A limited-memory multipoint symmetric secant method for bound constrained optimization

Oleg P. Burdakov * José Mario Martínez † Elvio A. Pilotta ‡

September 10, 2000

Abstract

A new algorithm for solving smooth large-scale minimization problems with bound constraints is introduced. The way of dealing with active constraints is similar to the one used in some recently introduced quadratic solvers. A limited-memory multipoint symmetric secant method for approximating the Hessian is presented. Positive-definiteness of the Hessian approximation is not enforced. A combination of trust-region and conjugate-gradient approaches is used to explore useful information. Global convergence is proved for a general model algorithm. Results of numerical experiments are presented.

Keywords: large-scale optimization, box constraints, active set methods, gradient projection, trust region, conjugate gradients, multipoint symmetric secant methods, global convergence.

*Division of Optimization, Department of Mathematics, Linköping University, S - 581 83 Linköping, Sweden. Part of the work of this author was done while he was visiting the University of Campinas under support of FAPESP (Grant 1197-11730-5). E-Mail: olbur@mai.liu.se

†Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. This author was supported by PRONEX-Optimization, FAPESP (Grant 90-3724-6), CNPq and FAEP-UNICAMP. E-Mail: martinez@ime.unicamp.br

‡Facultad de Matemática, Astronomía y Física, FaMAF-CIEM, Universidad Nacional de Córdoba, Ciudad Universitaria (5000), Argentina. This author was supported by PRONEX-Optimization, FAPESP (Grant 90-3724-6), CNPq, FAEP-UNICAMP and SECyT-UNC (Grant 194-2000). E-Mail: pilotta@mate.uncor.edu

1 Introduction

We consider the following bound (or box-) constrained optimization problem:

$$\text{Minimize } f(x) \text{ subject to } x \in \Omega. \tag{1}$$

We assume that the function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and the box Ω is given by

$$\Omega = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u\},$$

where $\ell < u$.

This problem is very important in practical optimization. A lot of applied problems admit mathematical models of type (1). Moreover, one of the most effective approaches for solving general constrained optimization problems, based on augmented Lagrangians, relies on effective algorithms for solving (1) (see [16, 17, 19, 38]). Finally, in recent works on complementarity and variational inequalities, these problems are reduced to bound constrained minimization problems in an efficient way (see [1, 2, 3, 4] and references therein).

All practical methods for solving (1) are iterative. Given $x^k \in \Omega$, many methods construct a quadratic model of f, whose gradient at x^k coincides with the gradient of f, and whose Hessian is an approximation of the Hessian of f. Many different (but related) ways of using this approximation were considered in recent publications. See [13, 16, 30, 34, 37].

In the algorithms introduced in this paper we also use quadratic models, but the way of treating constraints differs from the ones described in [13, 16, 30, 34, 37, 41]. Roughly speaking, our proposal is to treat constraints in the same way the quadratic solvers [5, 23, 33] do. This means that an algorithm for unconstrained minimization on the current face is used, until a separate indicator says that this is not worthwhile anymore. In this case, the face is abandoned along a direction defined in [31, 32, 33] for convex minimization. For this direction, interesting physical interpretations are given in [23]. See also [24, 25, 26, 27, 28, 29]. When, in the unconstrained search process within a face, the algorithm hits a bound, several new constraints are incorporated.

Existing algorithms use different ways of constructing Hessians for the quadratic models. The true Hessian of f, the limited-memory BFGS and SR1 quasi-Newton approximations are the best known alternatives (see [13, 16]). An interesting Gauss-Newton-type approximation of the Hessian of augmented Lagrangians was considered in [38]. The cases in which the true Hessian is very costly or difficult to compute and its finite-difference approximation is also time-consuming are not rare in practice. In these cases, it is possible to use the truncated-Newton approach [6, 21], where each "Hessian $\times$ vector" product is

replaced by an incremental quotient. However, since each of these products involves an additional gradient evaluation, this alternative can also be inefficient. Moreover, in this approach, information about the Hessian matrix obtained at the current iteration is not used in the next ones.

On the other hand, quasi-Newton approximations of the Hessian (see, e.g., [22]) are able to accumulate information along the process. These approximations involve only one gradient evaluation per iteration. In the large-scale case, conventional quasi-Newton methods generate dense Hessian approximations that cannot be stored (or manipulated) explicitly. To overcome these difficulties, limited-memory alternatives have been developed (see [13, 14, 35] and references therein).

Our limited-memory approach will be based on the multipoint symmetric secant approximations of the Hessian matrix proposed in [9]. These approximations are an extension of the classical multipoint secant scheme (see [39, 42] and references therein) with the advantage that they exploit the symmetry of the Hessian matrix in a natural way. They also differ from those introduced in [44]. The idea is that the Hessian approximation should be such that the gradient of the quadratic model should interpolate the gradient of f at some previous points. However, since this objective conflicts with the symmetry, the most "fresh" information carried by gradient values is privileged with respect to older information. The tendency to instability of the sequential secant methods is overcome with the approach developed in [10, 11, 12]. A combination of the box-trust-region (BTR) and conjugate gradient (CG) approaches allows us to take advantage of negative curvature information.

The organization of this paper is as follows. An algorithmic outline of the main algorithm is described in Section 2, where basic global convergence theorems are also proved. In Section 3 we present a sub-algorithm that can be used by the main algorithm for minimization within the current face. The sub-algorithm is based on the BTR and CG approaches, and it assumes that a Hessian approximation is given. A limited-memory multipoint symmetric secant approximation is introduced in Section 4. In Section 5 we discuss some implementation details. Numerical results are presented in Section 6. Finally, conclusions are given in Section 7.

2 Main algorithmic model and global convergence

The main algorithm presented in this paper is an active-set method with a special procedure for dropping constraints. It calls a sub-algorithm for minimization on the current face. The algorithm visits the different faces of the box using a

strategy that will be described below. First we need some general definitions.

As in [33], let us divide the feasible set Ω into disjoint faces, as follows. For all $I \subset \{1, 2, \ldots, n, n+1, n+2, \ldots, 2n\}$, we define

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

The closure of F_I is denoted by $\bar{F}_I$. Let $[F_I]$ denote the smallest linear manifold that contains F_I, and S_I denote the subspace obtained by the parallel translation of $[F_I]$. For brevity, $-\nabla f(x)$ will be called *antigradient*.

Given $x \in F_I$, the orthogonal projection of $-\nabla f(x)$ on S_I will be called *internal antigradient* and denoted by $g_I(x)$. The *chopped antigradient* (see [23, 33]) $g_C(x)$ is defined for $x \in \Omega$ as follows

$$[g_C(x)]_i = \begin{cases} -\frac{\partial f}{\partial x_i}(x), & \text{if } \ x_i = \ell_i \ \text{ and } \ \frac{\partial f}{\partial x_i}(x) < 0, \\ -\frac{\partial f}{\partial x_i}(x), & \text{if } \ x_i = u_i \ \text{ and } \ \frac{\partial f}{\partial x_i}(x) > 0, \\ 0, & \text{otherwise,} \end{cases}$$

where $i = 1, \ldots, n$. Observe that $[g_C(x)]_i = 0$ if $\ell_i < x_i < u_i$.

Since $g_C(x) \perp S_I$, we have that

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

Denote $g_P(x) = g_I(x) + g_C(x)$. The vector $g_P(x)$ will be called *projected antigradient*. Note that $x \in \Omega$ is a stationary point of problem (1) if and only if $g_P(x) = 0$. In general, the mapping $g_P(x)$ is not continuous, nevertheless, $x^k \to x$ and $g_P(x^k) \to 0$ imply that $g_P(x) = 0$ (see [15]).

Given $x^k \in \Omega$, the sub-algorithm computes a new iterate x^{k+1}. We assume that the sub-algorithm has the following properties:

P1. $f(x^{k+1}) < f(x^k)$.
P2. If $x^k \in F_I$ then $x^{k+1} \in \bar{F}_I$.
P3. If $\{x^k, x^{k+1}, x^{k+2}, \ldots\} \subset F_I$ is a set of infinitely many iterates generated by the sub-algorithm, then $g_I(x^k) \to 0$.

Below we present our main model algorithm. The symbol $\|\cdot\|$ will denote the Euclidean vector norm throughout the paper, although in many cases, any other norm can be used instead.

Algorithm 2.1. Assume that $x^0 \in \Omega$ is an arbitrary initial point, $\eta \in (0, 1)$, $0 < \tau_{min} \le \tau_{max} < \infty$, $0 < \beta_{min} \le \beta_{max} < 1$ and $\theta \in (0, 1)$. Let F_I be the face that contains the current iterate x^k. The new iterate x^{k+1} is computed as follows.

Step 1. If $\|g_P(x^k)\| = 0$ (x^k is a stationary point), stop. If

$$\frac{\|g_C(x^k)\|}{\|g_P(x^k)\|} \geq \eta, \tag{2}$$

compute x^{k+1} at Step 2, else compute x^{k+1} using the sub-algorithm.

Step 2. Choose $\tau_k \in [\tau_{min}, \tau_{max}]$. Let α_{max} be the maximum value of α, for which $x^k + \alpha g_C(x^k) \in \Omega$. Set $\alpha = \min\{\tau_k, \alpha_{max}\}$. If

$$f(x^k + \alpha g_C(x^k)) \leq f(x^k) - \theta\alpha\|g_C(x^k)\|^2 \tag{3}$$

set $\alpha_k = \alpha$, $x^{k+1} = x^k + \alpha_k g_C(x^k)$ and finish the k-th iteration. Else, choose new value for α in the interval $[\beta_{min}\alpha, \beta_{max}\alpha]$ and repeat test (3).

The global convergence theory for this algorithm generalizes the one given in [5] for quadratic minimization.

We assume that $\nabla f(x)$ satisfies a Lipschitz condition: there exists $L > 0$ such that

$$\|\nabla f(y) - \nabla f(x)\| \leq L\|y - x\|, \quad \forall x, y \in \Omega.$$

This implies that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L}{2}\|y - x\|^2, \quad \forall x, y \in \Omega. \tag{4}$$

Here and below, $\langle a, b\rangle$ denotes the scalar product $a^T b$ in $\mathbb{R}^n$.

Let us prove global convergence for Algorithm 2.1.

Theorem 2.1. *Algorithm 2.1 is well defined, and at least one of the limit points the generated sequence is a stationary point for problem (1).*

Proof. Denote

$$K = \{k \in \mathbb{N} \mid \|g_C(x^k)\|/\|g_P(x^k)\| \geq \eta\}.$$

To prove that the algorithm is well defined, it is sufficient to show that, for all $k \in K$, condition (3) is satisfied after a finite number of reductions of α. Indeed, for all $\alpha \geq 0$, from (4) we have

$$f(x^k + \alpha g_C(x^k)) \leq f(x^k) - \alpha\|g_C(x^k)\|^2 + \frac{\alpha^2 L}{2}\|g_C(x^k)\|^2.$$

This implies that (3) holds for $\alpha \leq \frac{2(1-\theta)}{L}$. Therefore, the new iterate is well defined.

Moreover, the value of α accepted at Step 2 of Algorithm 2.1 is bounded below by

$$\bar{\alpha} = \min\{\tau_{min}, \frac{2(1-\theta)}{L}\beta_{min}\} > 0.$$

Hence, at Step 2 we have

$$f(x^k) - f(x^{k+1}) \geq \theta\bar{\alpha}\eta\|g_P(x^k)\|^2. \tag{5}$$

Since $f(x^{k+1}) \leq f(x^k)$ for all $k \in I\!N$ and since $f(x)$ is bounded below on Ω, (5) implies that either K is finite or

$$\sum_{k\in K} \|g_P(x^k)\|^2 < \infty. \tag{6}$$

In the infinite case, (6) implies that $g_P(x^k) \to 0$ for $k \in K$. Consequently, every limit point of $\{x^k\}_{k\in K}$ is a stationary point.

If K is finite, there exists $k_0 \in I\!N$ and a face F_I such that $x^k \in F_I$ for all $k \geq k_0$. Therefore, x^{k+1} is computed by the sub-algorithm for all $k \geq k_0$. Then, by the property P3, $\lim_{k\to\infty} \|g_I(x^k)\| = 0$. But, for all $k \geq k_0$, inequality (2) does not hold. Hence $\lim_{k\to\infty} \|g_P(x^k)\| = 0$. As before, this means that every limit point of $\{x^k\}$ is stationary. ∎

Recall that the stationary points of our problem are characterized by $g_P(x) = 0$. If x is a stationary point, such that $x_i = \ell_i$ (or $x_i = u_i$) and $\frac{\partial f}{\partial x_i} = 0$, we say that this point is *degenerate*. In the following theorem we prove that, if degenerate points do not exist, the algorithm identifies the active constraints at the limit points in a finite number of iterations.

Theorem 2.2. *Assume that all the stationary points of (1) are nondegenerate. Then, there exists $I \subset \{1, 2, \ldots, 2n\}$ and $k_0 \in I\!N$ such that $x^k \in F_I$ for all $k \geq k_0$. Moreover, all the limit points of the sequence $\{x^k\}$ belong to F_I and are stationary.*

Proof. We exclude from our consideration the trivial case, when Algorithm 2.1 terminates in a finite number of iterations. Let us prove first that Step 2 cannot be executed infinitely many times. Assume, by contradiction, that the set of iterates x^{k+1} computed at Step 2 is infinite. Then there exists a constraint which is abandoned infinitely many times. Without loss of generality, assume that this constraint is $x_i = \ell_i$, i.e. there exists an infinite set K such that, for all $k \in K$,

$$x_i^k = \ell_i, \;\; x_i^{k+1} > \ell_i, \tag{7}$$

$$\frac{\partial f}{\partial x_i}(x^k) < 0, \tag{8}$$

and x^{k+1} is computed at Step 2. Let x^* be a limit point of $\{x^k\}_{k\in K}$. By Theorem 2.1, x^* is a stationary point. From (7) and (8), we have $x_i^* = \ell_i$ and $\frac{\partial f}{\partial x_i}(x^*) \leq 0$. But since x^* is stationary, $\frac{\partial f}{\partial x_i}(x^*) \geq 0$. Hence x^* is degenerate,

which contradicts the theorem assumption. Thus, we proved that there exists $k_0 \in \mathbb{N}$ and $I \subset \{1, 2, \ldots, 2n\}$ such that $x^k \in F_I$ for all $k \geq k_0$. This implies that x^{k+1} is computed by the sub-algorithm for all $k \geq k_0$. Then, according to the property P3, $g_I(x^k) \to 0$. By continuity, this implies that

$$\frac{\partial f}{\partial x_i}(x^*) = 0$$

for all i such that $i \notin I$ and $n + i \notin I$. Since x^* is nondegenerate, this implies that $\ell_i < x_i^* < u_i$. Therefore, $x^* \in F_I$. ∎

3 Minimization within a given face

Algorithm 3.1, which is presented below, is one of the possible implementations of the sub-algorithm, which is used at Step 1 of Algorithm 2.1 for the minimization within a given face F_I. Given $x^k \in F_I$ (that violates (2)), a symmetric Hessian approximation $B^k \in \mathbb{R}^{n \times n}$ and a trust region radius δ_k, Algorithm 3.1 generates $x^{k+1} \in \bar{F}_I$. To simplify the notation, suppose that the face F_I is the interior of Ω. The extension to a general F_I is straightforward. In Algorithm 3.1, the CG method is applied to the quadratic subproblem

$$\text{Minimize } Q(p) \equiv \frac{1}{2}\langle p, B^k p\rangle + \langle \nabla f(x^k), p\rangle. \tag{9}$$

Like in [45], the regular CG iterations are interrupted, when the constraints

$$\|p\|_\infty \leq \delta_k, \quad \ell \leq x^k + p \leq u \tag{10}$$

are violated or when a direction of negative curvature is encountered. A TR approach is used to decide whether the generated trial point is good enough. We also follow the classical rules to modify the trust-region radius.

The sub-algorithm can be stated formally as follows.

Algorithm 3.1.
Step 1. Starting with $p^0 = 0$, apply the CG method to (9). This method generates a search direction d^j and a new iterate p^{j+1} at its jth iteration ($j = 0, 1, \ldots$). Interrupt this process in any of the following three cases:

Case 1: $\nabla Q(p^j) = 0$. In this case, set $p_{trial} = p^j$.

Case 2: $Q(p)$ tends to $-\infty$ along d^j. In this case, proceed as in Case 3, with p^{j+1} replaced by $p^j + Md^j$, where M is a large positive number.

Case 3: p^{j+1} violates at least one of the constraints (10). In this case, define p' as the projection of p^{j+1} on the region given by (10), and define p'' as the farthest

point from p^j among those belonging to the segment $[p^j, p^{j+1}]$ and satisfying (10). If $Q(p') \leq Q(p'')$, then set $p_{trial} = p'$, else set $p_{trial} = p''$.
Step 2. Set $x_{trial} = x^k + p_{trial}$, and compute the following predicted and actual reductions in function value:

$$\Delta_{pred} = -Q(p_{trial}), \quad \Delta_{act} = f(x^k) - f(x_{trial}).$$

If $\Delta_{act} < 0.1\Delta_{pred}$,	set $\delta_k = 0.5\|p_{trial}\|_\infty$ and go to Step 1.
If $0.1\Delta_{pred} \leq \Delta_{act} \leq \Delta_{pred}$,	set $x^{k+1} = x_{trial}$ and $\delta_{k+1} = \delta_k$.
If $\Delta_{act} > \Delta_{pred}$,	set $x^{k+1} = x_{trial}$ and $\delta_{k+1} = 3\delta_k$.

At Step 2, we can use different numbers in $(0, 1)$ instead of 0.1 and 0.5. Analogously, we can use any number greater than 1 instead 3. Our choice of these parameters was motivated by numerical experience.

The existing theoretical results concerning trust-region methods (see e.g. [16, 19]) can be applied to show that, under boundedness assumptions on $\|B^k\|$, Algorithm 3.1 is well defined and enjoys the properties P1-P3.

The Hessian approximation B^k, that will be introduced in the next section, is a limited-memory approximation. Since this approximation is a low-rank modification of a multiple of the identity matrix, the matrix B^k has a small number of different eigenvalues. Then, the CG method uses a small number of iterations to solve the quadratic problem (9) or to identify a negative curvature direction of the quadratic. Negative curvature information is accumulated in the Hessian approximations thanks to the multipoint symmetric secant approach.

4 Limited-memory multipoint symmetric secant approximations

Let us describe now how to employ the basic idea of the multipoint symmetric secant method for generating the matrices B^k.

Denote $s^k = x^{k+1} - x^k$, $y^k = g^{k+1} - g^k$. Consider a special case assuming that the sequence of n vectors $s^0, \ldots, s^{n-1}$ have been generated somehow, and that they are linearly independent. The ideal aim would be to construct a Hessian approximation $B^n \in I\!R^{n\times n}$ such that

$$(B^n)^T = B^n, \tag{11}$$

$$B^n s^i = y^i, \quad i = 0, \ldots, n-1. \tag{12}$$

In general, this is impossible, because the system (11)-(12) (whose unknowns are the entries of B^n) is overdetermined. The information about the symmetry of the

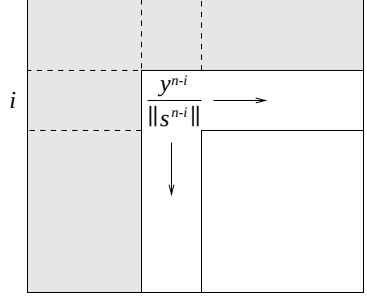

Figure 1: *Symmetric secant approximation of the Hessian matrix.*

Hessian matrix conflicts with the information carried by the pairs $\{s^i, y^i\}$. The idea of the sequential symmetric secant methods introduced in [9] is to release partially equations (12) in order to have B^n well defined. This can be accomplished in various ways. Uniqueness can be achieved by ranging the pairs $\{s^i, y^i\}$ according to the reliability of the information that they carry. For example, for $i > j$, one can consider $\{s^i, y^i\}$ as more reliable for the Hessian approximation than $\{s^j, y^j\}$. Therefore, in the process of constructing the Hessian approximation B^n, it is natural to use the pairs $\{s^i, y^i\}$ sequentially for $i = n-1, n-2, \ldots, 0$.

Suppose, for a moment, that each vector $s^{n-i}, i = 1, \ldots, n$, is parallel to the coordinate axis e^i. Then the first column and the first row of the Hessian matrix can be approximated by the standard finite-difference formula as $y^{n-1}/\|s^{n-1}\|$. The second column and row, in their parts outside the first column and row, are approximated by $y^{n-2}/\|s^{n-2}\|$, and so on. In order to fill the "nonfilled" part of the ith row and column, the components $y_j^{n-i}/\|s^{n-i}\|$, $j = i, \ldots, n$, are used (see Fig. 1).

In the general case of arbitrary vectors s^{n-i}, the space can be linearly transformed so that, in the new space, the vectors $\tilde{s}^{n-i}$ are parallel to the new coordinate axes $\tilde{e}^i$. Then the described approach can be used to approximate the Hessian matrix in the new space. After returning to the original space, we get the approximation

$$B^n = S^{-T}\mathrm{sym}(S^T Y)S^{-1}, \tag{13}$$

where $S = [s^{n-1}, s^{n-2}, \ldots, s^0]$, $Y = [y^{n-1}, y^{n-2}, \ldots, y^0] \in I\!R^{n\times n}$. For any matrix A, the symmetrization operation is defined as

$$(\mathrm{sym}A)_{ij} = \begin{cases} A_{ij}, & i \geq j, \\ A_{ji}, & \text{otherwise.} \end{cases}$$

Note that $B^n = f''$, if $f(x)$ is quadratic. If not, multipoint symmetric secant formula (13) gives a good approximation to $f''(x^n)$, provided that the matrix S is "safely" nonsingular (see [9]).

Let us compare the approximation (13) with the one $B^n = YS^{-1}$ given by the classic multipoint secant method (see [42]). In the new subspace, where $\tilde{s}^{k-i}$ is parallel to $\tilde{e}^i$, $i = 1, \ldots, n$, it is easy to see for each element of the approximations, how "fresh" is the information involved in its computation. Comparing these two

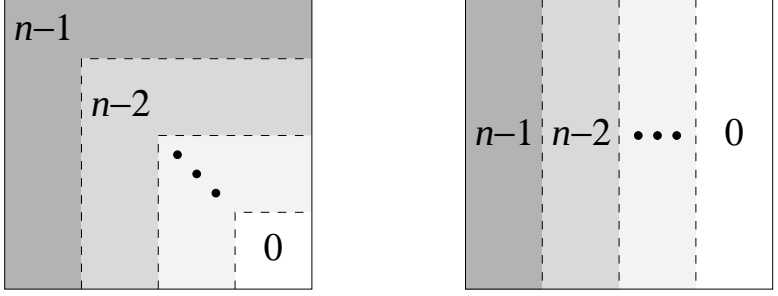

Figure 2: *The symmetric (left) and the classic (right) secant approximations with indication, for each element, the iteration at which its pair* $\{s, y\}$ *was computed.*

approximations, say, row by row (see Fig. 2), one can see that the symmetric one uses more "fresh" information than the classic one. An important property of (13) is that B^n can be obtained, for any initial $B^0 \in I\!R^{n\times n}$, as the result of n sequential updatings by the rank-two formula

$$B^{k+1} = B^k + \frac{(y^k - B^k s^k)(c^k)^T + c^k(y^k - B^k s^k)^T}{\langle s^k, c^k\rangle} - \frac{\langle y^k - B^k s^k, s^k\rangle c^k (c^k)^T}{\langle s^k, c^k\rangle^2}, \tag{14}$$

where c^k is any vector in $I\!R^n$, such that

$$\langle c^k, s^i\rangle = 0, \quad 0 \le i < k, \tag{15}$$

$$\langle c^k, s^k\rangle \neq 0. \tag{16}$$

The sequence $\{B^k\}_0^n$ is well defined by (14)-(16) in the sense that there is no break-down for all $k = 0, \ldots, n-1$. We assume, from now on, that B^0 is symmetric, although some of our further assertions do not require this assumption.

It can be easily shown by analogy with [12] that formulae (14)–(16) generate symmetric Hessian approximations that satisfy for all $k = 0, \ldots, n-1$ the following equations

$$(S^k)^T B^{k+1} S^k = \mathrm{sym}((S^k)^T Y^k), \tag{17}$$

$$s^T B^{k+1} S^k = s^T Y^k, \quad \forall s \perp S^k, \tag{18}$$

$$(S^k)^T B^{k+1} s = (Y^k)^T s, \quad \forall s \perp S^k, \tag{19}$$

where $S^k = [s^k, s^{k-1}, \ldots, s^0]$, $Y^k = [y^k, y^{k-1}, \ldots, y^0] \in I\!R^{n\times(k+1)}$. These equalities imply the secant equation

$$B^{k+1} s^k = y^k. \tag{20}$$

Note that the vector c^k, as well as B^{k+1}, are not uniquely defined by (15) and (16). Uniqueness can be obtained, if we assume that B^{k+1} is the solution of the following least-change problem:

$$\begin{array}{rl} \text{Minimize} & \|B - B^k\|_F, \\ \text{subject to} & (S^k)^T B S^k = \text{sym}((S^k)^T Y^k), \\ & s^T B S^k \quad = s^T Y^k, \quad \forall s \perp S^k, \\ & (S^k)^T B s \quad = (Y^k)^T s, \quad \forall s \perp S^k, \end{array} \tag{21}$$

where $\|\cdot\|_F$ is the Frobenius matrix norm, and B^k is supposed to satisfy equations similar to (17) and (18). The solution to this problem is unique, and it is given by formula (14) with

$$c^k = [I - S^{k-1}((S^{k-1})^T S^{k-1})^{-1}(S^{k-1})^T]s^k.$$

This means that the sequence $\{c^k\}_0^{n-1}$ can be obtained, e.g., by the Gram-Schmidt orthogonalization process applied to the sequence $\{s^k\}_0^{n-1}$. Denoting

$$C^k = \left[\frac{c_0}{\|c_0\|}, \ldots, \frac{c_k}{\|c_k\|}\right] \in I\!R^{n\times(k+1)},$$

we see that $(C^k)^T C^k = I$ and

$$c^k = [I - C^{k-1}(C^{k-1})^T]s^k. \tag{22}$$

This choice of c^k ensures that the equation

$$s^T B^k s = s^T B^0 s \quad \forall s \perp S^{k-1} \tag{23}$$

holds for all $k = 1, \ldots, n$. Note that the sequence of approximations B^k is uniquely defined by (17)-(19) and (23). Our limited-memory approach will be essentially based on this property.

In limited-memory methods, the Hessian matrix is approximated by a low-rank modification of a simple matrix B^0. In the next theorem, we present the multipoint symmetric secant approximations in the form that will be useful for implementation. For simplicity, the upper indices of S^k and Y^k will be omitted.

Theorem 4.1. *Let $S = [s^k, s^{k-1}, \ldots, s^0] \in I\!R^{n\times(k+1)}$ be a full-rank matrix. Suppose that the matrices $B^1, \ldots, B^{k+1}$ are generated by (14) and (22). Then, for any $B^0 \in I\!R^{n\times n}$,*

$$\begin{array}{rcl} B^{k+1} & = & (I - S(S^TS)^{-1}S^T)B^0(I - S(S^TS)^{-1}S^T) \\ & + & \left[\begin{array}{cc} S & Y \end{array}\right] \left[\begin{array}{cc} -(S^TS)^{-1}\text{sym}(Y^TS)(S^TS)^{-1} & (S^TS)^{-1} \\ (S^TS)^{-1} & 0 \end{array}\right] \left[\begin{array}{c} S^T \\ Y^T \end{array}\right], \end{array} \tag{24}$$

where $Y = [y^k, y^{k-1}, \ldots, y^0] \in \mathbb{R}^{n\times(k+1)}$.

Proof. Let $S_\perp \in \mathbb{R}^{n\times(n-k-1)}$ be any matrix such that

$$S_\perp^T S_\perp = I \quad \text{and} \quad S_\perp^T S = 0.$$

Then, equations (17)–(19) and (23) can be written as

$$\begin{bmatrix} S^T \\ S_\perp^T \end{bmatrix} B^{k+1} \begin{bmatrix} S & S_\perp \end{bmatrix} = \begin{bmatrix} \mathrm{sym}(S^T Y) & Y^T S_\perp \\ S_\perp^T Y & S_\perp^T B^0 S_\perp \end{bmatrix}. \tag{25}$$

By the hypothesis, the matrix $[S\ S_\perp] \in \mathbb{R}^{n\times n}$ is nonsingular. Clearly,

$$\begin{bmatrix} S & S_\perp \end{bmatrix}^{-1} = \begin{bmatrix} (S^T S)^{-1} S \\ S_\perp^T \end{bmatrix}.$$

Therefore, using

$$S_\perp S_\perp^T = I - S(S^T S)^{-1} S^T,$$
$$\mathrm{sym}(Y^T S) - S^T Y - Y^T S = -\mathrm{sym}(S^T Y),$$

formula (24) can be easily derived from (25). ∎

In limited-memory methods, the initial Hessian approximation is usually chosen as $B^0 = \gamma I$, where the positive scalar γ may change from iteration to iteration of the main algorithm.

In our approximation, we choose a small collection of vector pairs $\{s^i, y^i\}$ among those recently generated by the main algorithm. The number of these pairs, denoted by m, may also change from one iteration to another. Then we compute the matrices $S, Y \in \mathbb{R}^{n\times m}$ and we apply formula (24) with $B^0 = \gamma I$. This gives

$$\begin{aligned} B &= \gamma I \\ &+ \begin{bmatrix} S & Y \end{bmatrix} \begin{bmatrix} -W\mathrm{sym}(Y^T S)W - \gamma W & W \\ W & 0 \end{bmatrix} \begin{bmatrix} S^T \\ Y^T \end{bmatrix}, \end{aligned} \tag{26}$$

where $W = (S^T S)^{-1} \in \mathbb{R}^{m\times m}$. The size of the middle matrix is $2m \times 2m$. Since $m \ll n$, the matrix B is a low-rank correction of γI. This is the most essential property of the limited-memory methods. Therefore, the number of different eigenvalues of the Hessian approximation is small and, so, the CG method must converge in a small number of iterations.

Our limited-memory formulae (24) and (26) differ from the conventional ones, because they are based on different quasi-Newton methods. As in other limited-memory approaches, the Hessian approximation in our version is not stored explicitly. Instead, the smaller matrices $S, Y \in \mathbb{R}^{n\times m}$, $S^T S, Y^T S \in \mathbb{R}^{m\times m}$ are

stored and updated. The products of the form Bv and $u^T Bv$, are be computed by formula (26) using only the stored matrices.

If the vectors s^i used in (26) are linearly dependent, the matrix $S^T S$ is singular. Even if this is not the case, if the vectors are "almost" linearly dependent, the Hessian approximation may be poor. This is typical in sequential multipoint secant approximations. To avoid instability we use only a subset of the available vectors in (26). As in [9, 10, 11, 12], the vectors selected for this subset must be *safely linearly independent* in some sense. The stable methods [11, 12] enjoy superlinear convergence due to the fact that they use only safely linearly independent vectors s^i for their multipoint symmetric secant approximations. In the next section, we introduce a measure of linear independence. When this measure is above a fixed threshold value $\sigma \in (0, 1]$, we consider that the vectors are safely linearly independent.

Suppose B^k is the Hessian approximation at the kth iteration of the main algorithm. We limit the maximal number of the vector pairs $\{s^i, y^i\}$ used in constructing B^k by a parameter $m_1 \ll n$. Another parameter, $m_2 \geq m_1$, prevents from using the pairs $\{s^i, y^i\}$ with $i < k - m_2$ (we call such pairs *old*, in contrast to the other pairs that we call *recent*). A general scheme of our limited-memory algorithm can be presented as follows.

Algorithm 4.1. Given $m_1 \ll n$, $m_2 \geq m_1$, $\nu \in (0, 1)$, $\gamma_k > 0$ and a set of recent vector pairs $\{s^i, y^i\}$, execute the following steps.
Step 1. From the given set of recently computed vectors s^i choose a subset of at most m_1 vectors such that, first, s^{k-1} is included in the subset, second, the vectors in the subset are safely linearly independent. Use all s^i from the subset as columns (preferably in decreasing order of i) for composing the matrix S. Compose the matrix Y accordingly.
Step 2. Construct B^k by formula (26) with $\gamma = \gamma_k$.

Note that Step 1 can be implemented in several ways. The simplest choice that could meet all the requirements of Step 1 would be to compose S of just one column s^{k-1}. In this case, formula (26) is equivalent to the Powell-symmetric-Broyden update [22] of the matrix $B^0 = \gamma_k I$.

In our algorithm, we set initially $S = [s^{k-1}]$, and then we check, in decreasing order of the iteration number i, whether the safe linear independence of the columns of S is preserved after adding s^i as a new column. If this is the case, S is enlarged by adding s^i as its new last column. We stop after checking all s^i from the given set of "recent" vectors. Our implementation of this approach is discussed in the following section.

5 Implementation features

In this section we address some implementation issues concerning the algorithms presented in the previous sections of this paper.

5.1 Updating of S and B at the k-th iteration

After the computation of x^{k+1}, a new vector pair $\{s^k, y^k\}$ is available. We check whether the vector s^k and the columns of S^{k-1} are safely linearly independent. If so, we simply add s^k to S^{k-1} as a new column so that $S^k = [s^k\ S^{k-1}]$. Otherwise, we compose S^k with s^k and of some columns of S^{k-1} in such a way that the columns of S^k are safely linearly independent.

To maintain safe linear independence of the columns of S, we use and update the QR decomposition of the matrix $\bar{S}$. This matrix is such that $\bar{S}P = S$, where P is a permutation matrix which ensures that the columns s^i in S are ordered in decreasing order of the iteration number i. Thus, we assume that the following matrices are available from the previous iteration: the orthogonal matrix $Q^{k-1} \in I\!R^{n\times m}$, the upper-triangular matrix $R^{k-1} \in I\!R^{m\times m}$ and the permutation matrix $P^{k-1} \in I\!R^{m\times m}$, such that the matrix $S^{k-1} = Q^{k-1}R^{k-1}P^{k-1}$ has the desired ordering of columns. For $k = 0$, our initialization of S^{k-1} corresponds to the choice $m = 0$. To simplify the linear algebra involved, we do the same at the subsequent iterations, whenever the current face changes.

Our criterion of safe linear independence is based on the following definition. Given $\sigma \in (0, 1]$, a matrix $A = [a^1, \ldots, a^m] \in I\!R^{n\times m}$ with $m \le n$ is said to be *σ-regular*, if for all $i = 1, \ldots, m$,

$$|\sin\varphi_i| \ge \sigma, \tag{27}$$

where φ_i is the angle between the column a^i and the subspace generated by the preceding columns $a^1, \ldots, a^{i-1}$. Note that the column lengths $\|a^i\|$ are not essential in this definition.

An additional point to emphasize is that, if the matrix R in the QR decomposition of A is available, the left-hand side of inequality (27) can be easily computed by the formula

$$|\sin\varphi_i| = R_{ii}/\|a_i\|.$$

The outlined updating of $\bar{S}$ and of its QR decomposition is presented below by Algorithm 5.1. For simplicity, we use the notations $s_c = s^k$, $\bar{S}_c = \bar{S}^{k-1}$, $Q_c = Q^{k-1}$, $R_c = R^{k-1}$ and $P_c = P^{k-1}$ for the input variables with the subscript c standing for "current", and the notations $\bar{S} = \bar{S}^k$, $Q = Q^k$, $R = R^k$ and $P = P^k$ for the output variables. The dimension m is an input-output variable.

Algorithm 5.1. Assume that $m_1 \ll n$, $m_2 \geq m_1$, $m \leq m_1$, $\sigma \in (0,1)$, $s_c \in I\!R^n$ and that $\bar{S}_c \in I\!R^{n\times m}$ is a σ-regular matrix composed of some recent vectors s^i and of at most one old vector. Moreover, assume that its QR-factors are $Q_c \in I\!R^{n\times m}$ and $R_c \in I\!R^{m\times m}$, and $P_c \in I\!R^{m\times m}$ is a permutation matrix such that the columns s^i of $S_c \equiv \bar{S}_c P_c$ are ordered in decreasing order of the iteration number i.

Step 1. If $\bar{S}_c$ has an old vector in the last column then set $m \leftarrow m-1$, exclude the last columns in $\bar{S}_c$ and Q_c and exclude both the last columns and the last rows in R_c and P_c.

Step 2. If $(m < m_1)$ and (there is no old vector in $\bar{S}_c$) then:
Set $\bar{r} = Q_c^T s_c$, $q = s_c - Q_c\bar{r}$, $r = \|q\|$ and $q = q/r$.
If $r > \sigma\|s_c\|$ then
set $m = m+1$, $\bar{S} = [\bar{S}_c \; s_c]$, $Q = [Q_c \; q]$,
$R = \begin{bmatrix} R_c & \bar{r} \\ 0 & r \end{bmatrix}$, $P = \begin{bmatrix} 0 & P_c \\ 1 & 0 \end{bmatrix}$ and stop.

Step 3. Set $m = 1$, $\bar{S} = [s_c]$, $Q = [s_c/\|s_c\|]$, $R = [\|s_c\|]$, $P = [1]$.

Step 4. Checking one by one the vectors s^i that compose the columns of $\bar{S}_c$ in decreasing order of i, while $i > k - m_2$ and $m < m_1$, do:
Set $\bar{r} = Q^T s^i$, $q = s^i - Q\bar{r}$, $r = \|q\|$ and $q = q/r$.
If $r > \sigma\|s^i\|$ then
set $m = m+1$, $\bar{S} = [\bar{S} \; s^i]$, $Q = [Q \; q]$,
$R = \begin{bmatrix} R & \bar{r} \\ 0 & r \end{bmatrix}$, $P = \begin{bmatrix} P & 0 \\ 0 & 1 \end{bmatrix}$.

One can see that the output matrix $\bar{S}$ is σ-regular, and that its column vectors s^i are not old. Moreover, the perturbation matrix

$$P = \left[\begin{array}{ccc|ccc} & & & 1 & & 0 \\ & 0 & & & \ddots & \\ & & & 0 & & 1 \\ \hline 0 & & 1 & & & \\ & \iddots & & & 0 & \\ 1 & & 0 & & & \end{array}\right]$$

produces the desired ordering of columns in the matrix $S = \bar{S}P$.

Remark

Our process of updating the QR decomposition can be viewed as a sort of Gram-Schmidt orthogonalization. Numerical stability can be significantly improved with the use of the approaches discussed, e.g., in [7, 20, 36].

Having available the matrix S, we compute the corresponding matrix Y and, if required, construct the Hessian approximation by formula (26), in which we set $W = P^T R^{-1} R^{-T} P$.

Note that the Q-factor is not involved in this Hessian approximation. Therefore, some savings, both in the computational costs and in the memory requirements, can be obtained if, as in [35], we avoid the computation of the matrix Q in Algorithm 5.1. This means that all the ocurrences of products of the form $Q_c^T v$ and $Q^T v$ should be replaced by $R_c^{-T} \bar{S}_c^T v$ and $R^{-T} \bar{S}^T v$, respectively. Since in Algorithm 5.1 the vector v is either s_c or a column of $\bar{S}_c$, the major computations involve the small matrices $\bar{S}_c^T \bar{S}_c, R_c \in I\!R^{m \times m}$ and the small vector $\bar{S}_c^T s_c \in I\!R^m$. Since for $m \ll n$ the major cost of the implicit Hessian approximation B^{k+1} by formula (26) is determined by the computation of the two products $S^{k-1} s^k$ and $S^{k-1} y^k$, the overall computational cost of this approximation can be estimated as $2mn$ flops. The outlined approach is expected to improve in the future our implementation of Algorithm 4.1.

5.2 Computation of α at Step 2 of Algorithm 2.1

Recall that in Algorithm 2.1 we define the steplength α in the chopped direction $g_C(x_k)$ as the minimum between τ_k and α_{max}, where $\tau_k \in [\tau_{min}, \tau_{max}]$. To take into account second order information we adopted the spectral choice (see [40],[43]):

$$\tau_k = \max\left(\tau_{\min}, \min\left(\tau_{\max}, \frac{\langle s^k, s^k \rangle}{\langle s^k, y^k \rangle}\right)\right) \tag{28}$$

5.3 Initial Hessian approximation

For constructing the matrix B^k, it is necessary to specify in (26) the value of γ, which is associated with the initial Hessian approximation $B^0 = \gamma I$. For $k = 0$, following [22], we set $\gamma = |f(x^0)|$. If γ is less than a tolerance ε, we set $\gamma = 1$. For $k > 0$, we use the spectral choice (28) $\gamma = 1/\tau_k$.

6 Numerical experiments

Algorithm 2.1, along with sub-algorithm 3.1 and the multipoint symmetric secant approximations B^k, define an implementable algorithm for box-constrained minimization of smooth functions. Each particular implementation is determined by the choice of several parameters. Using a small set of test problems, we arrived to the following default options:

- $\varepsilon = 10^{-5}$, tolerance for 2-norm of projected gradient g_P (Algorithm 2.1).
- $\eta = 0.9$, tolerance in the test to leave the face (Algorithm 2.1).
- $\tau_{min} = 10^{-3}$, $\tau_{max} = 10^3$, bounds for the spectral parameter τ_k (Algorithm 2.1).
- $\theta = 10^{-4}$, line search parameter (Algorithm 2.1).
- $\beta_{min} = 0.1$, $\beta_{max} = 0.9$, parameters specifying the decrease interval for α (Algorithm 2.1).
- $M = 10$, multiplier for increasing search direction. (Case 2 in Algorithm 3.1.)
- $\delta_0 = 1.0$, initial trust region radius (Algorithm 3.1).
- $\sigma = 0.01$, threshold in the σ-regularity test (Algorithm 5.1).
- $m_1 = 5$, maximal number of columns in the matrix S (Algorithm 5.1).
- $m_2 = 7$, parameter protecting from using too old vectors s^i (Algorithm 5.1).

The resulting code, named BSS, was compared with the code LANCELOT [16, 18] on a set of 20 bound constrained problems from the CUTE collection [8]. BSS and LANCELOT were implemented in Fortran 77 and compiled with f77 compiler. All the experiments were done with the `-O` optimization compiler option on a SUN UltraSPARC1 station.

LANCELOT was used with the following default options:

- `bandsolver-preconditioned-cg-solver-used 5`
- `exact-Cauchy-point-required`
- `solve-bqp-accurately`
- `gradient-tolerance 1.0D-05`

- `constraints-tolerance 1.0D-06`
- `maximum-number-of-iterations 5000`

The numerical results are presented by Table 1. We list the name of the problem, the type of the objective function ("`q`", "`ssq`" and "`o`" stand for quadratic, sum of squares and other, respectively), the number of variables, the number of iterations (*It*) and the CPU time (in seconds) for BSS and LANCELOT. For LANCELOT, the results are presented for the three options:

- `exact-second-derivatives-used` (LAN(1))
- `bfgs-approximation-used` (LAN(2))
- `sr1-approximation-used` (LAN(3))

Problem	Type	n	BSS		LAN(1)		LAN(2)		LAN(3)	
			It	Time	It	Time	It	Time	It	Time
BQPGABIM	q	50	34	0.03	4	0.04	1272	10.71	9	0.11
BQPGASIM	q	50	22	0.02	3	0.04	5000	46.82	10	0.13
DECONVB	ssq	61	724	3.00	14	0.36	41	0.77	14	0.46
HARKERP2	q	100	113	0.37	8	0.94	8	0.95	8	0.95
HS110	ssq	100	2	0.01	1	0.04	1	0.04	1	0.04
S368	o	100	26	3.44	8	3.14	$^{(*)}$5000	2158.73	52	29.87
EXPLIN	o	500	77	0.09	11	0.45	16	0.47	2016	0.48
EXPLIN2	o	500	236	0.66	13	0.48	18	0.51	18	0.52
EXPQUAD	o	500	4284	64.63	792	105.74	1914	32.96	495	30.99
QRTQUAD	o	500	2378	19.72	859	239.15	54	8.34	527	87.13
CVXBQP1	q	10000	15	1.23	1	4.76	1	4.97	1	4.87
HATFLDC	ssq	10000	396	184.74	4	4.63	4	5.01	4	4.76
MCCORMCK	o	10000	17	5.87	4	4.73	6	6.34	6	6.42
NONSCOMP	ssq	10000	23	3.05	8	6.63	5	5.48	8	6.27
NCVXBQP1	q	10000	5	0.66	4	8.31	4	8.80	4	8.55
NCVXBQP2	q	10000	60	8.96	5	11.94	5	12.75	5	11.50
PENTDI	q	10000	3	0.37	1	3.05	20	6.40	3	3.17
PROBPENL	o	10000	3	0.61	1	27.63	2	51.45	2	51.73
QUDLIN	q	10001	53	2.51	4	8.30	4	8.01	4	7.81
TORSION6	q	14884	292	123.73	8	17.29	9	16.95	9	17.02

Table 1. *Performance of BSS versus LANCELOT.*

BSS and LANCELOT found the same solutions, except the problem S368, marked off by $^{(*)}$, for which the maximum number of iterations was reached by LAN(2).

The reported results were obtained for $\eta = 0.9$. This is a rather conservative strategy that worked better than the "greedy" ones, which correspond to small values of η. This means that, in general, it is worthwhile to stay in the current

face, exploiting the quadratic model, instead of trying to change frequently the active constraints.

We used different values of m_1 in the range $[3, 20]$ and we defined $m_2 = 1.5 * m_1$. The results were not very sensitive to the choice of m_1. This behavior is due to the strategy of resetting the matrix S when change of faces occur.

Note that too large values of σ (say, above 0.1), would be very restrictive in the sense that too many vectors s^i would be rejected in the process of Hessian approximation. The numerical results were not too different for the values $\sigma = 0.1, 0.01$ and 0.001, mainly because of the resetting strategy. Since $\sigma = 0.01$ produced slightly better results, we adopted this parameter for our implementation.

These tests do not aim to establish the superiority of one algorithm over he other but only to assess the reliability of BSS. For this reason, we ran LANCELOT only with its default parameters.

In general, the number of iterations of BSS is larger than the one of the code LANCELOT with second derivatives, but this is not reflected in the computer time. The reason is that our subproblems are very cheap due to the low-rank character of the Hessian approximations, and so, the number of the CG-iterations in the trust-region subproblems is very small.

7 Conclusions

Active set methods are among the most traditional tools of constrained optimization. Their appeal comes from the fact that they allow the algorithmic designer to take full advantage of previously developed unconstrained optimization techniques. As far as new ideas in unconstrained minimization continue to be introduced, the implementation of active set methods based on those ideas is a natural task.

The unconstrained optimization technique exploited in this paper is the memoryless multipoint symmetric secant scheme, which is related to quasi-Newton methods. The fulfillment of several secant equations within a given face (or subspace) usually ensures Newton-like properties of the search directions generated on that face. On the other hand, since the approximate Hessians so far generated are not necessarily positive definite, a trust-region strategy for global convergence is necessary. A small number of low-rank corrections guarantees that the Hessian approximations possess a small number of different eigenvalues and, so, the CG method is efficient for solving the resulting quadratic subproblems.

The comparison with LANCELOT reveals that the method introduced here is reliable, and that it is able to compete with well established optimization

solvers. It is interesting to observe that the new method worked very well in problems where the performance of LANCELOT was rather poor (NCVXBQP1, PENTDI, PROBPENL, QUDLIN), whereas LANCELOT was more efficient in others (HATFLDC,TORSION6). This fact indicates that the trust-region strategy of LANCELOT and of other box-constrained solvers is complementary to the active-set strategy in the sense that difficult problems for one of them are relatively easy for the other.

Box-constrained minimization subroutines are usually employed in the implementation of augmented Lagrangian algorithms for general nonlinear programming (see [17, 38]). We plan to adapt our method for that purpose in the near future. Moreover, as it was mentioned in Sub-section 5.1, we also plan to employ some ideas of [35] to improve the computational efficiency of our limited-memory multipoint symmetric secant approximation of the Hessian. Note that in the current implementation we reset the matrix S when the working face changes. With an affordable complication of the linear algebra involved, this resetting procedure can be avoided. Probably, this will contribute to the overall improvement of the efficiency of the method.

Acknowledgements

The authors are indebted to two anonymous referees for useful comments.

References

[1] R. Andreani, A. Friedlander and J. M. Martínez, On the solution of finite-dimensional variational inequalities using smooth optimization with simple bounds, Journal on Optimization Theory and Applications 94 (1997) 635-657.

[2] R. Andreani and J. M. Martínez, On the solution of the extended linear complementarity problem, Linear Algebra and its Applications 281 (1998) 247-257.

[3] R. Andreani and J. M. Martínez, On the reformulation of nonlinear complementarity problems using the Fischer-Burmeister function, Applied Mathematics Letters 12 (1999) 7-12.

[4] R. Andreani and J.M. Martínez, Reformulation of variational inequalities on a simplex and the compactification of complementarity problems, SIAM Journal on Optimization 10 (2000) 878-895.

[5] R.H. Bielschowsky, A. Friedlander, F.M. Gomes, J.M. Martínez and M. Raydan, An adaptive algorithm for bound constrained quadratic minimization, Investigación Operativa 7 (1998) 67-102.

[6] A.G. Biryukov, On the difference-approximation approach to the solution of systems of nonlinear equations, Soviet Math. Dokl. 17(1983) 660-664.

[7] A. Bjorck, *Numerical Methods for Least-squares Problems*, SIAM (1996).

[8] I. Bongartz, A.R. Conn, N.I.M. Gould and Ph.L. Toint, CUTE: constrained and unconstrained testing environment, ACM Transactions on Mathematical Software 21 (1995) 123-160.

[9] O.P. Burdakov, Methods of secant type for systems of equations with symmetric Jacobian matrix, Numerical Functional Analysis and Optimization 6 (1983) 183-195.

[10] O.P. Burdakov, Stable versions of the secant method for solving systems of equations, U.S.S.R. Comput. Math. and Math. Phys. 23 (1983) 1-10.

[11] O.P. Burdakov, On superlinear convergence of some stable variants of the secant method, ZAMM 66 (1986) 615-622.

[12] O.P. Burdakov, Stable symmetric secant methods with restarts, Cybernetics 27 (1991) 390-396.

[13] R.H. Byrd, P. Lu, J. Nocedal and C. Zhu, A limited memory algorithm for bound constrained minimization, SIAM Journal on Scientific Computing 16 (1995) 1190-1208.

[14] R.H. Byrd, J. Nocedal and R.B. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, Mathematical Programming 63 (1994) 129-156.

[15] P.H. Calamai and J.J. Moré, Projected gradient methods for linearly constrained problems, Math. Progr.39 (1987) 93-116.

[16] 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 Journal on Numerical Analysis 25 (1988) 433-460.

[17] 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 Journal on Numerical Analysis 28 (1991) 545-572.

[18] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, Springer Verlag (1992).

[19] A.R. Conn, N.I.M. Gould and Ph.L. Toint, *Trust-Region Methods*, SIAM-MPS (2000).

[20] A. Dax, A modified Gram-Schmidt algorithm with iterative orthogonalization and column pivoting, Linear Algebra and its Applications 310 (2000) 25-42.

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

[22] J.E. Dennis and R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall (1983).

[23] Z. Dostál, Box constrained quadratic programming with proportioning and projections, SIAM Journal on Optimization 7 (1997) 871-887.

[24] Z. Dostál, A. Friedlander, S.A. Santos and J. Malík, Analysis of semicoercive contact problems using symmetric BEM and augmented Lagrangians, Engineering Analysis with Boundary Elements 18, (1997) 195-201.

[25] Z. Dostál, A. Friedlander and S.A. Santos, Solution of contact problems using subroutine BOX-QUACAN, Investigación Operativa 7 (1997) 13-22.

[26] Z. Dostál, A. Friedlander and S.A. Santos, Analysis of block structures by augmented Lagrangians with adaptive precision control, in: *Proceedings of GEOMECHANICS'96*, ed. Z. Rakowski, A. A. Balkema, Rotterdam, (1997) 175-180.

[27] Z. Dostál, A. Friedlander and S.A. Santos, Solution of coercive and semicoercive contact problems by FETI domain decomposition, in: *Contemporary Mathematics 218 - The Tenth International Conference on Domain Decomposition Methods*, eds. J. Mandel, C. Farhat and X. Cai, Boulder (CO) (1998) 82-93.

[28] Z. Dostál, A. Friedlander and S.A. Santos, Augmented Lagrangians with adaptive precision control for quadratic programming with equality constraints, Computational Optimization and Applications 14 (1999) 1-17.

[29] Z. Dostál, F. A. M. Gomes and S.A. Santos, Duality-based domain decomposition with natural coarse-space for variational inequalities, Journal of Computational and Applied Mathematics 126 (2000) 397-415.

[30] F. Facchinei, J.J. Júdice and J. Soares, An active set Newton algorithm for large scale nonlinear programs with box constraints, SIAM Journal on Optimization 8 (1998) 158-186.

[31] A. Friedlander and J.M. Martínez, On the numerical solution of bound constrained optimization problems, RAIRO Operations Research 23 (1989) 319-341.

[32] A. Friedlander and J.M. Martínez, New algorithms for maximization of concave functions with box constraints, RAIRO Operations Research 26 (1992) 209-236.

[33] A. Friedlander and J.M. Martínez, On the maximization of a concave quadratic function with box constraints, SIAM Journal on Optimization 4 (1994) 177-192.

[34] A. Friedlander, J.M. Martínez and S.A. Santos, A new trust region algorithm for bound constrained minimization, Applied Mathematics and Optimization 30 (1994) 235-266.

[35] P.E. Gill and M.W. Leonard, Limited-memory reduced-Hessian methods for large-scale unconstrained optimization, Report NA 97-1, Department of Mathematics, University of California, San Diego (2000).

[36] G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 1989.

[37] C.J. Lin and J.J. Moré, Newton's method for large bound-constrained optimization problems, SIAM Journal on Optimization 9 (1999) 1100-1127.

[38] N. Krejić, J.M. Martínez, M.P. Mello and E.A. Pilotta, Validation of an augmented Lagrangian algorithm with a Gauss-Newton Hessian approximation using a set of hard-spheres problems, Computational Optimization and Applications 16 (2000) 247-263.

[39] J.M. Martínez, Three new Algorithms based on the sequential secant method, BIT 19 (1979) 236-243.

[40] J.M. Martínez, E.A. Pilotta, M. Raydan, Spectral gradient method for linearly constrained optimization, Technical Report 22/99, IMECC, University of Campinas (UNICAMP), Campinas, Brazil (1999).

[41] J.J. Moré and D.J. Thuente, Line search algorithms with guaranteed sufficient decrease, ACM Transactions on Mathematical Software 20 (1994) 286-307.

[42] J.M. Ortega and W. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, 1970.

[43] M. Raydan, On the Barzilai and Borwein choice of steplength for the gradient method, SIAM Journal on Optimization 7 (1993) 26-33.

[44] R.B. Schnabel, Quasi-Newton methods using multiple secant equations, Technical report CU-CS-247-83, Department of Computer Science, University of Colorado, Boulder, CO (1983).

[45] T. Steihaug, The conjugate gradient method and trust regions in large scale optimization, SIAM Journal on Numerical Analysis, 20 (1983) 626-637.