	73(14)	247	256	164	2.13	3.34	1.43	407
15	27	81	27	well	397(131)	209	288	266	0.63	0.89	0.94	1,227
16	27	81	27	ill	191(28)	287	256	204	1.19	1.23	1.11	805
Totals/averages					1,128(244)	1,498	1,882	1,538	1.43	2.46	1.50	

Table 3
Comparison of primal and dual algorithms—Run 3

Problem type	Problem characteristics				Number of basis changes				No. of operations relative to dual			No. of operations (in 1000s)
	n	m	q^*	$\mu(G)$	Dual	Feas. pt.	Alg. 1	Alg. 2	Feas. pt.	Alg. 1	Alg. 2	Dual
17	81	81	9	well	9	44	67	51	2.48	4.50	1.69	546
18	81	81	9	ill	9	38	67	47	2.26	4.68	1.47	546
19	81	81	27	well	29(1)	32	67	83	0.94	2.15	1.96	1,175
20	81	81	27	ill	31(2)	38	71	59	1.01	2.17	1.28	1,224
21	81	243	27	well	59(6)	129	244	154	1.69	4.33	2.08	2,287
22	81	243	27	ill	33(3)	105	206	112	2.05	5.25	1.87	1,683
23	81	293	81	well	251(85)	133	300	258	0.62	1.43	1.46	6,444
24	81	243	81	ill	108(14)	155	304	220	1.14	2.53	2.11	3,862
Totals/averages					529(111)	674	1,326	984	1.52	3.38	1.74	

of the phase-two computations. Computational results obtained using a simplex-type phase-one algorithm are given in the next section.

Powell [31] recently showed that Rosen's feasible point method can cycle. No cycling, however, was encountered in any of our tests. All of the algorithms used the same matrix factorizations and updating procedures described in Section 4, and were coded modularly in FORTRAN so that they all used many of the same subroutines.* All computations were performed in double precision on an IBM/3033.

On the average our feasible point algorithm and algorithm 2 (the better of the two primal algorithms) each required approximately 50% and 75% more basis changes, respectively, than did the dual algorithm. The total numbers of basis changes required in all runs reported in Tables 1–3 were 3,670 for the feasible point routine and 4,924 for algorithm 1 and 4,068 for algorithm 2 compared with 2,346 for the dual algorithm. Even in those cases least favorable to it, the dual algorithm proved to be superior to the primal algorithms. A total of 1,548 basis changes were required by it versus 1,986 for algorithm 1 and 1,748 for algorithm 2 in phase-two in those runs in which the optimal solution was a vertex. For all runs with the same number of variables n, the average number of constraints active at the optimum was $\frac{4}{9}n$. In fact the dual algorithm required fewer basis changes than both the feasible point algorithm and algorithms 1 and 2 in all cases except for problem types 7, 8, and 15 in run 2 and problem type 23 in run 3. These cases were the most difficult for the dual to solve. They all had their optimal solutions at a vertex, large optimal dual variables and well conditioned Hessians (except for problem type 8); hence the distances $\|G^{-1}Nu\|$ between the unconstrained and constrained optimal solutions were very large. Clearly, the less constraints that are active at the optimal solution and the closer this solution is to the unconstrained minimum the more favorable it is for the dual algorithm in comparison to primal algorithms.

One reason for the superior performance of the dual algorithm is that it appears not to add many constraints to the active set that are not in the final basis. The figures in parentheses in Tables 1, 2, and 3 are the number of times the dual algorithm had to drop a constraint from the basis during the solution of all problems of that type. In 110 of the 168 problems run no constraint had to be dropped. (The number of basis changes required by the dual algorithm equals the number of replications—five in runs 1 and 2 and one in run 3—times q^* when no drops occur for that problem type.) Also approximately two-thirds of all drops occurred in the problems where the optimal solution occurred at a vertex and was far from the unconstrained minimum. This is not surprising since under such conditions it would be quite likely for the dual algorithm to add a 'wrong' constraint to the basis which it would subsequently have to drop.

Our feasible point routine produced an initial feasible point with ap-

* These codes and the random problem generator are available from the authors.

proximately $\min(n, \frac{1}{2}m)$ constraints active on the average. (It almost invariably found a vertex when m was equal to $3n$.) Of these initial constraints one-half were, on the average, also in the final optimal basis. This is to be expected since the feasible point routine was started from a random point and used no information about the objective function in its computations. On all problems with $m = n$, the feasible point routine only added constraints to the basis. Even so, it was inferior to the dual algorithm because it added too many constraints—approximately $\frac{1}{2}m$ versus $\frac{1}{3}m$ or $\frac{1}{9}m$.

In Tables 1–3 we also give the numbers of operations required by the primal algorithms relative to the dual. These were obtained by computing the ratio of the number of operations (multiplications + divisions + 10*square roots) required by each method to that required by the dual method for all replications of a particular problem type. Averages of these ratios over all problem types are given at the bottom of each table.

The relative performance of the algorithms in terms of operation counts was quite similar to that in terms of basis changes. For instance, the comparisons were least favorable to the dual algorithm for problem types 7 and 15 in run 2 and 23 in run 3. Also, in no case did either of the combined feasible point-primal algorithms require less than approximately one and one-half as many operations as did the dual algorithm. On the average the algorithm 2 combination (the better of the two) required approximately three times as much computation and basis changes as the dual algorithm.

As operation counts are dependent upon an algorithm's implementation we tried to implement all algorithms in the same way. Some differences, however, should be noted. In the dual algorithm we computed the step direction in the primal space as $z = Hn^+$ as indicated in (4.7) and (4.8), while in the primal algorithms the step direction was computed as $z = -Hg$. Since the vector $d = J^+n$ (subsequently transformed if constraints are dropped from the active set) is needed for updating the factors J and R when a constraint is added to the active set, this saved some operations in the dual algorithm. Further, it meant that the gradient $g(x)$ was never needed by the dual algorithm. If one always starts either of the primal algorithms at a constrained stationary point then they can be implemented so as to incorporate similar efficiencies. This was not done because our primal algorithms were also used within primal–dual algorithms [18, 24] which called them at nonstationary points. In any case these changes to the primal codes would not have significantly affected our results. Perhaps a more important factor is that the inner products necessary for computing the slacks for all inactive constraints are computed in the outer iteration in the dual algorithm to determine the constraint to add to the active set, whereas they are computed in the inner iteration in primal algorithms to determine the step length. Moreover, if the quantities $n_i^T z$, $i \in K \setminus A$ are not stored, then two inner products per inactive constraint are required in the primal algorithms.

6. Computational results: Solving quadratic programs generated by a successive quadratic programming code

According to recent studies successive quadratic programming algorithms, and in particular an algorithm developed by Powell [30] which is based upon earlier work by Biggs [4] and Han [21], are among the best methods currently available for solving general nonlinear programming problems. These methods search in directions obtained by solving a strictly convex quadratic programming problem at each iteration. Consequently, they require efficient algorithms for solving such quadratic programs if they are themselves to be efficient.

In this section we present the computational results obtained when we used our dual algorithm to solve the sequences of quadratic programs generated by Powell's algorithm in the course of its solution of six nonlinear programming test problems. We also compare the performance of the dual algorithm with that of the feasible point-primal quadratic programming code [11, 12] due to Fletcher that is normally used by Powell's code VFO2AD. Fletcher's feasible point routine determines a feasible vertex and is more closely related to a standard phase 1 simplex algorithm than it is to the feasible point routine described in the previous section. Fletcher's primal quadratic programming algorithm [13] is identical to algorithm 1 of the last section when applied to a strictly convex QPP except for its implementation; e.g., the matrices H and N^* rather than factorizations are stored and updated.

The quadratic programs that have to be solved at each iteration of Powell's algorithm to determine a direction of search d are of the form

$$\begin{aligned} &\text{minimize} && \nabla f^{\mathrm{T}} d + \tfrac{1}{2} d^{\mathrm{T}} B d + h(\xi), && (6.1\mathrm{a})\\ &\text{subject to} && \nabla c_i^{\mathrm{T}} d + c_i \xi_i = 0, \quad i = 1, \dots, k', && (6.1\mathrm{b})\\ &&& \nabla c_i^{\mathrm{T}} d + c_i \xi_i \geq 0, \quad i = k'+1, \dots, k, &&\\ &&& 0 \leq \xi \leq 1 && \end{aligned}$$

where ξ_i has the value

$$\xi_i = \begin{cases} 1, & c_i > 0, \\ \xi, & c_i \leq 0. \end{cases}$$

∇f and ∇c_i, $i = 1, \dots, k$ are the gradients of the nonlinear programming problem's objective function $f(x)$ and the k' equality and $k - k'$ inequality constraints $c_i(x) = 0$, and $c_i(x) \geq 0$ evaluated at x, the current estimate of the solution. B is a positive definite symmetric quasi-Newton approximation to the Hessian of the Lagrangian of the nonlinear programming problem with respect to the x variables and $h(\xi)$ is a scalar function of ξ chosen so as to make ξ as large as possible subject to the constraints (6.1b). Since the dual method requires a strictly convex objective function we used $h(\xi) = 10^6(\frac{1}{2}\xi^2 - 2\xi)$ rather than $h(\xi) = -10^6\xi$ as in Powell's original code. Also it was not necessary to add artificial lower and upper bounds to each quadratic program as required by Fletcher's feasible point code. In spite of these differences in implementation both quadratic programming codes resulted in essentially identical nonlinear programming iterations.

The results of our computational tests are summarized in Table 4. The numbers of variables and constraints for the generated quadratic program (given in parentheses in Table 4) are respectively one and two more than these numbers for the original nonlinear problem because of the variable ξ and the bounds on it. To obtain the results in this table the number of basis changes and operations were summed over all quadratic programming problems generated during the solution of a given nonlinear problem. Observe that although now on the average the feasible point routine requires fewer basis changes than the dual, the total for it and Fletcher's primal algorithm is more than 30% more than that required by the dual algorithm. This is in spite of the fact that the optimal solutions of all of the quadratic programs solved were on manifolds of dimension less than or equal to two. In Colville 3 all optimal solutions occurred at vertices (see the columns with headings 'Variables' and 'Average q^*').

The set of constraints that are active at the optimal solution of the quadratic program (6.1) tends not to change very much from one iteration of Powell's algorithm to the next. Note that even if the active set remains the same the constraints in that set may change because the linearization is carried out at a new point. In four of the six problems run the optimal basis stayed the same for all iterations while in the other two problems the overlap was, on the average, greater than 90%. To take advantage of this we modified the dual algorithm so that our algorithm was executed in two passes on all iterations of Powell's algorithm other than the first. In the first pass all constraints not in the optimal active set A^* on the previous iteration of Powell's algorithm were ignored. In addition in Step 1 p was chosen as the first violated constraint (i.e., the one with the lowest index) rather than the most violated constraint. In the second pass the full set of constraints was considered and the dual algorithm was started from the optimal solution of $P(A^*)$ obtained by the first pass. This resulted in fewer drops and an overall computational savings of approximately 20%. The results presented in Table 4 are for this variant of the dual algorithm. A consequence of this strategy was that the dual algorithm rarely had to drop constraints; in 49 QPP's solved only a total of 23 drops were required. The figures in parentheses next to the number of basis changes required by the dual (Fletcher's primal) algorithm give the number of times a constraint had to be dropped from (added to) the basis.

The results reported for Fletcher's code are for a variant that included in the initial infeasible basis at the start of the feasible point routine those constraints

Table 4
Comparison of algorithms when used as subroutines in Powell's successive quadratic programming code

Problem	Number of				No. of basis changes			No. of operations *relative to dual*				No. of operns. for the dual
	Vari-ables	Con-straints	QPPs	Avg. q^*	Feas. pt.	Primal	Dual	Feas. pt.	Primal	Tot.	Cholesky	
Powell	5(6)	3(5)	6	4.0	4	12	24	0.47	1.59	2.06	0.14	5,658
POP	3(4)	7(9)	6	2.2	30	15(2)	13	1.49	2.32	3.81	0.25	1,556
Triangle	7(8)	9(11)	13	6.1	66	67(21)	83(2)	0.89	2.82	3.71	0.11	26,103
Colville 1	5(6)	15(17)	5	5.2	28	10(3)	34(4)	0.93	1.89	2.82	0.10	6,608
Colville 2	15(16)	20(22)	15	14.0	124	203(86)	239(15)	0.74	2.89	3.63	0.07	240,615
Colville 3	5(6)	16(18)	4	6.0	16	6(3)	28(2)	0.66	1.77	2.43	0.10	5,078
Totals/averages			49		268	313(115)	421(23)	0.86	2.21	3.07	0.13	

that were in the previous optimal basis as long as they were still sufficiently linearly independent.

The number of operations relative to the dual reported in Table 4 were calculated as described in the last section. However, we did not include the work required to factor B in counting the number of operations for the dual algorithm because nonlinear programming codes which are based upon variable metric formulas should update the Cholesky factors of B rather than B itself. The extra work for these factorizations was computed and is given in the table relative to the work required for the dual algorithm. The number of operations required by Fletcher's primal code relative to the dual was much greater than one would expect from the difference in the numbers of basis changes required by the two algorithms. This shows that Fletcher's implementation of algorithm 1 is rather inefficient.

Our computational results also show that using our dual algorithm to solve the QPP's that were generated by Powell's successive quadratic programming code took between four and fourteen times as much work as factoring the Hessian, and on the average only about seven and one-half times as much work.

The results presented in Table 4 differ from preliminary results presented in [18] and [24] as a result of some changes to the dual code and because the earlier results did not account for the situations in which Fletcher's code exchanged one constraint in the basis for another not in the basis.

It should also be mentioned that Fletcher's code experienced some numerical problems in the course of solving the Colville 2 problem which caused the system to print several underflow error messages (on an IBM 3033). The dual code on the other hand experienced no numerical difficulties.

7. Relationship to other dual methods

It is easy to verify that our original QPP (1.1) and the problem

$$\begin{aligned} &\text{maximize} && b^{\mathrm{T}}u - \tfrac{1}{2}x^{\mathrm{T}}Gx, && (7.1\text{a})\\ &\text{subject to} && Gx - Cu = -a && (7.1\text{b})\\ &\text{and} && u \ge 0 && (7.1\text{c}) \end{aligned}$$

dual to (1.1) both must satisfy the Kuhn–Tucker necessary conditions (1.1b), (7.1b), (7.1c) and $u^{\mathrm{T}}s = 0$ at their respective optimal solutions. In [39] Van de Panne and Whinston apply a primal simplex method, first proposed by Dantzig [10] for solving convex quadratic programs, to the dual of a QPP to obtain a dual method. In the positive definite case Dantzig's primal method is mathematically, but not computationally, equivalent to the projection methods of Fletcher [13] and Goldfarb [16] as shown in [16].

We shall now briefly describe this dual method as applied to problem (1.1) and show that it follows the same solution path as our dual algorithm. Consequently, our algorithm is mathematically equivalent to applying either Dantzig's, Fletcher's, or Goldfarb's primal algorithm to the dual of the given problem.

As in Dantzig's primal method the computations performed in the Van de Panne–Whinston dual method are based upon using simplex pivots applied to tableaux which can be obtained from the tableau

x	$s \geq 0$	$u \geq 0$	
G	0	$-C$	$-a$
C^{T}	$-I$	0	b

A major iteration begins with what is called a 'standard' tableau; i.e., one whose corresponding solution is dual feasible and which has no basic (or nonbasic) pairs. A basic (nonbasic) pair means that both u_i and s_i are basic (nonbasic). Such a solution x is also referred to in the literature as 'complementary'. In our terminology x is the solution of the subproblem P(A), where $A = \{i \in K \mid s_i \text{ is nonbasic}\}$; i.e., (x, A) is an S-pair. The method chooses the dual variable u_p corresponding to a negative slack s_p to increase and add to the basis. This corresponds to step 1 of our algorithm. If s_p is reduced to zero before any basic dual variable, then it is removed from the basis, a simplex pivot is performed, and once again the solution is 'complementary'. This corresponds to a full step ($t = t_2$) in our method. Otherwise one of the dual variables, say u_k, goes to zero and is dropped from the basis yielding a 'noncomplementary' solution (the tableau is called 'nonstandard'), with a basic pair (u_p, s_p) and a nonbasic pair (u_k, s_k). In our method this corresponds to a partial step ($t = t_1$). Next the slack s_k is increased and added to the basis. This corresponds to dropping constraint k from the active set A in our method and can be shown to have the effect of also increasing s_p. Again either s_p will be reduced to zero before any basic dual variable and it will be dropped from the basis to yield a complementary solution, or one of the dual variables (other than u_p, as it will increase) will be dropped from the basis, again leaving a noncomplementary solution with one basic and one nonbasic pair. In the latter case, the above procedure for a noncomplementary solution is repeated. Thus, starting from a standard tableau (S-pair) one proceeds through a sequence of nonstandard tableaus (partial steps) until one again obtains a standard tableau (full step).

A comparison between the actual computations performed by our dual algorithm and the Van de Panne–Whinston algorithm is given in [24]. It is shown there that our algorithm is far more efficient. It is also more numerically stable. The only noncomputational difference between the two algorithms is that the latter does not allow for the possibility of primal infeasibility. This shortcoming is easily rectified by terminating with an indication that (1.1b) is infeasible when Rule 2 in [39] cannot be applied.

Another dual method that is related to our method is the one proposed by Lemke [26]. Lemke first eliminates the primal variables from the dual QPP (7.1) using (7.1b)—i.e.,

$$x = G^{-1}(Cu - a)$$

to obtain the following dual problem:

$$\underset{u \geq 0}{\text{minimize}} \quad w(u) = -s_0^{\mathrm{T}}u - \tfrac{1}{2}u^{\mathrm{T}}Qu \tag{7.2}$$

where $s_0 = C^{\mathrm{T}}x_0 - b$, $x_0 = -G^{-1}a$, and $Q = C^{\mathrm{T}}G^{-1}C$. Then starting from the dual feasible point $u = 0$ ($x = x_0$, the unconstrained minimum of $f(x)$) Lemke essentially applies Beale's conjugate direction approach [2, 3] for solving a QPP to (7.2).

Specifically, Lemke's method employs the 'explicit inverse' of a basis matrix B which has columns of the identity matrix corresponding to the normals of all active dual constraints and columns of the form Qd_i where d_i is the direction of a dual step previously taken. Starting from $u = 0$ and $B = I$, if some component, say the pth, of the gradient of $w(u)$, $y = -(s_0 + Qu)$ (corresponding to a zero dual variable) is negative ($s_p(x) < 0$ in the primal QPP), Lemke's algorithm drops u_p from the 'dual' active set and moves in the direction $-d_p$, where d_p is the pth row of B^{-1}. If the minimum of $w(u)$ along the semi-infinite ray $\{\bar{u} = u - \tau d_p \mid \tau \geq 0\}$ satisfies the dual constraints $u \geq 0$, then a step to this point is taken, the above procedure is repeated and the solution path, in both x and u variables, is the same as the one followed by our dual algorithm. The sequence of points u so obtained are the minima of $w(u)$ on faces of the nonnegative orthant of increasing dimension. The corresponding points x are the solutions of subproblems P(A), where A is of increasing cardinality. However, if a dual constraint is encountered before a minimum is reached along $-d_p$, the two algorithms in general follow different solution paths from that point on. On the next iteration our algorithm will proceed in the direction of the minimum of $w(u)$ on the new constraint manifold in the dual space, whereas Lemke's algorithm will take as many as k Q-conjugate steps, where k is the dimension of the manifold, to reach this minimum. If this minimum is reached by both algorithms, they will again begin to follow the same solution path. On the other hand, if some dual constraint is encountered by either algorithm before this minimum is reached, then the two algorithms can subsequently proceed through totally different dual and primal active sets. (See the discussion of Beale's algorithm in [16] for a more complete analysis.)

Different solution paths will result if problem (1.1) with

$$G = \tfrac{1}{3}\begin{bmatrix} 19 & 4 & -8 \\ 4 & 4 & -2 \\ -8 & -2 & 4 \end{bmatrix}, \quad a = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}, \quad \text{and} \quad C = I_3,$$

is solved choosing successively the first, second and third constraints to add to the active set in our algorithm while making corresponding choices of the dual variable to drop from the dual active set in Lemke's algorithm. On the other hand, both algorithms follow the same solution path on the example given in the appendix with the first set of active set changes specified there.

Another method which invites comparison with ours is one proposed by Bartels et al. in Section 12 of [1]. That method also starts from the unconstrained minimum of (1.1a) and uses the factorizations (4.1) and (4.2), updating Q and R rather than J and R. However, instead of following a purely dual approach it uses the principal pivoting method [7].

8. Observations

In our computational tests the dual method was found to require fewer basis changes than the primal methods with which it was compared. Without any doubt the principal reason for this was that no phase-one was needed to find a dual feasible point. For this reason, one would expect primal methods which use an l_1 penalty term (e.g., see [6]) to penalize constraint violations rather than a separate phase-one procedure to also require less basis changes than the primal methods that we tested.

There are modifications that can be made to the dual algorithm to make it more efficient even though the number of basis changes may increase. At any iteration, arbitrary subsets of constraints can be temporarily ignored when computing slacks and seeking a violated constraint in Step 1. One such strategy was described in Section 6, and many others are possible. When the number of constraints is large, the potential for reducing the cost per iteration is sufficient to make such strategies worthwhile. Clearly this is analogous to 'partial pricing' in the simplex method for linear programming.

It is important to point out that there are many numerically stable ways to implement the dual algorithm. Several are described in [17]. Although the discussion there is in terms of primal algorithms the factorizations described are also applicable to our dual algorithm. The implementation that we chose is not designed to take advantage of simple upper and lower bounds on the variables or general sparsity. If one wishes to do this or to extend the algorithm so that it can solve problems with positive semi-definite and indefinite Hessians, then an implementation based upon the Cholesky factorization of $Z^{\mathrm{T}}GZ$, where the columns of Z form a basis for the null space of N, would appear to be more appropriate. (It should be noted that Van de Panne and Whinston's dual method can solve semidefinite quadratic programs.) If in addition the constraints are given in the standard linear programming form (i.e. $Ax = b$, $l \leq x \leq u$), then an implementation which also uses an LU factorization of the basis matrix as in [29] might be best.

Our final remarks concern the problems of degeneracy and 'near' degeneracy. For reasons of numerical stability it is important in both primal and dual algorithms to avoid intermediate active sets which are nearly linearly dependent. In dual algorithms one usually has a choice of which constraint to add to the active set, while in primal algorithms that selection is determined by feasibility conditions except in exactly degenerate cases. (The situation is reversed with regard to dropping constraints from the active set.) Consequently, one has greater control over the linear independence of the active set in dual methods.

As far as degeneracy is concerned no special procedure, such as small perturbations of the constraints, is required to prevent 'cycling' in our method as is the case for primal methods. This advantage is also claimed by Lemke [26] for his method.

Because of computer rounding errors a degenerate situation may in practice appear to be nearly degenerate. Consider the behavior of a reasonable implementation of our algorithm in such a situation. If the current iterate x is 'nearly' feasible, then the algorithm will stop since all violations of the 'nearly dependent' inactive constraints should fall within a prescribed tolerance for termination. If x is so infeasible that the algorithm does not terminate, then we expect that there is a constraint violated by x which, when added to the active set, will take us away from the current near degenerate situation and will avoid an ill-conditioned intermediate active set constraint matrix. Choosing the most violated constraint is, in fact, a good strategy for doing this, In contrast, primal methods cannot choose which constraint to add to the active set (unless a small amount of infeasibility is tolerated). Therefore ill-conditioned intermediate states are more likely to occur, as are instances in which many small steps are taken in essentially the same manifold as a result of interchanging linearly dependent or nearly linearly dependent active constraints.

On the other hand an ill-conditioned Hessian matrix G might be expected to cause more numerical problems for a dual method than for a primal method since the former starts from the unconstrained minimum $x^0 = -G^{-1}a$.

Acknowledgement

Thanks are due to the associate editor and referee for their useful suggestions.

Appendix: An example

To illustrate the various parts of the dual algorithm, we apply it to the following problem.

Problem:

$$
\begin{aligned}
\text{Minimize} \quad & f(x) = 6x_1 + 2(x_1^2 - x_1x_2 + x_2^2), \\
\text{subject to} \quad & x_1 \geq 0, \\
& x_2 \geq 0, \\
& x_1 + x_2 \geq 2.
\end{aligned}
$$

This problem is depicted in Fig. 1 along with all possible solution paths. More than one path is possible because one has the freedom of choosing any violated constraint in Step 1 of the algorithm. Since

$$G^{-1} = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix}^{-1} = \frac{1}{6}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \quad \text{and} \quad a = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$$

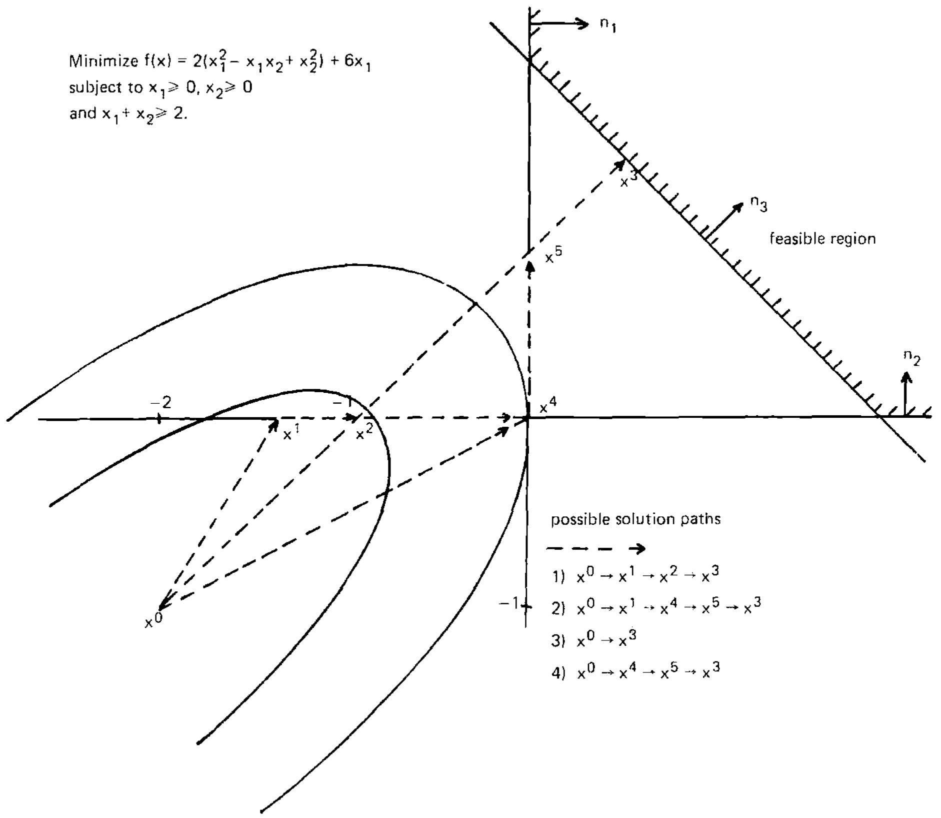

Fig. 1. An example showing all possible solution paths for the dual algorithm where any violated constraint can be chosen for *p* in Step 1.

Table A.1

Iteration	At start of iteration					Computed during iteration							Remarks
	x	s	f	A	u	p	z	r	t_1	t_2	k	u^+	
Solution path $x^0 \to x^1 \to x^2 \to x^3$													
1	$\begin{bmatrix} -2 \\ -1 \end{bmatrix}^*$	$\begin{bmatrix} -2 \\ -1 \\ -5 \end{bmatrix}$	-6	$\emptyset^*$	—	2	$\begin{bmatrix} \frac{1}{6} \\ \frac{1}{3} \end{bmatrix}$	—	∞	3	—		full step; add constraint 2
2	$\begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}^*$	$\begin{bmatrix} -\frac{3}{2} \\ 0 \\ -\frac{7}{2} \end{bmatrix}$	$-\frac{9}{2}$	$\{2\}^*$	(3)	3	$\begin{bmatrix} \frac{1}{4} \\ 0 \end{bmatrix}$	$(\frac{3}{2})$	2	14	2		partial step; drop constraint 2
3	$\begin{bmatrix} -1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -1 \\ 0 \\ -3 \end{bmatrix}$	-4	$\emptyset$	—	3	$\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$	—	∞	3	—		full step; add constraint 3
4	$\begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \end{bmatrix}^*$	$\begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \\ 0 \end{bmatrix}$	$\frac{13}{2}$	$\{3\}^*$	5								stop; all constraints satisfied
Solution path $x^0 \to x^1 \to x^4 \to x^5 \to x^3$													
2′	$\begin{bmatrix} -\frac{3}{2} \\ 0 \end{bmatrix}^*$	$\begin{bmatrix} -\frac{3}{2} \\ 0 \\ -\frac{7}{2} \end{bmatrix}$	$-\frac{9}{2}$	$\{2\}^*$	(3)	1	$\begin{bmatrix} \frac{1}{4} \\ 0 \end{bmatrix}$	$(\frac{1}{2})$	6	6	2		full step; add constraint 1
3′	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}^*$	$\begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix}$	0	$\{1, 2\}^*$	$\binom{0}{6}$	3	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$	0	∞	2		dual step; drop constraint 2
4′	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}^*$	$\begin{bmatrix} 0 \\ 0 \\ -2 \end{bmatrix}$	0	$\{1\}^*$	(6)	3	$\begin{bmatrix} 0 \\ \frac{1}{4} \end{bmatrix}$	$(\frac{3}{2})$	4	8	1		partial step; drop constraint 1
5′	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$	2	$\emptyset$	—	3	$\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}$	—	∞	1	—		full step; add constraint 3
6′	$\begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \end{bmatrix}^*$	$\begin{bmatrix} \frac{1}{2} \\ \frac{3}{2} \\ 0 \end{bmatrix}$	$\frac{13}{2}$	$\{3\}^*$	5								stop; all constraints satisfied

the unconstrained minimum occurs at $x^0 = (-2, -1)^T$ with function value -6; hence we begin with the S-pair $(x^0, \emptyset)$, $H = G^{-1}$, $A = \emptyset$, and $q = 0$.

The computations for the two solution paths labeled $(x^0 \to x^1 \to x^2 \to x^3)$ and $(x^0 \to x^1 \to x^4 \to x^5 \to x^3)$ in Fig. 1 are summarized in Table A.1 below. These two paths diverge during the second iteration because in the first case the most violated constraint $p = 3$ is chosen to add to the active set while in the second case $p = 1$ is chosen. Observe that the most violated constraint is also not chosen in iteration 1. The second set of iterations is given to illustrate Step 2(c(ii))—the purely dual step—of the algorithm. Since $t_1 = 0$ in this illustration, the dual variables do not change. However, it is easy to construct examples in which $|z| = 0$ and $t_1 > 0$ and a nonzero step is taken in the dual space while no step is taken in the primal space.

References

[1] R.H. Bartels, G.H. Golub, and M.A. Saunders, "Numerical techniques in mathematical programming", in: J.B. Rosen, O.I. Mangasarian and K. Ritter, eds., *Nonlinear programming* (Academic Press, New York, 1970) pp. 123–176.

[2] E.M.L. Beale, "On minimizing a convex function subject to linear inequalities", *Journal of the Royal Statistical Society Series B* 17 (1955) 173–184.

[3] E.M.L. Beale, "On quadratic programming", *Naval Research Logistics Quarterly* 6 (1959) 227–243.

[4] M.C. Biggs, "Constrained minimization using recursive quadratic programming: some alternative subproblem formulations", in: L.C.W. Dixon and G.P. Szego, eds., *Towards global optimization* (North-Holland, Amsterdam, 1975) pp. 341–349.

[5] J.W. Bunch and L. Kaufman, "Indefinite quadratic programming", Computing Science Technical Report 61, Bell. Labs., Murray Hill, NJ (1977).

[6] A.R. Conn and J.W. Sinclair, "Quadratic programming via a nondifferentiable penalty function", Department of Combinatorics and Optimization Research Report CORR 75-15, University of Waterloo, Waterloo, Ont. (1975).

[7] R.W. Cottle and G.B. Dantzig, "Complementary pivot theory of mathematical programming", in: G.B. Dantzig and A.F. Veinott, eds., *Lectures in applied mathematics II, Mathematics of the decision sciences, Part I* (American Mathematical Society, Providence, RI, 1968) pp. 115–136.

[8] J.W. Daniel, W.B. Graggs, L. Kaufman and G.W. Stewart, "Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorizations", *Mathematics of Computation* 30 (1976) 772–795.

[9] A. Dax, "The gradient projection method for quadratic programming", Institute of Mathematics Report, The Hebrew University of Jerusalem (Jerusalem, 1978).

[10] G.B. Dantzig. *Linear programming and extensions* (Princeton University Press, Princeton, NJ, 1963) Chapter 24, Section 4.

[11] R. Fletcher, "The calculation of feasible points for linearly constrained optimization problems", UKAEA Research Group Report, AERE R6354 (1970).

[12] R. Fletcher, "A FORTRAN subroutine for quadratic programming", UKAEA Research Group Report, AERE R6370 (1970).

[13] R. Fletcher, "A general quadratic programming algorithm", *Journal of the Institute of Mathematics and Its Applications* (1971) 76–91.

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

[15] P.E. Gill and W. Murray, "Numerically stable methods for quadratic programming", *Mathematical programming* 14 (1978) 349–372.

[16] D. Goldfarb, "Extension of Newton's method and simplex methods for solving quadratic programs", in: F.A. Lootsma, ed., *Numerical methods for nonlinear optimization* (Academic Press, London, 1972) pp. 239–254.

[17] D. Goldfarb, "Matrix factorizations in optimization of nonlinear functions subject to linear constraints", *Mathematical Programming* 10 (1975) 1–31.

[18] D. Goldfarb and A. Idnani, "Dual and primal–dual methods for solving strictly convex quadratic programs", in: J.P. Hennart, ed., *Numerical Analysis, Proceedings Cocoyoc, Mexico* 1981, Lecture Notes in Mathematics 909 (Springer-Verlag, Berlin, 1982) pp. 226–239.

[19] A.S. Goncalves, "A primal–dual method for quadratic programming with bounded variables", in F.A. Lootsma, ed., *Numerical methods for nonlinear optimization* (Academic Press, London, 1972) pp. 255–263.

[20] M.D. Grigoriadis and K. Ritter, "A parametric method for semidefinite quadratic programs", *SIAM Journal of Control* 7 (1969) 559–577.

[21] S-P. Han, "Superlinearly convergent variable metric algorithms for general nonlinear programming problems", *Mathematical Programming* 11 (1976) 263–282.

[22] S-P. Han, "Solving quadratic programs by an exact penalty function", MRC Technical Summary Report No. 2180, M.R.C., University of Wisconsin (Madison, WI, 1981).

[23] A.U. Idnani, "Extension of Newton's method for solving positive definite quadratic programs—A computational experience", Master's Thesis, City College of New York, Department of Computer Science (New York, 1973).

[24] A.U. Idnani, "Numerically stable dual projection methods for solving positive definite quadratic programs", Ph.D. Thesis, City College of New York, Department of Computer Science (New York, 1980).

[25] C.L. Lawson and R.J. Hanson, *Solving least squares problems* (Prentice-Hall, Engelwood Cliffs, N.J., 1974).

[26] C.E. Lemke, "A method of solution for quadratic programs", *Management Science* 8 (1962) 442–453.

[27] R. Mifflin, "A stable method for solving certain constrained least squares problems", *Mathematical Programming* 16 (1979) 141–158.

[28] W. Murray, "An algorithm for finding a local minimum of an indefinite quadratic program", NPL NAC Report No. 1 (1971).

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

[30] M.J.D. Powell, "A fast algorithm for nonlinearly constrained optimization calculations", in: *Numerical analysis, Dundee*, 1977, Lecture Notes in Mathematics 630 (Springer Verlag, Berlin, 1978) pp. 144–157.

[31] M.J.D. Powell, "An example of cycling in a feasible point algorithm", *Mathematical Programming* 20 (1981) 353–357.

[32] K. Ritter, "Ein Verfahren zur Lösung parameter-abhängiger, nichtlinearer Maximum-Probleme", *Unternehmensforschung* 6 (1962) 149–166; English Transl., *Naval Research Logistics Quarterly* 14 (1967) 147–162.

[33] J.B. Rosen, "The gradient projection method for nonlinear programming, Part 1. Linear constraints", *SIAM Journal of Applied Mathematics* 8 (1960) 181–217.

[34] J.B. Rosen and S. Suzuki, "Construction of nonlinear programming test problems", *Communications of the ACM* (1965) 113.

[35] K. Schittkowski, *Nonlinear programming codes—Information, tests, performance*, Lecture Notes in Economics and Mathematical Systems, No. 183 (Springer-Verlag, Berlin, 1980).

[36] K. Schittkowski and J. Stoer, "A factorization method for the solution of constrained linear least squares problems allowing subsequent data changes", *Numerische Mathematik* 31 (1979) 431–463.

[37] J. Stoer, "On the numerical solution of constrained least squares problems", *SIAM Journal on Numerical Analysis* 8 (1971) 382–411.

[38] H. Theil and C. Van De Panne, "Quadratic programming as an extension of conventional quadratic maximization", *Management Science* 7 (1960) 1–20.
[39] C. Van de Panne and A. Whinston, "The simplex and the dual method for quadratic programming", *Operations Research Quarterly* 15 (1964) 355–389.
[40] R.B. Wilson, "A simplicial algorithm for concave programming", Dissertation, Garduate School of Business Administration, Harvard University (Boston, MA, 1963).
[41] P. Wolfe, "The simplex method for quadratic programming", *Econometrica* 27 (1959) 382–398.