NONMONOTONE STRATEGY FOR MINIMIZATION OF QUADRATICS WITH SIMPLE CONSTRAINTS*

M. A. Diniz-Ehrhardt, Campinas, Z. Dostál, Ostrava,
M. A. Gomes-Ruggiero, J. M. Martínez and S. A. Santos, Campinas

(Received December 8, 1998, in revised version July 30, 1999)

Abstract. An algorithm for quadratic minimization with simple bounds is introduced, combining, as many well-known methods do, active set strategies and projection steps. The novelty is that here the criterion for acceptance of a projected trial point is weaker than the usual ones, which are based on monotone decrease of the objective function. It is proved that convergence follows as in the monotone case. Numerical experiments with bound-constrained quadratic problems from `CUTE` collection show that the modified method is in practice slightly more efficient than its monotone counterpart and has a performance superior to the well-known code `LANCELOT` for this class of problems.

Keywords: quadratic programming, conjugate gradients, active set methods

MSC 2000: 65K10, 49M07, 65F15, 90C20

1. Introduction

The problem of minimizing a quadratic function f subject to bounds on the variables has many practical applications. Many times, physical and engineering problems can be modelled as box-constrained quadratic minimization problems with a large number of variables (see, for example, [13], [14], [15], [25], [26]). On the other hand, quadratic problems with bounds appear as *subproblems* in the context of methods for minimizing arbitrary functions with nonlinear constraints. See, for example, [6], [7], [20], [29]. For these reasons, a lot of algorithms have been developed with

* This work was supported by Grant No. 201/97/0421 of the Grant Agency of the Czech Republic, FAPESP grants 90-3724-6, 95-6498-8 and 97-12676-4, FINEP, CNPq and FAEP-UNICAMP.

the aim of solving this problem efficiently. See [2], [5], [8], [11], [12], [16], [18], [19], [24], [27], [28], [31] and references therein. Some of these algorithms [2], [8], [12], [18], [19], [28] combine active set strategies with projections on the feasible set which, in this case, are very simple to compute. See [1].

In all known active-projection algorithms, given the current feasible iterate x^k, a trial point z is computed and, if z is nonfeasible, a corrected trial point z' is defined as the projection of z to the feasible box. Usually, for accepting the projected trial point it is required that $f(z') < f(x^k)$. Otherwise, the direction $z - x^k$ is reduced and a new projection is computed.

The main contribution of this paper is in showing that the acceptance criterion above can be relaxed, both from the theoretical and the practical point of view. In theory, we show that under a relaxed form of the acceptance criterion we obtain the same results as those that hold under monotonicity. In practice, we observe that the relaxed criterion offers a more effective way of solving the problems.

In Section 2 of this paper we describe the nonmonotone algorithm and prove its convergence. Convergence proofs are similar to those given for the monotone method in [2]. In Section 3 we describe the implementation and present numerical experiments. Finally, conclusions are given in Section 4.

2. The nonmonotone algorithm

The problem considered in this work is

$$\text{minimize } f(x) \text{ subject to } x \in \Omega, \tag{1}$$

where $\Omega = \{x \in \mathbb{R}^n \mid l \leqslant x \leqslant u,\ l < u\}$ is compact, $f(x) = \frac{1}{2}x^T Hx + b^T x$ and $l, u \in \mathbb{R}^n$. We denote

$$\gamma = \min\{u_i - l_i,\ i = 1, \dots, n\}$$

and

$$g(x) \equiv -\nabla f(x) \equiv -(Hx + b)$$

for all $x \in \mathbb{R}^n$. Let $L > 0$ be such that $\|H\| \leqslant L$, where $\|\cdot\|$ denotes the 2-norm of vectors or matrices. Therefore, for all $x, z \in \mathbb{R}^n$, we have that

$$f(z) - f(x) - \nabla f(x)^T (z - x) = \frac{1}{2}(z - x)^T H(z - x) \leqslant \frac{L}{2}\|z - x\|^2. \tag{2}$$

Given $I \subset \{1, 2, \dots, 2n\}$ such that i and $n + i$ do not belong to I simultaneously, we define the open face $F_I \subset \Omega$ as

$$F_I = \{x \in \Omega \mid x_i = l_i \text{ if } i \in I,\ \ x_i = u_i \text{ if } n + i \in I,\ \ l_i < x_i < u_i \text{ otherwise}\}.$$

As in [2], [18], [20], we denote by $\overline{F}_I$ the closure of each open face and by $[F_I]$ the smallest linear manifold that contains F_I. For each $x \in \Omega$ let us define the (negative) *projected gradient* $g_P(x) \in \mathbb{R}^n$ as

$$g_P(x)_i = \begin{cases} 0 & \text{if} \quad x_i = l_i \text{ and } \dfrac{\partial f}{\partial x_i}(x) > 0, \\ 0 & \text{if} \quad x_i = u_i \text{ and } \dfrac{\partial f}{\partial x_i}(x) < 0, \\ -\dfrac{\partial f}{\partial x_i}(x) & \text{otherwise.} \end{cases} \tag{3}$$

The stationary points of (1) are defined by

$$g_P(x) = 0. \tag{4}$$

As is well known, local minimizers of (1) are stationary points. For each $x \in F_I$ let us define the *internal gradient* $g_I(x) \in \mathbb{R}^n$ as

$$g_I(x)_i = \begin{cases} 0 & \text{if} \quad x_i = l_i \text{ or } x_i = u_i, \\ -\dfrac{\partial f}{\partial x_i}(x) & \text{otherwise.} \end{cases} \tag{5}$$

We also define, for $x \in F_I$,

$$g_C(x)_i = \begin{cases} 0 & \text{if} \quad l_i < x_i < u_i, \\ 0 & \text{if} \quad x_i = l_i \quad \text{and} \quad \dfrac{\partial f}{\partial x_i}(x) > 0, \\ 0 & \text{if} \quad x_i = u_i \quad \text{and} \quad \dfrac{\partial f}{\partial x_i}(x) < 0, \\ -\dfrac{\partial f}{\partial x_i}(x) & \text{otherwise.} \end{cases} \tag{6}$$

The vector $g_C(x)$ was introduced in [17], and named *chopped gradient.* Observe that for all $x \in F_I$ we have

$$g_P(x) = g_I(x) + g_C(x)$$

and that $g_I(x) \perp g_C(x)$.

Algorithm 2.1, given below, describes the method analyzed in this paper. As in [2], [12], [18], [20], when it is recommendable to abandon some constraints, the algorithm leaves the closure of a face $\overline{F}_I$ following the direction $g_C(x^k)$. A fraction of the decrease obtained at this iteration is kept in memory in order to be used later. In fact, when, at a later iteration, we need to add constraints to the active set, the objective function only needs to decrease in relation to the last leaving-face iteration.

This will allow us to use projections to the feasible set to define the iterations where constraints must be added in a more agressive way than permitted by monotone criteria.

Algorithm 2.1.
Let $\eta \in (0,1)$ and $\sigma \in (0,1]$ be given independently of k, let $x^0 \in \Omega$ be an arbitrary initial point and let $c_0 \geqslant f(x^0)$. The algorithm defines a sequence $\{x^k\}$ in Ω and stops when $\|g_P(x^k)\| = 0$. Let us assume that $x^k \in \Omega$ is such that $\|g_P(x^k)\| \neq 0$. Let $I = I(x^k)$ be such that $x^k \in F_I$ and let the function $\Phi(x)$ be defined as

$$\Phi(x) = \operatorname{argmin}\{f(y) \mid y = x + \lambda g_C(x) \text{ and } y \in \Omega\}. \tag{7}$$

The following steps define the procedure for obtaining x^{k+1}.

Step 1: If

$$\|g_C(x^k)\| > \eta \|g_P(x^k)\|, \tag{8}$$

then set $x^{k+1} = \Phi(x^k)$ and define

$$c_{k+1} = f(x^k) - \sigma\big[f(x^k) - f(x^{k+1})\big]. \tag{9}$$

Else go to Step 2.

Step 2: Compute a point $z^k \in [F_I]$ such that $f(z^k) < f(x^k)$. If $z^k \in F_I$ then set $x^{k+1} = z^k$ and $c_{k+1} = c_k$. Else go to Step 3.

Step 3: Find $x^{k+1} \in \overline{F}_I - F_I$ such that $f(x^{k+1}) \leqslant c_k$. Define $c_{k+1} = c_k$.

Projections are not mentioned explicitly in Algorithm 2.1. However, they are implicit at Step 3, when we seek x^{k+1} on the boundary of F_I. The goal is that the number of active constraints at x^{k+1} should be, in this case, much greater than the number of active constraints at x^k. For this reason, the decreasing criterion is, in general, weaker than that used at iterations of a different type.

When the algorithm explores a face F_I at Step 2, a particular unconstrained quadratic algorithm must be used. In the Algorithmic assumption below, we state the condition that must be fulfilled by such an algorithm in order to fit in with convergence requirements. Later, we show that three reasonable choices for this algorithm satisfy the Algorithmic assumption.

Algorithmic assumption. *For all $k \in \mathbb{N}$, if $x^k \in F_I$, then there exists $j > k$ such that $x^j \notin F_I$ or the algorithm finishes at some $x^j \in F_I$ such that $g_I(x^j) = 0$ and thus $g_P(x^j) = 0$.*

Let us show that this algorithmic assumption is reasonable, in the sense that it is satisfied when one computes z^k using well-known procedures.

The first procedure that we wish to consider is based on classical conjugate gradients [23]. Assume that $k = 0$ or $x^{k-1} \notin F_I$, whereas $x^k \in F_I$. The minimization of f on $[F_I]$ is an unconstrained quadratic minimization problem that can be solved using conjugate gradient iterations with x^k as the initial point. Successive conjugate gradient iterates are denoted $x^{k+1}, x^{k+2}, \ldots$ as far as they belong to F_I and the condition (8) does not hold. The sequence of conjugate gradient iterations is going to be interrupted when one of the following conditions takes place:

(a) An iterate does not belong to F_I.

(b) The conjugate gradient method finds a direction along which the quadratic tends to $-\infty$.

In the case (a) we call z^j the first conjugate gradient iterate that does not belong to F_I. Since $z^j - x^j$ is a descent direction for the quadratic f, it turns out that $x^j + \lambda^+_{\text{break}}(z^j - x^j) \in \overline{F}_I - F_I$ and $f(x^j + \lambda^+_{\text{break}}(z^j - x^j)) < f(x^j) \leqslant c_j$, where

$$\lambda^+_{\text{break}} = \max\{\lambda \geqslant 0 \mid [x^j, x^j + \lambda(z^j - x^j)] \subset \overline{F}_I\}.$$

Therefore, the choice of x^{k+1} at Step 3 is possible.

Assume now that (b) holds, x^j is the last conjugate gradient iterate that belongs to F_I, and d is the direction along which f tends to $-\infty$. If f tends to $-\infty$ along d we define

$$\lambda^+_{\text{break}} = \max\{\lambda \geqslant 0 \mid [x^j, x^j + \lambda d] \subset \overline{F}_I\}$$

and observe, as in the case (a), that $x^j + \lambda^+_{\text{break}} d \in \overline{F}_I - F_I$ and $f(x^j + \lambda^+_{\text{break}} d) < f(x^j) \leqslant c_j$. So, the choice of Step 3 is possible.

If f also tends to $-\infty$ along $-d$, we also define

$$\lambda^-_{\text{break}} = \min\{\lambda \leqslant 0 \mid [x^j, x^j + \lambda d] \subset \overline{F}_I\}.$$

So, either $f(x^j + \lambda^+_{\text{break}} d) < f(x^j) \leqslant c_j$ or $f(x^j + \lambda^-_{\text{break}} d) < f(x^j) \leqslant c_j$. In both cases, the choice of Step 3 is possible.

It has been proved in [20] that, if neither (a) nor (b) take place, then, after a finite number of steps, the conjugate gradient iterate has null gradient. Therefore, either there exists $j \geqslant k$ such that x^j satisfies (8) or $g_P(x^j) = 0$.

The second procedure we wish to analyze for the computation of z^k is based on the Cholesky factorization of the submatrix of H that is formed by the rows and columns corresponding to the free variables on F_I. If the Cholesky factorization can be completed, it can be used to compute the minimizer z^k of the quadratic f on $[F_I]$. If $z^k \in F_I$ then either $g_P(z^k) = 0$ or the condition (8) must be satisfied. If $z^k \notin F_I$ we proceed as in the conjugate gradient case. So, we only need to analyze the case in which the matrix is not positive definite, so that the Cholesky factorization cannot be completed. In this case, if the current point is not a minimizer, standard inexpensive procedures [21] allow one to compute a direction d such that f tends to $-\infty$ and we can find, as above, a point on the boundary such that the objective function value is smaller than $f(x^k)$.

Finally, a preconditioned conjugate gradient (PCG) procedure could be used at Step 2 of Algorithm 2.1. See [22]. In this case, the analysis is similar to that of the ordinary conjugate gradient algorithm except that z^k should be defined as the result of the application of more than one PCG iterations, since this algorithm is not necessarily monotone for the original quadratic. Nevertheless, the rest of the standard analysis is valid.

Below, we show that Algorithm 2.1 is well defined, that is to say, that all iterations can be completed. As other results of this section, the proof is similar to a proof given in [2] for a monotone algorithm with a different algorithmic assumption.

Theorem 1. *Algorithm* 2.1 *is well defined.*

Proof. If the condition (8) at Step 1 is satisfied, then $g_C(x^k) \neq 0$, so $\Phi(x^k)$ is well defined. If the condition (8) does not hold, we execute Step 2. Since $g_I(x^k) \neq 0$, the existence of $z^k \in [F_I]$ such that $f(z^k) < f(x^k)$ is guaranteed.

Now, if $z^k \notin \overline{F}_I$, since $\varphi(\lambda) \equiv f(x^k + \lambda(z^k - x^k))$ is a one-dimensional quadratic, we have

$$f(x^k + \lambda^+_{\text{break}}(z^k - x^k)) < f(x^k),$$

or

$$f(x^k + \lambda^-_{\text{break}}(z^k - x^k)) < f(x^k),$$

where

$$\lambda^+_{\text{break}} = \max\{\lambda \geqslant 0 \mid [x^k, x^k + \lambda(z^k - x^k)] \subset \overline{F}_I\}$$

and

$$\lambda^-_{\text{break}} = \min\{\lambda \leqslant 0 \mid [x^k, x^k + \lambda(z^k - x^k)] \subset \overline{F}_I\}.$$

Therefore, the choice of $x^{k+1} \in \overline{F}_I - F_I$ satisfying $f(x^{k+1}) < f(x^k)$, in Step 3, is possible. $\square$

The following lemma quantifies the amount of decrease of the objective function when a leaving-face iteration is computed at Step 1 of Algorithm 2.1. In monotone algorithms, all the iterates x^j such that $j \geqslant k+1$ satisfy $f(x^j) < f(x^k)$. Our new algorithm is greedy in the sense that changing the current face is considered a desirable feature, when these changes do not damage convergence. For this reason, in the nonmonotone algorithm the decrease of f at new iterations is only a fraction σ of the decrease at the latest leaving-face iteration.

Lemma 1. *If x^{k+1} is obtained at Step 1 of Algorithm 2.1 then*

$$f(x^k) - f(x^{k+1}) \geqslant \min\left\{\frac{\eta\gamma}{2}\|g_P(x^k)\|,\ \frac{\eta^2}{2L}\|g_P(x^k)\|^2\right\}.$$

P r o o f. The proof of this lemma was given in [2]. Let us sketch it here for the sake of completeness. Since x^{k+1} is obtained at Step 1, hence $g_C(x^k) \neq 0$. Hence, $x^k + \lambda g_C(x^k) \in \Omega$ for all $\lambda \in [0, \tilde{\lambda}]$, where $\tilde{\lambda} = \gamma / \|g_C(x^k)\|$. Let us consider the quadratic function given by

$$\mu(\lambda) = f(x^k + \lambda g_C(x^k)) = f(x^k) + \lambda \nabla f(x^k)^T g_C(x^k) + \frac{1}{2}\lambda^2 g_C(x^k)^T H g_C(x^k).$$

If $g_C(x^k)^T H g_C(x^k) > 0$ then the unique minimizer of $\mu(\lambda)$ is given by

$$\lambda^* = \frac{\|g_C(x^k)\|^2}{g_C(x^k)^T H g_C(x^k)}.$$

There exist three possibilities:
(i) $g_C(x^k)^T H g_C(x^k) > 0$ and $x^k + \lambda^* g_C(x^k) \notin \Omega$;
(ii) $g_C(x^k)^T H g_C(x^k) > 0$ and $x^k + \lambda^* g_C(x^k) \in \Omega$;
(iii) $g_C(x^k)^T H g_C(x^k) \leqslant 0$.

In the first case, we obtain

$$f(x^k) - f(x^{k+1}) > \frac{\gamma}{2}\|g_C(x^k)\| > \frac{\eta\gamma}{2}\|g_P(x^k).$$

If (ii) holds, we have

$$f(x^k) - f(x^{k+1}) > \frac{1}{2L}\|g_C(x^k)\|^2 > \frac{\eta^2}{2L}\|g_P(x^k)\|^2.$$

Finally, when (iii) holds, then

$$f(x^k) - f(x^{k+1}) > \gamma\|g_C(x^k)\| > \eta\gamma\|g_P(x^k)\|.$$

The desired result follows from these inequalities. □

The following is a global convergence result. It says that, given an arbitrary tolerance $\varepsilon > 0$ the algorithm necessarily finds an iterate such that the norm of the projected gradient g_P is smaller than ε after a finite number of iterations. By the compactness of the feasible region, this implies that there exists a cluster point where the projected gradient vanishes.

Theorem 2. *Let the sequence $\{x^k\}$ be generated by Algorithm 2.1. Then either the algorithm terminates at a point x^k such that $g_P(x^k) = 0$ or the sequence is infinite and the condition (8) is satisfied infinitely many times. In the latter case, calling $K_1 \subset \mathbb{N}$ the set of indices k such that (8) holds, we have $\lim_{k \in K_1} \|g_P(x^k)\| = 0$. Moreover, any limit point of the subsequence $\{x^k\}_{k \in K_1}$ is stationary.*

P r o o f. If the condition (8) is satisfied only a finite number of times, it follows that there exists $k \in \mathbb{N}$ such that $x^j \in \overline{F}_I$ for all $j \geqslant k$. But the face to which x^{j+1} belongs is necessarily contained in the face to which x^j belongs, therefore, there exist k' and F_J such that $x^j \in F_J$ for all $j \geqslant k'$. Therefore, by the Algorithmic assumption, there exists $j \geqslant k'$ such that $g_I(x^j) = 0$. Since condition (8) does not hold at x^j it follows that $g_P(x^j) = 0$.

Assume that the algorithm does not terminate and so the condition (8) is satisfied whenever $k \in K_1$, where K_1 is an infinite subset of $\mathbb{N}$. Let us prove that $\lim_{k \in K_1} g_P(x^k) = 0$. If this is not true, there exists $\varepsilon > 0$ and an infinite set of indices $K_2 \subset K_1$ such that

$$\|g_P(x^k)\| > \varepsilon \quad \text{for all } k \in K_2. \tag{10}$$

By Lemma 1 and the conditions of Steps 2 and 3, we have that $\lim_{k \to \infty} c_k = -\infty$ and hence, $\lim_{k \to \infty} f(x^k) = -\infty$. This is impossible, since f is continuous and Ω is compact. Therefore, (10) cannot be true. Therefore, $\lim_{k \in K_1} g_P(x^k) = 0$. Let K_2 be an infinite subset of K_1 such that $\lim_{k \in K_2} x^k = x^*$. Since $g_P(x)$ is lower-semicontinuous it follows that $g_P(x^*) = 0$, as we wanted to prove. □

A stationary point x^* of (1) is said to be degenerate if there exists $i \in \{1, \ldots, n\}$ such that $x_i^* = \ell_i$ or $x_i^* = u_i$, whereas $\frac{\partial f}{\partial x_i}(x^*) = 0$. Below we show that, in the absence of degenerate points, convergence of Algorithm 2.1 takes place in a finite number of iterations.

Theorem 3. *If all stationary points of a sequence $\{x^k\}$ generated by Algorithm 2.1 are non-degenerate, then there exists $\underline{k} \in \mathbb{N}$ such that $g_P(x^{\underline{k}}) = 0$.*

P r o o f. By Theorem 2, we only need to prove that the inequality (8) cannot hold infinitely many times. Suppose, by contradiction, that the test (8) is satisfied for all $k \in K_1$, K_1 being an infinite subset of $\mathbb{N}$. Since the number of open faces is finite, there exists a face F_J and an infinite set $K_3 \subset K_1$ such that $x^k \in F_J$ and $x^{k+1} \notin \overline{F}_J$ for all $k \in K_3$. Therefore, for all $k \in K_3$, x^{k+1} is obtained by Step 1 of Algorithm 2.1. This implies that one of the constraints that define F_J must be relaxed in an infinite subset $K_4 \subset K_3$. We may suppose, without loss of generality, that this constraint is $x_i = \ell_i$. So, for $k \in K_4$, we have $i \in I(x^k)$ and $i \notin I(x^{k+1})$. This implies that, for all $k \in K_4$,

$$-\frac{\partial f}{\partial x_i}(x^k) > 0.$$

But, by Theorem 2, $\lim\limits_{k \in K_4} g_P(x^k) = 0$. So,

$$\lim_{k \in K_4} \max\left\{0, -\frac{\partial f}{\partial x_i}(x^k)\right\} = 0.$$

Therefore,

$$\lim_{k \in K_4} \frac{\partial f}{\partial x_i}(x^k) = 0. \tag{11}$$

Let x^* be a limit point of $\{x^k\}_{k \in K_4}$. By Theorem 2, x^* is a stationary point and, since $x_i^k = \ell_i$ for all $k \in K_4$, we see that $x_i^* = \ell_i$. Finally, by (11), $\frac{\partial f}{\partial x_i}(x^*) = 0$. This implies that x^* is a degenerate stationary point, contradicting the hypothesis of the theorem. $\square$

It is worth noticing that for a strictly convex quadratic f and sufficiently large η the finite termination property holds even for a degenerate solution [12].

3. Numerical experiments

For implementing the idea introduced in this paper, we modified the quadratic programming code described in [2], [18], [20], which had been extensively tested both in academic and practical problems [10]. Step 2 of Algorithm 2.1 was implemented using the conjugate gradient method. When a conjugate gradient iterate z not belonging to F_I is found, we compute the maximum steplength λ_{break} such that $x^k + \lambda_{\text{break}}(z - x^k)$ does not violate the constraints. Clearly, $f(x^k + \lambda_{\text{break}}(z - x^k)) < f(x^k) \leqslant c_k$, but we do not use this point as next iterate because the number of active constraints would be generally increased only by one. Instead, we multiply the steplength by a factor

(5 in our experiments) and project the corresponding point $y^\nu + 5^\nu \lambda_{\text{break}}(z - x^k)$ to Ω for $\nu = 0, 1, 2, \ldots$ obtaining the projected feasible point $y^{\nu+1}$, where $y^0 = x^k$. This extrapolation process is interrupted when $y^{\nu+1} = y^\nu$ or when $f(y^{\nu+1}) > c_k$. Then, we choose $x^{k+1} = y^\nu$. We proceed in a similar way when the conjugate gradient method finds a direction of nonpositive curvature. We tested Algorithm 2.1 with $\sigma = 0.1$ (nonmonotone version) against the monotone method described in [2], where the extrapolation process described above is interrupted whenever $f(y^{\nu+1}) \geqslant f(y^\nu)$. In all the experiments we used $\eta = 0.9$ and $c_0 = f(x^0)$. We declared convergence if $\|g_P(x^k)\| \leqslant 10^{-5}$.

In addition to our basic algorithm with $\sigma = 0.1$ and its monotone counterpart, we ran the well-known code `LANCELOT` with the same set of problems, namely all the bound-constrained quadratic problems of the `CUTE` collection with the largest admissible dimension (greater than or equal to 1000) without modification of the internal variables of the "double large" installation [3], and the following choices:

- `exact-second-derivatives-used`
- `cg-method-used (CG)` or `pentadiagonal-preconditioned-cg-method-used (PCG)`
- `exact-Cauchy-point-required (EX)` or `inexact-Cauchy-point-required (IN)`
- `infinity-norm-trust-region-used`
- `gradient-tolerance` 10^{-5}
- `maximum-number-of-iterations 1000`

The tests were developed in `Fortran77` double precision and run in a `SUN Ultra1 Creator`. The results are given in Tab. 1, where the following notation is used: `N` denotes the dimension of each problem; the value `IT` gives the number of inner iterations (performed by the plain or preconditioned conjugate gradient method) and `T` is the CPU time in seconds spent by each test. The notation `CGEX, CGIN, PCGEX, PCGIN` was defined in the choices stated above. Tab. 2 contains the results for the monotone and nonmonotone algorithms, the total number of iterations (`IT`), the number of matrix-vector products performed by each one of them (`PROD`), and the CPU time in seconds (`T`) being reported. We remark that the number of matrix-vector products performed by `LANCELOT` is not included in Tab. 1 because it is not reported by the code.

Tabs. 3 and 4 summarize the geometric means of the comparative numerical results reported in Tabs. 1 and 2, with similar notation. This average was chosen to accomodate the very different and problem dependent order of magnitude of the results. The numbers show that for the tests using `LANCELOT`, the combination preconditioned conjugate gradient and exact Cauchy point performed best. The nonmonotone algorithm is slightly superior to the monotone one and has a better performance than LANCELOT. Considering the average time spent per iteration, that is T/IT, we obtain 0.065, 0.051, 0.088 and 0.074 seconds for the options CGEX, CGIN, PCGEX, PCGIN of LANCELOT and 0.078, 0.077 for the monotone algorithm and the nonmonotone algorithm, respectively. Therefore, as expected, the preconditioned version is more expensive than the plain conjugate gradient, and computing inexact Cauchy points is slightly cheaper than working with the exact ones. The average time per iteration of the quadratic solver is practically the same for the monotone and nonmonotone versions, and comparable with the preconditioned/inexact option of LANCELOT.

PROBLEM (N)	CGEX		CGIN		PCGEX		PCGIN	
	IT	T	IT	T	IT	T	IT	T
BIGGSB1 (1000)	66509	117.50	66510	112.40	500	5.40	501	5.40
BQPGAUSS (2003)	9511	117.10	9083	113.20	2928	46.40	2771	44.10
CHENHARK (1000)	14	0.03	17	0.04	3	0.03	4	0.04
CVXBQP1 (10000)	1	1.20	6411	95.90	1	1.20	6411	156.50
JNLBRNG1 (15625)	2556	256.60	2369	241.20	1810	205.10	1847	209.90
JNLBRNG2 (15625)	2673	257.10	2700	260.90	912	101.30	857	98.10
JNLBRNGA (15625)	2135	202.70	2134	202.20	1327	144.50	1359	145.70
JNLBRNGB (15625)	4439	390.80	4617	402.90	329	36.90	364	41.20
NCVXBQP1 (10000)	0	1.20	10000	190.30	0	1.20	10003	329.30
NCVXBQP2 (10000)	435	3.70	10183	196.60	407	4.30	10024	334.40
NCVXBQP3 (10000)	366	3.60	10056	195.70	359	4.10	9987	331.70
NOBNDTOR (14884)	1539	167.50	1539	167.70	790	108.20	790	107.00
OBSTCLAE (15625)	7608	834.00	7614	858.90	7409	1090.90	7410	1097.20
OBSTCLAL (15625)	805	73.60	805	70.50	481	51.70	481	52.00
OBSTCLBL (15625)	3259	328.10	2578	260.80	2761	349.90	2117	271.30
OBSTCLBM (15625)	1483	167.80	601	62.10	1377	201.30	506	66.50
OBSTCLBU (15625)	1102	113.50	1110	112.90	806	101.40	821	101.90
ODNAMUR (11130)	51556	1224.10	49092	1190.9	30006	1377.70	31458	1410.30
PENTDI (1000)	0	0.02	0	0.02	0	0.02	0	0.02

Table 1. Comparative results of LANCELOT.

PROBLEM (N)	CGEX		CGIN		PCGEX		PCGIN	
	IT	T	IT	T	IT	T	IT	T
TORSION1 (14884)	1347	125.90	1347	124.90	794	90.00	794	89.00
TORSION2 (14884)	5053	563.30	4994	558.50	4339	646.10	4652	692.40
TORSION3 (14884)	390	31.40	390	30.40	242	23.50	242	23.40
TORSION4 (14884)	5954	651.10	9042	887.20	5640	795.80	8745	1115.40
TORSION5 (14884)	114	8.70	114	8.30	73	7.00	73	7.40
TORSION6 (14884)	7355	746.90	10477	913.70	4892	521.80	6442	700.60
TORSIONA (14884)	1339	134.80	1339	133.90	796	97.00	796	97.00
TORSIONB (14884)	5000	593.80	5084	603.10	4025	621.40	4249	693.90
TORSIONC (14884)	390	34.70	390	33.40	242	25.50	242	25.50
TORSIOND (14884)	9430	986.50	9396	995.70	9134	1224.80	9194	1265.60
TORSIONE (14884)	114	9.80	114	9.30	73	7.80	73	7.80
TORSIONF (14884)	5343	484.10	11201	1064.20	4980	577.30	10171	1185.90

Table 1 (cont.). Comparative results of LANCELOT.

PROBLEM (N)	MONOTONE ALGORITHM			NONMONOTONE ALGORITHM		
	IT	PROD	T	IT	PROD	T
BIGGSB1 (1000)	3610	3631	6.20	3518	3530	5.96
BQPGAUSS (2003)	6363	6537	84.62	5836	6043	76.99
CHENHARK (1000)	17	25	0.05	14	20	0.05
CVXBQP1 (10000)	1	14	0.47	1	14	0.47
JNLBRNG1 (15625)	1131	1715	185.07	658	1010	101.99
JNLBRNG2 (15625)	935	1053	109.69	938	1065	106.20
JNLBRNGA (15625)	483	558	54.61	485	560	53.84
JNLBRNGB (15625)	3554	3669	348.70	2870	3008	281.39
NCVXBQP1 (10000)	1	8	0.34	1	8	0.33
NCVXBQP2 (10000)	45	81	1.87	52	100	2.22
NCVXBQP3 (10000)	59	99	2.26	53	123	2.67
NOBNDTOR (14884)	431	478	51.45	449	512	56.21

Table 2. Comparative results of quadratic solvers.

PROBLEM (N)	MONOTONE ALGORITHM			NONMONOTONE ALGORITHM		
	IT	PROD	T	IT	PROD	T
OBSTCLAE (15625)	386	589	63.79	340	505	47.36
OBSTCLAL (15625)	251	317	27.31	280	360	29.92
OBSTCLBL (15625)	375	826	89.85	377	837	88.03
OBSTCLBM (15625)	199	314	36.79	206	349	38.96
OBSTCLBU (15625)	313	475	50.40	372	615	63.20
ODNAMUR (14884)	37778	40247	1540.64	35222	41559	1574.50
PENTDI (1000)	1	2	0.02	1	2	0.02
TORSION1 (14884)	363	395	37.32	381	403	37.33
TORSION2 (14884)	435	612	63.72	391	438	41.24
TORSION3 (14884)	164	168	13.16	164	168	13.11
TORSION4 (14884)	165	181	14.81	165	181	14.80
TORSION5 (14884)	76	78	5.59	78	81	5.84
TORSION6 (14884)	79	93	7.29	79	94	7.31
TORSIONA (14884)	349	385	38.82	385	427	42.15
TORSIONB (14884)	443	607	67.38	405	442	44.27
TORSIONC (14884)	183	193	16.37	176	199	16.58
TORSIOND (14884)	178	193	16.95	178	193	16.63
TORSIONE (14884)	79	87	6.75	84	92	7.12
TORSIONF (14884)	79	97	8.35	83	104	8.90

Table 2 (cont.). Comparative results of quadratic solvers.

CGEX		CGIN		PCGEX		PCGIN	
IT	T	IT	T	IT	T	IT	T
906.05	59.02	2022.80	103.69	501.71	43.95	1130.40	83.38

Table 3. Geometric means of the comparative results of LANCELOT.

MONOTONE ALGORITHM			NONMONOTONE ALGORITHM		
IT	PROD	T	IT	PROD	T
191.50	279.43	14.92	186.61	275.07	14.38

Table 4. Geometric means of the comparative results of quadratic solvers.

To illustrate an individual analysis of the performance of each problem, in Fig. 1 we plot the ratios between the results of the nonmonotone and of the monotone algorithm as far as iterations are concerned. The numbers in the horizontal axes correspond to the order of appearance of the problems in Tab. 2. Although the large majority of the results is concentrated in the range [0.9, 1.1], namely 77% of the problems, there are more problems for which the ratios are below 0.9 (13%) than above 1.1 (10%), indicating a slight advantage towards nonmonotonicity. Problems `CHENHARK` (number 3), `JNLBRNG1` (number 5) and `JNLBRNGB` (number 8) are more favorable to the nonmonotone strategy, whereas the opposite happens to problems `NCVXBQP2` (number 10), `OBSTCLAL` (number 14) and `OBSTCLBU` (number 17), for which the monotone algorithm performs better.

In Figs. 2 and 3 we visualize the comparative results between the nonmonotone algorithm and the combination that performed best for `LANCELOT` according to Tab. 3, namely, using the preconditioned conjugate gradient and computing the exact Cauchy point (`PCGEX`). We plot the logarithms, with the base 10, of the ratios between the results of the nonmonotone algorithm and `LANCELOT`, analyzing, in Fig. 2, the number of iterations performed, and, in Fig. 3, the CPU time spent. Using the notation

$$\varrho_{IT} = \frac{\#\ \text{iterations of nonmonotone algorithm}}{\#\ \text{PCG iterations of } \texttt{LANCELOT}}$$

and

$$\varrho_{TE} = \frac{\text{CPU time spent by nonmonotone algorithm}}{\text{CPU time spent by } \texttt{LANCELOT}},$$

Figs. 2 and 3 can be summarized as follows:

Range	$\frac{1}{70} \leqslant \varrho_{IT} < 0.8$	$0.8 \leqslant \varrho_{IT} < 1.25$	$1.25 \leqslant \varrho_{IT} \leqslant 9$
Problems	64%	23%	13%

Table 5. Statistical results of Fig. 2.

Range	$\frac{1}{80} \leqslant \varrho_{TE} < 0.8$	$0.8 \leqslant \varrho_{TE} < 1.25$	$1.25 \leqslant \varrho_{TE} \leqslant 8$
Problems	71%	19%	10%

Table 6. Statistical results of Fig. 3.

From Tab. 5 and 6 we observe that, although the nonmonotone algorithm can perform worse than `LANCELOT` (for problem `JNLBRNGB`, number 8, the nonmonotone algorithm is around ten times more costly than `LANCELOT`, both in terms of iterations and CPU time), a worst performance of `LANCELOT` can reach more than fifty times the amount of work spent by the nonmonotone algorithm. This is the case of problems `TORSION4` (number 23), `TORSION6` (number 25), `TORSIOND` (number 29) and `TORSIONF` (number 31).

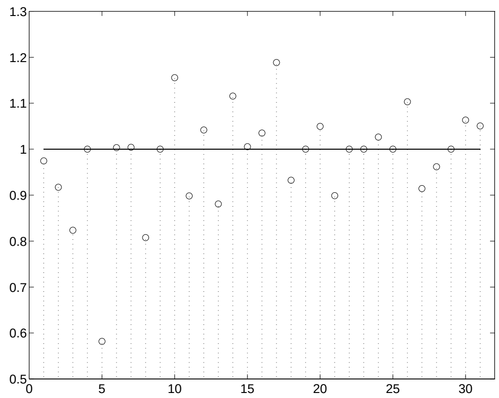

Figure 1. Ratios between the number of iterations of nonmonotone and monotone algorithms for solving `CUTE` bound-constrained quadratic problems.

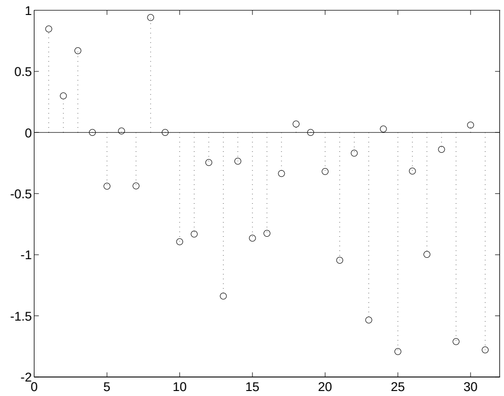

Figure 2. Logarithms, with the base 10, of the ratios between the number of iterations performed by the nonmonotone algorithm and by `LANCELOT` for solving `CUTE` bound-constrained quadratic problems.

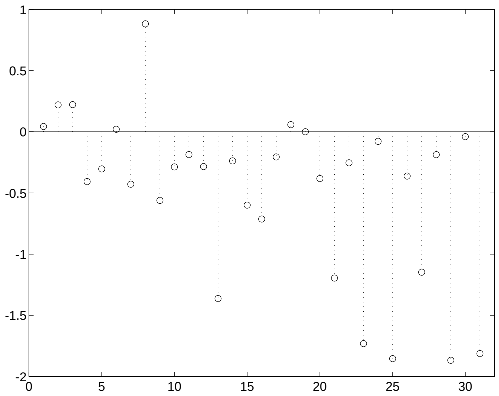

Figure 3. Logarithms, with the base 10, of the ratios between the CPU time spent by the nonmonotone algorithm and by `LANCELOT` for solving `CUTE` bound-constrained quadratic problems.

4. Conclusions

Algorithms for bound constrained quadratic minimization that combine active set strategies with projections to the feasible set are among the most effective for solving practical problems. Projection steps are crucial, since thanks to them many constraints can be added to the working set per iteration, thus decreasing drastically the number of iterations used to solve large-scale problems. The theoretical and practical results of this paper show that it is worthwhile to relax the monotone decrease criterion for the objective function in order to improve practical performance. In other words, the nonmonotone algorithm was shown to be a valid approach. Although the numerical results relative to bound-constrained quadratic problems from `CUTE` do not point very significantly either towards the monotone or to the nonmonotone strategies, we observe that the latter is more relaxed and try, by being more agressive, to change more drastically the active set from one iteration to the other, hopefully decreasing the total amount of work done, despite this may not be the case for some problems. However, when compared with `LANCELOT`, using the best combination of choices for the class of solved problems, the nonmonotone algorithm, with plain conjugate gradients, showed a similar or superior performance for more than 85% of the tests as far as number of iterations and CPU time are concerned. Future research includes a study on preconditioning our family of algorithms for bound-constrained quadratic minimization.

Acknowledgments. The authors are indebted to A. R. Conn, N. I. M. Gould and Ph. L. Toint for making the software `LANCELOT` available for academic research.

References

[1] *D. P. Bertsekas*: Projected Newton methods for optimization problems with simple constraints. SIAM J. Control Optim. *20* (1982), 141–148.

[2] *R. H. Bielschowsky, A. Friedlander, F. A. M. Gomes, J. M. Martínez and M. Raydan*: An adaptive algorithm for bound constrained quadratic minimization. Investigación Oper. *7* (1997), 67–102.

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

[4] *P. Ciarlet*: The Finite Element Method for Elliptic Problems. North Holland, Amsterdam, 1978.

[5] *T. F. Coleman, L. A. Hulbert*: A direct active set algorithm for large sparse quadratic programs with simple bounds. Math. Programming *45* (1989), 373–406.

[6] *A. R. Conn, N. I. M. Gould and Ph. L. Toint*: Global convergence of a class of trust region algorithms for optimization with simple bounds. SIAM J. Numer. Anal. *25* (1988), 433-460; see also SIAM J. Numer. Anal. *26* (1989), 764–767.

[7] *A. R. Conn, N. I. M. Gould and Ph. L. Toint*: A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM J. Numer. Anal. *28* (1988), 545–572.

[8] *R. Dembo, U. Tulowitzki*: On the minimization of quadratic functions subject to box constraints. In: Working Paper B-71. School of Organization and Management, Yale University, New Haven, 1983.

[9] *J. E. Dennis, L. N. Vicente*: Trust-region interior-point algorithms for minimization problems with simple bounds. In: Applied Mathematics and Parallel Computing (Festschrift for Klaus Ritter) (H. Fischer, B. Riedmüller and S. Schäffer, eds.). Physica-Verlag, Springer-Verlag, 1996, pp. 97–107.

[10] *M. A. Diniz-Ehrhardt, M. A. Gomes-Ruggiero and S. A. Santos*: Comparing the numerical performance of two trust-region algorithms for large-scale bound-constrained minimization. Investigación Oper. *7* (1997), 23–54.

[11] *M. A. Diniz-Ehrhardt, M. A. Gomes-Ruggiero and S. A. Santos*: Numerical analysis of leaving-face parameters in bound-constrained quadratic minimization. Relatório de Pesquisa RP52/98. IMECC, UNICAMP, Campinas, Brazil, 1998.

[12] *Z. Dostál*: Box constrained quadratic programming with proportioning and projections. SIAM J. Optim. *7* (1997), 871–887.

[13] *Z. Dostál, A. Friedlander and S. A. Santos*: Solution of contact problems of elasticity by FETI domain decomposition. Contemp. Math. *218* (1998), 82–93.

[14] *Z. Dostál, F. A. M. Gomes Neto and S. A. Santos*: Solution of contact problems by FETI domain decomposition with natural coarse space projection. Comput. Methods Appl. Mech. Engrg. *190* (2000), 1611–1627.

[15] *Z. Dostál, V. Vondrák*: Duality based solution of contact problems with Coulomb friction. Arch. Mech. *49* (1997), 453–460.

[16] *L. Fernandes, A. Fischer, J. J. Júdice, C. Requejo and C. Soares*: A block active set algorithm for large-scale quadratic programming with box constraints. Ann. Oper. Res. *81* (1998), 75–95.

[17] *A. Friedlander, J. M. Martínez*: On the numerical solution of bound constrained optimization problems. RAIRO Rech. Opér. *23* (1989), 319–341.

[18] *A. Friedlander, J. M. Martínez*: On the maximization of a concave quadratic function with box constraints. SIAM J. Optim. *4* (1994), 177–192.

[19] *A. Friedlander, J. M. Martínez and M. Raydan*: A new method for large-scale box constrained quadratic minimization problems. Optimization Methods and Software *5* (1995), 57–74.

[20] *A. Friedlander, J. M. Martínez and S. A. Santos*: A new trust region algorithm for bound constrained minimization. Appl. Math. Optim. *30* (1994), 235–266.

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

[22] *G. H. Golub, Ch. F. Van Loan*: Matrix Computations. The Johns Hopkins University Press, Baltimore and London, 1989.

[23] *M. R. Hestenes, E. Stiefel*: Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards *B 49* (1952), 409–436.

[24] *J. J. Júdice, F. M. Pires*: Direct methods for convex quadratic programming subject to box constraints. Investigação Operacional *9* (1989), 23–56.

[25] *Y. Lin, C. W. Cryer*: An alternating direction implicit algorithm for the solution of linear complementarity problems arising from free boundary problems. Appl. Math. Optim. *13* (1985), 1–17.

[26] *P. Lötstedt*: Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. SIAM J. Sci. Comput. *5* (1984), 370–393.

[27] *P. Lötstedt*: Solving the minimal least squares problem subject to bounds on the variables. BIT *24* (1984), 206–224.

[28] *J. J. Moré, G. Toraldo*: On the solution of large quadratic programming problems with bound constraints. SIAM J. Optim. *1* (1991), 93–113.

[29] *R. H. Nickel, J. W. Tolle*: A sparse sequential programming algorithm. J. Optim. Theory Appl. *60* (1989), 453–473.

[30] *M. Raydan*: On the Barzilai and Borwein choice of steplength for the gradient method. IMA J. Numer. Anal. *13* (1993), 321–326.

[31] *E. K. Yang, J. W. Tolle*: A class of methods for solving large, convex quadratic programs subject to box constraints. Tech. Rep. UNC/ORSA/TR-86-3, Dept. of Oper. Research and Systems Analysis, Univ. of North Carolina, Chapel Hill, NC. (1986).

Author's address: *M. A. Diniz-Ehrhardt, M. A. Gomes-Ruggiero, J. M. Martínez, S. A. Santos*, Institute of Mathematics, Statistics and Scientific Computation, (IMECC), State University of Campinas (UNICAMP), CP 6065, 13083-970 Campinas SP, Brazil, e-mail: `martinez@ime.unicamp.br`; *Z. Dostál* on visit to IMECC-UNICAMP, permanent address Department of Applied Mathematics, VŠB—Technical University of Ostrava, Ostrava, Czech Republic, e-mail: `zdenek.dostal@vsb.cz`.