Applied Numerical Mathematics 112 (2017) 51–64

Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum

Non-monotone algorithm for minimization on arbitrary domains with applications to large-scale orthogonal Procrustes problem

J.B. Francisco [1], F.S. Viloche Bazán [2], M. Weber Mendonça [*,1]

Department of Mathematics, Federal University of Santa Catarina, 88040-900, Florianópolis, SC, Brazil

ARTICLE INFO

Article history:
Received 29 September 2015
Received in revised form 2 July 2016
Accepted 30 September 2016
Available online 5 October 2016

Keywords:
Non-monotone algorithm
Arbitrary domains
Trust-region method
Large-scale orthogonal Procrustes problem

ABSTRACT

This paper concerns a non-monotone algorithm for minimizing differentiable functions on closed sets. A general numerical scheme is proposed which combines a regularization/trust-region framework with a non-monotone strategy. Global convergence to stationary points is proved under usual assumptions. Numerical experiments for a particular version of the general algorithm are reported. In addition, a promising numerical scheme for medium/large-scale orthogonal Procrustes problem is also proposed and numerically illustrated.

1. Introduction

Many applications from science and engineering require the solution of optimization problems for which the objective function has to satisfy a number of constraints imposed by physical reasons. A great deal of work has been done on algorithms which generate sequences of iterates (feasible or not) hopefully approximating an optimal solution to the optimization problem. Global optima are hard to obtain in general (except in special cases, such as convex programming or some restricted least square problems). Hence, instead of looking for global optimization algorithms, one very often develops algorithms capable of generating sequences converging to local minima (or even to stationary points). In this paper, we propose a non-monotone globally convergent algorithm for solving the problem,

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{s.t.} \quad & x \in \Gamma, \end{aligned} \tag{1}$$

where $\Gamma \subseteq \mathbb{R}^n$ is a closed set and the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable on some open convex set $\widehat{\Gamma} \supseteq \Gamma$. Usually, Γ is characterized by equality and inequality constraints of the form $\Gamma = \{x \in \mathbb{R}^n \mid h_i(x) = 0,\ i = 1, \ldots, m \text{ and } c_j(x) \le 0,\ j = 1, \ldots, p\}$ where h_i, c_i are differentiable functions in $\mathbb{R}^n$.

Non-monotone optimization schemes are interesting as they allow the iterates to jump through several basins of attraction, so that this class of methods is meaningful in problems with local minima lying in a narrow valley and consequently with a higher expectation for attaining a global minimum. Furthermore, since non-monotone conditions are less restrictive

* Corresponding author.
E-mail addresses: juliano.francisco@ufsc.br (J.B. Francisco), fermin.bazan@ufsc.br (F.S. Viloche Bazán), melissa.mendonca@ufsc.br (M. Weber Mendonça).
[1] These authors were supported by CNPq, Brazil, grant 479729/2011-5.
[2] This author was supported by CNPq, Brazil, grant 307476/2014-6.

http://dx.doi.org/10.1016/j.apnum.2016.09.018
0168-9274/© 2016 IMACS. Published by Elsevier B.V. All rights reserved.

in the step length than the monotone ones, these methods are interesting to globalize optimization methods with fast local convergence [6,19,26,34].

As for the algorithm proposed in this work, given a current point $x_k \in \Gamma$, a proper $\rho_k > 0$ and symmetric matrices A_k, B_k (with A_k symmetric positive definite), we minimize the quadratic model $Q_k(x) \equiv \langle \nabla f(x_k), x - x_k\rangle + 1/2(x-x_k)^T(B_k + \rho_k A_k)(x-x_k)$ over the subset Γ and then take the computed minimizer as the next iterate x_{k+1}, i.e., the point x_{k+1} is computed by solving the subproblem

$$\begin{aligned} \text{minimize} \quad & Q_k(x) \equiv \nabla f(x_k)^T(x-x_k) + \tfrac{1}{2}(x-x_k)^T(B_k+\rho_k A_k)(x-x_k) \\ \text{s.t.} \quad & x \in \Gamma. \end{aligned} \tag{2}$$

In this formulation, ρ_k acts like a regularization parameter, that is, it controls the stepsize, and so it plays a crucial role in the convergence of the generated sequence: if ρ_k is too large it can lead to slow convergence, whereas ρ_k too small can result in divergence. Therefore, it is convenient to choose ρ_k small enough when x_k is close to the solution in a way that Q_k of (2) is a more reliable quadratic model for f around x_k. In such a case, the local convergence can be accelerated with a suitable choice for B_k (e.g., $B_k \approx \nabla^2 f(x_k)$). In our algorithm, this trade-off is adjusted gradually in such a way that convergence to a stationary point is always guaranteed. Recently, several papers have appeared highlighting the effectiveness of non-monotone sufficient decrease conditions [6,9,10,12,13,18,19,28–31,34,37]. Essentially, most of them are concerned with the scheme devised in [18] or the one proposed in [35], both for unconstrained optimization. In this paper, we gradually increase ρ_k in order to satisfy the non-monotone sufficient decrease condition of [35] and, because the iterate x_{k+1} is computed by solving problem (2), the proposed algorithm can be regarded as a variation of that described in [12] where the authors used a non-monotone scheme of Grippo et al. [18] to obtain global convergence. Furthermore, when $A_k = I$, subproblem (2) is equivalent to

$$\begin{aligned} \text{minimize} \quad & \nabla f(x_k)^T(x-x_k) + \tfrac{1}{2}(x-x_k)^T B_k(x-x_k) \\ \text{s.t.} \quad & x \in \Gamma \text{ and } \|x-x_k\| \le \Delta_k, \end{aligned} \tag{3}$$

for a proper Δ_k depending on ρ_k (in fact, $\Delta_k = \|x^*(\rho_k) - x_k\|$ wherein $x^*(\rho_k)$ is a solution of (2)). In addition, $\rho_k \to \infty$ in subproblem (2) is equivalent to $\Delta_k \to 0$ in (3). Therefore, our algorithm can be seen as a variation of Algorithm 2.1 devised in [22] for solving (1).

It is worth remarking that when ρ_k is sufficiently large so that $(B_k + \rho_k A_k)$ is symmetric positive definite, problem (2) can be rewritten as a constrained least squares problem:

$$\min \|G_k^T x - (G_k^T x_k - G_k^{-1}\nabla f(x_k))\|_2^2 \quad \text{subject to } x \in \Gamma, \tag{4}$$

where $G_k G_k^T$ is the Cholesky factorization of $(B_k + \rho_k A_k)$. Therefore, our method becomes more attractive whenever (4) is easy to solve. As examples of such problems we cite the linearly constrained minimization problem, minimization over the symmetric matrices set [20] as well as minimization under orthogonality constraints [7,8,23,25,32,33,36].

To illustrate how this new algorithm can be employed, we propose a particular version of it that is well suited for solving large-scale orthogonal Procrustes problems (OPP). Given $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times p}$, the OPP consists of finding a matrix $X \in \mathbb{R}^{m\times q}$ with orthogonal columns such that the residual $\|AX - B\|_F^2$ is minimum. When $m = p$ this problem has a closed-form solution obtained from the singular value decomposition (SVD) of the matrix $A^T B$ [14]. The case is different when $p < m$ because we do not know the solution in closed form or because the objective function may have several local minima. Existing algorithms for these problems are iterative and require a sequence of SVD computations of $m \times q$ matrices, which can be prohibitive when m is large [1,12,36]. To circumvent possible difficulties with SVD computations in large-scale problems, we follow well known strategies for solving linear systems with multiple right-hand sides [21] which exploit the approximation properties of the block Lanczos bidiagonalization process. The basic idea of this approach is to project the OPP onto a Krylov subspace of small (but increasing) dimension, generated by the block Lanczos bidiagonalization process [14,21]. Proceeding this way the large-scale problem is transformed into a computationally much more tractable problem involving matrices of order $kp \times p$, $k \in \{1, 2, \ldots\}$. Numerical examples will illustrate that good approximate solutions to the OPP are obtained with fairly small k.

Our paper is organized as follows: In Section 2 we describe our algorithm. The theoretical analysis of the algorithm is addressed in Section 3. More precisely, we give a global convergence proof, that is, we prove that sequence $\{x_k\}$ converges to stationary points irrespective of the initial guess taken. Numerical results on nonlinear optimization problems involving linear constraints are presented in Section 4. Further, a numerical scheme to solve large-scale Orthogonal Procrustes Problems based on our algorithm is also presented. The article ends with some conclusions in Section 5.

We finish this section with the notation used throughout the paper. The gradient ∇f will be denoted by g. $C^1[a,b]$ is the set of continuously differentiable functions in $[a,b]$. $\mathbb{N} = \{0, 1, 2, \ldots\}$. Let $K \subseteq \mathbb{N}$ be an infinite subset of $\mathbb{N}$. $K_1 \overset{\infty}{\subseteq} K$ means that K_1 is an infinite set as well. Further, if $\{x_k\}_{k\in K_1}$ is an infinite sequence, $\lim_{k\in K_1} x_k$ denotes $\lim_{k\to\infty} x_k$ restricted to $k \in K_1$. $\|\cdot\|$ denotes the euclidean norm in $\mathbb{R}^n$, $\langle\cdot,\cdot\rangle$ the euclidean inner product and $\|\cdot\|_F$ the Frobenius norm in $\mathbb{R}^{m\times n}$. Given $A \in \mathbb{R}^{n\times n}$, $\mathrm{Tr}(A)$ means the trace of matrix A and $\mathrm{diag}(A) \in \mathbb{R}^n$ means its diagonal. Let us define $\mathbb{S}_n = \{A \in \mathbb{R}^{n\times n} \mid A^T = A\}$ and $\mathbb{S}_n^+ = \{A \in \mathbb{S}_n \mid x^T A x > 0 \text{ for all } x \in \mathbb{R}^n\}$, that is, the subset of symmetric and symmetric positive definite matrices, respectively.

2. Model algorithm

In this section, the main steps of our model algorithm are presented. As previously mentioned in the introduction, we incorporate a non-monotone scheme due to Zhang and Hager [35]. To summarize, by starting with an initial guess x_0, $C_0 = f(x_0)$, $q_0 = 1$, $\eta_k \in [\eta_{min}, \eta_{max}] \subseteq [0, 1]$ and $\delta \in (0, 1)$, our algorithm generates a sequence $\{x_k\}$ such that

$$f(x_{k+1}) \leq C_k + \delta \Phi_k(x_{k+1}) \tag{5}$$

where

$$\Phi_k(x) = g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T B(x - x_k), \tag{6}$$

for a proper matrix $B \in \mathbb{S}_n$, and C_k being updated as

$$C_{k+1} = \frac{\eta_k q_k C_k + f(x_{k+1})}{q_{k+1}}, \tag{7}$$

with

$$q_{k+1} = \eta_k q_k + 1. \tag{8}$$

Thus the algorithm proposed in this work corresponds to a variation of an algorithm by Francisco and Bazán [12], who use the non-monotone scheme of Grippo et al. [18]. In that case, a starting parameter $\mathcal{M} \in \mathbb{N}$ is chosen and a sequence $m(k)$ is updated such that $0 \leq m(k) \leq \min\{m(k-1)+1, \mathcal{M}\}$, for every $k \geq 1$ (with $m(0) = 0$). Finally, instead of (7), $C_k = \max\{f(x_{k-j}) \mid j \in \{0, 1, \ldots, m(k)\}\}$ in equation (5). We will show in Section 4 a brief comparison of both strategies through a performance profile.

Taking the aforementioned facts into account, the algorithm proposed in this work can be established as follows:

Algorithm 1. Initialization: Choose $x_0 \in \Gamma$, $\mu \in (0, 1]$, $\delta \in (0, 1)$, $0 < \rho_a \leq \rho_b < +\infty$, $1 < \zeta_1 \leq \zeta_2 < +\infty$, $0 \leq \eta_{min} \leq \eta_{max} \leq 1$ and $\eta_0 \in [\eta_{min}, \eta_{max}]$. Set $q_0 \leftarrow 1$, $C_0 \leftarrow f(x_0)$ and $k \leftarrow 0$.

Step 1. Compute $g(x_k)$, C_k according to (7) and set $\rho \in [\rho_a, +\infty)$.
Step 2. Pick $A \in \mathbb{S}_n^+$. If $\rho < \rho_b$, pick $B \in \mathbb{S}_n$, otherwise, pick $B = \rho I$.
Step 3. Define

$$Q_k(x) = g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T(B + \rho A)(x - x_k) \tag{9}$$

and let $\bar{x}_k$ be the global solution of

$$\begin{array}{rl} \text{minimize} & Q_k(x) \\ \text{s.t.} & x \in \Gamma. \end{array} \tag{10}$$

Step 4. Compute $x_k^+ \in \Gamma$ such that

$$Q_k(x_k^+) \leq \mu Q_k(\bar{x}_k). \tag{11}$$

If $Q_k(x_k^+) = 0$, terminate the execution declaring x_k as a stationary point of (1).
Step 5. Define $\Phi_k(x)$ as (6).
If

$$f(x_k^+) \leq C_k + \delta \Phi_k(x_k^+), \tag{12}$$

define $\rho_k = \rho$, $x_{k+1} = x_k^+$, $A_k = A$, $B_k = B$, choose $\eta_k \in [\eta_{min}, \eta_{max}]$, set $k \leftarrow k+1$ and go back to Step 1.
Else, choose $\rho_{new} \in [\zeta_1 \rho, \zeta_2 \rho]$, set $\rho = \rho_{new}$ and go back to Step 2.

Notice that $\mu = 1$ means that $x_k^+ \in \Gamma$ computed at Step 4 is the exact solution of subproblem (10). Notice also that such x_k^+ becomes an inexact solution whenever $\mu \in (0, 1)$: the closer μ is to 0 the more inexact x_k^+ is. The possibility $\mu \in (0, 1)$ is suitable when subproblem (10) is hard or computationally expensive. With respect to Step 4, if $Q_k(x_k^+) = 0$ it follows that $Q_k(\bar{x}_k) = 0 = Q_k(x_k)$ and so x_k is a solution of (10) as well. Therefore, since $\nabla Q_k(x_k) = \nabla f(x_k)$, x_k is a stationary point in the sense of Bouligand cones (or even as mentioned in [12, Definition 1]); in other words, we have that $\langle \nabla f(x_k), d \rangle \geq 0$ for all $d \in \mathcal{B}_\Gamma(x_k)$, wherein $\mathcal{B}_\Gamma(x_k)$ denotes the Bouligand cone of Γ in x_k (see [27] for further details).

3. Convergence analysis

Let the subsets of matrices chosen at Step 2 of Algorithm 1 be denoted by $\mathbb{B} = \{B\} \subseteq \mathbb{S}_n$ and $\mathbb{A}^+ = \{A\} \subseteq \mathbb{S}_n^+$. Our convergence analysis relies on the following assumptions:

A1. $\Gamma^0 = \{x \in \Gamma \mid f(x) \le f(x_0)\}$ is a bounded subset.
A2. The subsets of matrices $\mathbb{B}$ and $\mathbb{A}^+$ are uniformly bounded, that is, there exists $\widetilde{M} \ge 0$ such that $\|B\|_F \le \widetilde{M}$ and $\|A\|_F \le \widetilde{M}$ for all matrices B and A chosen at Step 2.
A3. There exists $\gamma > 0$ such that $x^T A x \ge \gamma \|x\|^2$ for all $A \in \mathbb{A}^+$ and $x \in \mathbb{R}^n$.
A4. We assume that the gradient of f is Lipschitz continuous in $\widehat{\Gamma}$. In other words, there exists $L_f \in \mathbb{R}$ such that

$$\|g(y) - g(x)\| \le L_f \|y - x\|, \tag{13}$$

for all $x, y \in \widehat{\Gamma} \supseteq \Gamma$.

Inequality (13) is met in a wide class of applications, e.g. when f has Hessian bounded in some norm or it is a twice continuously differentiable function in some compact set containing $\widehat{\Gamma}$. For later use, we notice that

$$|f(y) - f(x) - \langle g(x), y - x\rangle| = \left| \int_0^1 \langle g(x + t(y-x)) - g(x), y - x\rangle dt \right| \le \frac{L_f}{2}\|y - x\|^2,$$

for all $x, y \in \widehat{\Gamma}$, that is,

$$f(y) \le f(x) + \langle g(x), y - x\rangle + \frac{L_f}{2}\|y - x\|^2, \tag{14}$$

for all $x, y \in \widehat{\Gamma}$. Assumption A1 is often assumed in minimization algorithms; it ensures that $\{x_k\}$ has at least one accumulation point. Assumption A2 is essential to maintain the stability of the iterates. Assumption A3 ensures that $(\rho/2)(x - x_k)^T A (x - x_k)$ in (9) is a regularization term whose role is similar to the trust region in the trust-region like algorithms. Note that both A2 and A3 are user-controlled assumptions.

The next Lemma shows that Algorithm 1 is in fact non-monotone and $\{C_k\}_{k\in\mathbb{N}}$ is a decreasing sequence. These properties will be essential to prove the main results of this section.

Lemma 1. *For $k \ge 0$ let $x_{k+1} \in \Gamma$ be generated by Algorithm 1 and suppose that C_{k+1} is updated according to (7) and (8). Then, $f(x_{k+1}) \le C_{k+1} \le C_k$.*

Proof. Notice that

$$Q_k(x_{k+1}) = \Phi_k(x_{k+1}) + (\rho_k/2)(x_{k+1} - x_k)^T A_k (x_{k+1} - x_k) < 0.$$

This and Assumption A3 imply $\Phi_k(x_{k+1}) \le -(\rho_k\gamma)/2\|x_{k+1} - x_k\|^2$. By using this inequality in (5) we have that $f(x_{k+1}) \le C_k - (\delta\rho_k\gamma)/2\|x_{k+1} - x_k\|^2$. Thus, since $q_k \ge 1$ for all k, from (7),

$$C_{k+1} \le C_k - \frac{\delta\rho_k\gamma}{2q_{k+1}}\|x_{k+1} - x_k\|^2 < C_k. \tag{15}$$

Now (7) and (8) imply $f(x_{k+1}) - C_{k+1} = \eta_k q_k (C_{k+1} - C_k) \le 0$. Therefore $f(x_{k+1}) \le C_{k+1} \le C_k$. □

Our following result shows that the loop defined from Step 2 to Step 5 in Algorithm 1 finishes after a finite number of cycles.

Lemma 2. *Algorithm 1 is well defined.*

Proof. Suppose $x_k \in \Gamma$ is not a stationary point. We will prove that condition (12) is fulfilled for all $\rho \ge \max\{\rho_b, L_f\}$. To this end note that $\Phi_k(x) = \langle g(x_k), x - x_k\rangle + \rho/2\|x - x_k\|^2$ for all $\rho \ge \rho_b$. Then, for $\rho \ge \max\{\rho_b, L_f\}$ we have from (14) that

$$f(y) \le f(x_k) + \langle g(x_k), y - x_k\rangle + \frac{\rho}{2}\|y - x_k\|^2 = f(x_k) + \Phi_k(y)$$

for all $y \in \Gamma$. Then, since inequality $Q_k(x_k^+) \le \mu Q_k(\bar{x}_k) < 0$ always holds true, it follows that $\Phi_k(x_k^+) < 0$. Therefore, since $f(x_k) \le C_k$ (by Lemma 1 and the fact that $C_0 = f(x_0)$), we have that $x_k^+ \in \Gamma$ computed at Step 4 fulfills

$$f(x_k^+) \le C_k + \delta\Phi_k(x_k^+),$$

for all $\rho \ge \max\{\rho_b, L_f\}$. □

The next technical results are essential to prove global convergence to stationary points. First, it is worth noting that sequence $\{x_k\}$ can be assumed to be infinite, otherwise, we would have that $\Phi(x_k^+) = 0$ for some k and, since $\nabla\Phi(x_k) = \nabla f(x_k)$, x_k would be a stationary point.

Lemma 3. *Sequence $\{\rho_k\}_k$ is bounded.*

Proof. By contradiction, let us suppose that there exists $N_1 \overset{\infty}{\subseteq} \mathbb{N}$ such that $\lim_{k\in N_1} \rho_k = \infty$. Then, for every $k \in N_1$, there exists $\bar\rho_k \in [\rho_k/\zeta_2, \rho_k/\zeta_1]$ and $x_k^+ \in \Gamma$ such that $f(x_k^+) > C_k + \delta\Phi_k(x_k^+)$. Since $\lim_{k\in N_1} \bar\rho_k = \infty$ and $\Phi_k(x_k^+) < 0$, there exists $k_0 \in N_1$ such that $\bar\rho_k \geq \max\{\rho_b, L_f\}$, $B = \bar\rho_k I$ at Step 2 and consequently, from Lemma 1, $f(x_k^+) > f(x_k) + g(x_k)^T(x_k^+ - x_k) + (\bar\rho_k/2)\|x_k^+ - x_k\|^2$ for all $k \geq k_0$, which is in contradiction with (14). $\square$

Lemma 4. *Sequence $\{x_k\}_{k\in\mathbb{N}} \subset \Gamma^0$.*

Proof. Since $C_0 = f(x_0)$, the proof follows straightforwardly from Lemma 1. $\square$

Lemma 5. *$\{C_k\}_{k\in\mathbb{N}}$ is convergent.*

Proof. From Lemma 4 and from the continuity of f, we have that $\{f(x_k)\}$ is bounded from below. From Lemma 1 we have that $\{C_k\}_{k\in\mathbb{N}}$ is non-increasing and bounded from below and therefore convergent. $\square$

Lemma 6. *Let $\{x_k\}_{k\in\mathbb{N}}$ be a sequence generated by Algorithm 1. Then*

(i) $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$ *if* $\eta_{max} < 1$;
(ii) $\lim_{k\to\infty} f(x_k) = \lim_{k\to\infty} C_k$ *if* $\eta_{max} < 1$;
(iii) $\limsup_{k\to\infty}(f(x_{k+1}) - C_{k+1}) = \limsup_{k\to\infty} \Phi_k(x_{k+1}) = 0$ *if* $\eta_{max} = 1$.

Proof. We note from (8) that

$$q_k = 1 + \sum_{i=1}^{k}\prod_{j=1}^{i}\eta_{k-j}$$

for all $k \geq 1$. Hence we have that

$$\frac{1}{1-\eta_{min}} \leq q_k \leq \min\left\{1+k, \frac{1}{1-\eta_{max}}\right\} \tag{16}$$

for all $k \in \mathbb{N}$ (here $1/(1-\eta_{max}) = \infty$ if $\eta_{max} = 1$). Now, since by assumption $\eta_{max} < 1$, from (15) and Lemma 5 we have that $\lim_{k\to\infty}\|x_{k+1} - x_k\| = 0$ and thus (i) is proved.

On the other hand, from (5), (7) and (8), it follows that

$$\eta_{max} q_k (C_{k+1} - C_k) \leq f(x_{k+1}) - C_{k+1} \leq \eta_{min} q_k (C_{k+1} - C_k). \tag{17}$$

Then, by supposing that $\eta_{max} < 1$ and by (16) and (17), we obtain that

$$\frac{\eta_{max}}{1-\eta_{max}}(C_{k+1} - C_k) \leq f(x_{k+1}) - C_{k+1} \leq \frac{\eta_{min}}{1-\eta_{min}}(C_{k+1} - C_k),$$

and (ii) follows from Lemma 5. Also, since $f(x_{k+1}) \leq C_{k+1}$ for all $k \in \mathbb{N}$ (see Lemma 1), by Lemma 5 and (17) it turns out that

$$-\infty < \sum_{k=0}^{\infty} \frac{f(x_{k+1}) - C_{k+1}}{q_k} \leq 0.$$

Analogously, we prove from (5), (7) and (8) that

$$-\infty < \delta \sum_{k=0}^{\infty} \frac{\Phi_k(x_{k+1})}{q_{k+1}} \leq 0.$$

Thus, since $q_k \leq k+1$ for all k (see (16)), it follows that

$$\limsup_{k\to\infty}(f(x_{k+1}) - C_{k+1}) = \limsup_{k\to\infty}\Phi_k(x_{k+1}) = 0,$$

which concludes the proof. $\square$

Remark 1. It is worth mentioning that Lemma 4 and Assumption A1 imply that $\{x_k\}_{k\in\mathbb{N}}$ has at least one limit point. In addition, if the number of limit points is finite, from Lemma 6-(i) and by a similar proof as given in [24, p. 476], we can show that $\{x_k\}_{k\in\mathbb{N}}$ is a convergent sequence.

The main theoretical result of the section assures that Algorithm 1 is globally convergent when $\eta_{max} < 1$.

Theorem 7. *Suppose that $\eta_{max} < 1$. Then all accumulation points of $\{x_k\}_{k\in\mathbb{N}}$ are stationary points of problem (1). Furthermore, if f has a finite number of stationary points in Γ^0, then $\{x_k\}$ is convergent.*

Proof. Let x^* be an accumulation point of $\{x_k\}$ and consider $\mathcal{N}_1 \overset{\infty}{\subseteq} \mathbb{N}$ such that $\lim_{k\in\mathcal{N}_1} x_k = x^*$. From Assumption A2 it follows that there exist $\mathcal{N}_2 \overset{\infty}{\subseteq} \mathcal{N}_1$ and $\bar{A}, \bar{B} \in \mathbb{S}_n$ such that $\lim_{k\in\mathcal{N}_2} B_k = \bar{B}$ and $\lim_{k\in\mathcal{N}_2} A_k = \bar{A}$. Also, from Lemma 3 there exists $\bar{\rho}$ such that $\rho_k \le \bar{\rho}$ for all $k \in \mathbb{N}$. Now, define $Q^\star(x) = \nabla f(x^*)^T(x - x^*) + (1/2)(x - x^*)^T(\bar{B} + \bar{\rho}\bar{A})(x - x^*)$ and let $\bar{x} \in \Gamma$ be a solution of

$$\begin{aligned} \text{minimize} \quad & Q^\star(x) \\ \text{s.t.} \quad & x \in \Gamma. \end{aligned} \tag{18}$$

Since $f(x_{k+1}) \le C_k + \delta\Phi_k(x_{k+1})$ and $\eta_{max} < 1$, Lemma 6-(ii) implies that $\lim_{k\in\mathcal{N}_2} \Phi_k(x_{k+1}) = 0$. Thus, since $\Phi_k(x_{k+1}) \le Q_k(x_{k+1}) \le 0$, it follows that

$$\lim_{k\in\mathcal{N}_2} Q_k(x_{k+1}) = 0.$$

Now, from the continuity of ∇f, note that

$$\begin{aligned} 0 \ge \mu Q^\star(\bar{x}) &\ge \lim_{k\in\mathcal{N}_2} \mu \left[\nabla f(x_k)^T(\bar{x} - x_k) + (1/2)(\bar{x} - x_k)^T(B_k + \rho_k A_k)(\bar{x} - x_k)\right] \\ &\ge \lim_{k\in\mathcal{N}_2} Q_k(x_{k+1}) = 0, \end{aligned}$$

that is, x^* is also a solution of (18) and so it is a stationary point of (1) (in the sense of Bouligand cones). The second part of the theorem follows straightforwardly from Remark 1. □

Remark 2. Some comments concerning the case when $\eta_{max} = 1$ are in order. By Lemma 6-(iii), we have that there exists $\bar{N} \overset{\infty}{\subseteq} \mathbb{N}$ such that $\lim_{k\in\bar{N}} \Phi_k(x_{k+1}) = 0$. Therefore, by a similar proof given in Theorem 7 we can prove that all accumulation points of $\{x_k\}_{k\in\bar{N}}$ are stationary for (1). In other words, sequence $\{x_k\}_{k\in\bar{N}}$ has at least one limit point that is a stationary point.

Remark 3. If x_k^+ (computed at Step 4 of Algorithm 1) is not in Γ^0 defined in Assumption A1, from Lemma 1 we have that $C_k \le C_0 = f(x_0)$ and thus certainly this point will not satisfy the non-monotone sufficient decrease condition (12). Hence, instead of set $x_k^+ \in \Gamma$ at Step 4, we can set $x_k^+ \in \Gamma^0$ and, consequently, without affecting the theoretical results of this section, Assumption A4 can be replaced by ∇f being Lipschitz continuous in some open set containing Γ^0; this can be of interest in certain problems where Assumption A4 holds true only in an open subset containing Γ^0.

4. Numerical experiments

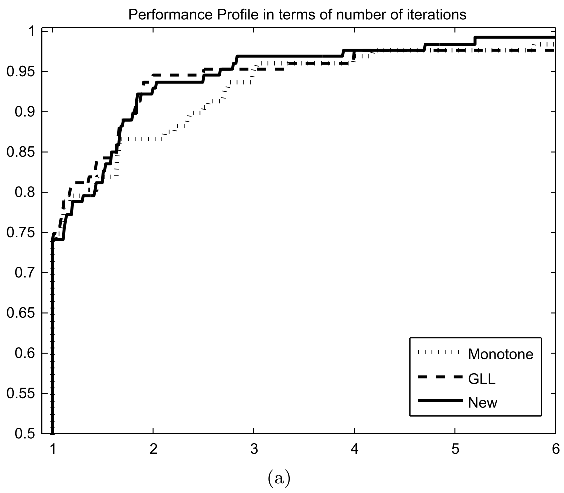
This section is divided into two parts. In the first part we consider a specialized version of Algorithm 1 that is well suited for solving minimization problems with linear constraints and apply it to a number of test problems taken from the CUTEst test collection [16]. The second part concerns a numerical method for large scale orthogonal Procrustes problem based on a combination of the above mentioned specialized version of Algorithm 1 and the block Lanczos bidiagonalization scheme [14,21].

Regarding the algorithm in the first part, we adopt a special choice for A and B as multiples of the identity matrix. This choice is based on experiments highlighting the behavior of the Barzilai–Borwein scheme [4] when combined with non-monotone line search techniques. As a result we obtain a non-monotone spectral projected gradient (NSPG) version to minimize continuous differentiable functions on arbitrary closed sets. For description purposes, let us define

$$\sigma_k = \begin{cases} \dfrac{(g(x_k) - g(x_{k-1}))^T(x_k - x_{k-1})}{\|x_k - x_{k-1}\|^2}, & \text{if } k \ne 0, \\ 1, \ \text{if } k = 0. \end{cases} \tag{19}$$

So by setting $B = (\sigma_k/2)I$ and $A = I$ at Step 2 of Algorithm 1, the following scheme is established.

Algorithm 2 *(Non-monotone Spectral Projected Gradient version).* Choose $x_0 \in \Gamma$, $\delta \in (0, 1)$, $0 < \rho_a \le \rho_b < +\infty$ and $1 < \zeta_1 \le \zeta_2 < +\infty$. Set $q_0 \leftarrow 1$, $C_0 \leftarrow f(x_0)$ and $k \leftarrow 0$.



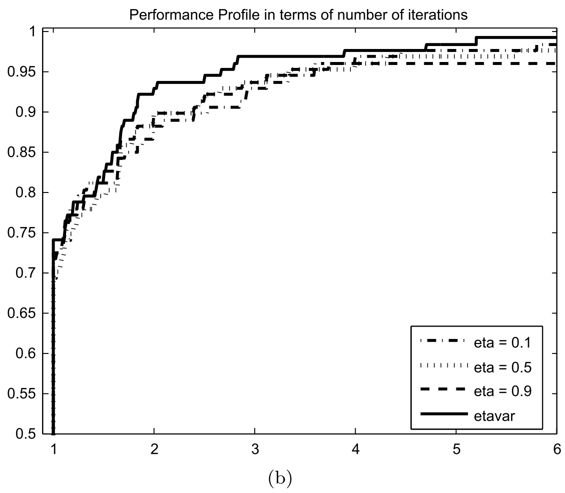


Fig. 1. Performance profiles in terms of number of iterations for the methods implemented, including the monotone, non-monotone following Grippo et al., and our strategy for varying values of η_k.

Step 1. Compute $g(x_k)$, C_k as (7), σ_k as (19) and set $\rho = \max\{\min\{\sigma_k/2, \rho_b\}, \rho_a\}$.

Step 2. Define $w_k = x_k - \dfrac{2}{\sigma_k + 2\rho} g(x_k)$ and let x_k^+ be the global solution of

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|x - w_k\|^2 \\ \text{s.t.} & x \in \Gamma. \end{array} \tag{20}$$

If $\|x_k^+ - w_k\|^2 = 2\|g(x_k)\|^2/(\sigma_k + 2\rho)^2$, terminate the execution declaring x_k as a stationary point of (1).

Step 3. If

$$f(x_k^+) \le C_k + \delta \left(g(x_k)^T (x_k^+ - x_k) + (\sigma_k/4)\|x_k^+ - x_k\|^2 \right), \tag{21}$$

define $\rho_k = \rho$, $x_{k+1} = x_k^+$, set $k \leftarrow k+1$ and go back to Step 1. Else, choose $\rho_{new} \in [\zeta_1\rho, \zeta_2\rho]$, set $\rho = \rho_{new}$ and go back to Step 2.

In our numerical experiments with Algorithm 2 we consider a subset of the CUTEst test collection containing 127 problems with linear constraints. Most of the problems have quadratic objective functions, except for 28, which have general objective functions. Since our algorithm requires an initial feasible point, we chose not to include in our results problems which did not converge due to infeasible initial points; in these cases, all variants tested presented the same behavior. We also excluded from our experiments problems where the number of constraints and the number of variables was too large ($n + m > 10000$). For the quadratic subproblem, the solver can be chosen by the user, as our convergence theory accepts approximate solutions for the subproblem and this should not change the behavior of the method.

Our implementation was done in Fortran 2008, and our tests were run in a PC with a quad-core 3.30 GHz processor and 4 GB of RAM. The subproblems were solved using the QPC solver which is part of the GALAHAD package [15], with default parameters. We set a tolerance of $\epsilon = 10^{-5}$ for our criticality measure, and we chose $\delta = 0.1$, $\rho_a = 0.5$, $\rho_b = 10^5$, $\zeta_1 = \zeta_2 = 5$, $\eta_{min} = 0$ and $\eta_{max} = 0.9$.

The problems, as well as their corresponding dimensions, are listed in Table 1.

The complete results for various η_k values are displayed in Table 2. For comparison, we also list results obtained with $\eta_k = 0$ for all k (which corresponds to a monotone method) as well as the number of iterations and the computational time in seconds (in parentheses) required to achieve convergence (denoted by *CPU Time*).

In our implementation, we have taken advantage of the flexibility allowed in the convergence theory and tested both constant and varying values of η_k. In the varying case, we have chosen to start with a value closer to one and gradually reduce the value of η_k as we approached the solution; this strategy has been described in [35], and appears to perform reasonably well in practice. Results obtained with this strategy are labeled as *etavar*. In our case we choose $\eta_0 = 0.9$ and take $\eta_{k+1} = 0.9\eta_k$ at each successful iteration.

Figs. 1(a) and 1(b) show the performance profiles of two sets of tests; for details concerning performance profiles the reader is referred to [11]. Fig. 1(a) shows a comparison of the performance of the monotone method and the non-monotone one (with *etavar* parameter choice), denoted by *new*. Further, for the sake of comparison, we replace the non-monotone strategy of [35] with that described in [18] and we denoted its performance with *GLL*. Fig. 1(b) displays results obtained with the new strategy for different values of η_k.

Table 1
Problems where Algorithm 2 was tested.

Problem name	Number of variables	Number of constraints
A0NSDSDS	6012	2004
A5NSDSDM	6012	2004
A5NSSNSM	6012	2004
ALLINQP	100	50
AUG2DQP	24	9
AUG2D	24	9
AUG3DCQP	7463	2000
AUG3DC	3873	1000
AUG3DQP	156	27
AUG3D	904	180
AVGASA	8	10
AVGASB	8	10
BLOCKQP1	25	11
BLOCKQP2	25	11
BLOCKQP3	2005	1001
BLOCKQP4	2005	1001
BLOCKQP5	2005	1001
BT3	5	3
CB	11163	244
CVXQP1	1000	500
CVXQP2	1000	250
CVXQP3	10	6
DALE	16514	405
DEGENQPC	50	125025
DEGENQP	50	19625
DEGTRIDL	1001	1
DTOC1L	5998	3996
DUAL1	85	1
DUAL2	96	1
DUAL3	111	1
DUAL4	75	1
DUALC1	9	215
DUALC2	7	229
DUALC5	8	278
DUALC8	8	503
EXPFITA	5	22
EXPFITB	5	102
EXPFITC	5	502
FCCU	19	8
GENHS28	10	8
GMNCASE2	175	1050
GOULDQP1	32	17
GOULDQP3	699	349
HATFLDH	4	7
HIER13	2020	3313
HONG	4	1
HS105	8	1
HS112	10	3
HS118	15	17
HS21	2	1
HS24	2	3
HS28	3	1
HS35	3	1
HS35MOD	3	1
HS36	3	1
HS37	3	2
HS41	4	1
HS44NEW	4	6
HS44	4	6
HS48	5	2
HS49	5	2
HS51	5	3
HS52	5	3
HS53	5	3
HS55	6	6
HS62	3	1
HS76	4	3
HS86	5	10
HS9	2	1
HUBFIT	2	1
KSIP	20	1001
LEUVEN7	300	946
LISWET10	403	400
LISWET11	2002	2000
LISWET12	103	100
LISWET1	103	100
LISWET2	2002	2000
LISWET3	2002	2000
LISWET4	2002	2000
LISWET5	2002	2000
LISWET6	2002	2000
LISWET7	2002	2000
LISWET8	2002	2000
LISWET9	103	100
LOTSCHD	12	7
LSQFIT	2	1
MOSARQP1	2500	700
MOSARQP2	900	600
NCVXQP1	50	25
NCVXQP2	50	25
NCVXQP3	100	50
NCVXQP4	100	25
NCVXQP5	50	12
NCVXQP6	100	25
NCVXQP7	50	36
NCVXQP8	100	75
NCVXQP9	50	36
OSORIO	10201	202
PORTFL1	12	1
PORTFL2	12	1
PORTFL3	12	1
PORTFL4	12	1
PORTFL6	12	1
PORTSNQP	1000	2
PORTSQP	1000	1
POWELL20	10	10
PRIMAL1	325	85
PRIMAL2	649	96
PRIMAL3	745	111
PRIMAL4	1489	75
PRIMALC5	287	8
QPCBOEI1	384	351
QPNBAND	10000	5000
RDW2D52U	18	1
SOSQP2	20	11
STCQP1	4097	2052
STCQP2	4097	2052
STNQP1	4097	2052
STNQP2	4097	2052
TABLE7	624	17
TABLE8	1271	72
TARGU	162	63
TFI3	3	101

Table 2
Results comparing the monotone and non-monotone algorithms.

Problem	n. iterations (CPU time)				
	$\eta = 0$	$\eta = 0.1$	$\eta = 0.5$	$\eta = 0.9$	*etavar*
A0NSDSDS	23(16.32)	23(16.34)	23(16.34)	23(16.36)	14(16.38)
A5NSDSDM	23(16.38)	23(16.39)	23(16.50)	23(16.52)	14(16.54)
A5NSSNSM	23(16.46)	23(16.48)	23(16.46)	23(16.44)	14(16.56)
ALLINQP	267(0.34)	266(0.33)	187(0.34)	105(0.18)	160(0.45)
AUG2DQP	9(0.01)	9(0.01)	9(0.01)	9(0.01)	9(0.01)
AUG2D	12(0.01)	12(0.01)	12(0.01)	12(0.01)	12(0.01)
AUG3DCQP	1(0.27)	1(0.27)	1(0.27)	1(0.27)	1(0.27)
AUG3DC	1(0.02)	1(0.02)	1(0.02)	1(0.02)	1(0.02)
AUG3DQP	7(0.02)	7(0.02)	7(0.02)	7(0.02)	7(0.02)
AUG3D	11(0.02)	11(0.02)	11(0.02)	11(0.02)	11(0.02)
AVGASA	4(0.00)	8(0.01)	8(0.01)	8(0.01)	6(0.01)
AVGASB	4(0.00)	8(0.01)	8(0.01)	8(0.01)	6(0.01)
BLOCKQP1	4(0.00)	4(0.00)	4(0.00)	11(0.01)	1(0.01)
BLOCKQP2	3(0.00)	3(0.00)	3(0.00)	3(0.00)	3(0.01)
BLOCKQP3	8(0.32)	8(0.32)	8(0.31)	8(0.32)	8(0.31)
BLOCKQP4	10(0.40)	10(0.41)	10(0.40)	10(0.40)	10(0.42)
BLOCKQP5	8(0.26)	8(0.26)	8(0.26)	8(0.27)	8(0.26)
BT3	7(0.01)	7(0.01)	7(0.00)	7(0.00)	7(0.00)
CB	36(18.12)	17(16.17)	17(16.13)	17(16.11)	27(16.17)
CVXQP1	41(2.65)	41(2.61)	72(9.57)	72(9.58)	42(9.52)
CVXQP2	115(3.48)	111(3.36)	54(0.71)	54(0.70)	38(0.71)
CVXQP3	3(0.00)	10(0.01)	10(0.01)	10(0.01)	8(0.01)
DALE	27(14.20)	13(10.77)	13(10.78)	13(10.79)	10(10.83)
DEGENQPC	2(1.74)	2(1.76)	2(1.74)	2(1.73)	2(1.73)
DEGENQP	2(10.13)	2(10.16)	2(9.93)	2(9.95)	2(10.15)
DEGTRIDL	21(0.08)	21(0.07)	21(0.08)	21(0.07)	21(0.08)
DTOC1L	19(2.17)	19(2.13)	19(2.13)	19(2.16)	19(2.15)
DUAL1	70(0.08)	67(0.09)	10(0.01)	10(0.01)	64(0.01)
DUAL2	52(0.07)	40(0.07)	9(0.01)	9(0.01)	35(0.01)
DUAL3	65(0.10)	58(0.09)	10(0.02)	10(0.02)	47(0.02)
DUAL4	33(0.03)	32(0.03)	11(0.02)	11(0.02)	20(0.02)
DUALC1	84(0.67)	36(0.30)	56(0.24)	56(0.24)	40(0.23)
DUALC2	5(0.03)	18(0.07)	28(0.10)	55(0.18)	14(0.14)
DUALC5	20(0.07)	20(0.07)	25(0.10)	42(0.16)	18(0.16)
DUALC8	23(0.24)	26(0.28)	56(0.40)	56(0.41)	39(0.41)
EXPFITA	497(0.42)	497(0.44)	497(0.42)	497(0.43)	497(0.42)
EXPFITB	268(1.56)	268(1.57)	268(1.57)	268(1.57)	268(1.53)
EXPFITC	106(1.04)	106(1.04)	106(1.04)	106(1.05)	106(1.04)
FCCU	11(0.01)	11(0.01)	11(0.01)	11(0.01)	11(0.01)
GENHS28	12(0.01)	12(0.01)	12(0.01)	12(0.01)	11(0.01)
GMNCASE2	26(2.79)	30(4.07)	24(3.30)	26(2.80)	34(4.44)
GOULDQP1	30(0.03)	30(0.03)	30(0.03)	30(0.03)	30(0.03)
GOULDQP3	5(0.12)	5(0.12)	5(0.12)	5(0.12)	5(0.12)
HATFLDH	1(0.00)	1(0.01)	1(0.00)	1(0.00)	1(0.00)
HIER13	10(7.12)	36(20.21)	95(49.52)	405(198.68)	52(63.06)
HONG	8(0.01)	8(0.01)	8(0.01)	8(0.01)	8(0.01)
HS105	113(0.08)	38(0.05)	58(0.07)	57(0.05)	62(0.08)
HS112	171(0.11)	41(0.04)	109(0.09)	153(0.10)	83(0.09)
HS118	433(0.31)	433(0.30)	433(0.31)	433(0.30)	433(0.31)
HS21	35(0.02)	34(0.02)	35(0.02)	35(0.02)	34(0.02)
HS24	6(0.01)	6(0.01)	6(0.01)	6(0.00)	6(0.00)
HS28	14(0.01)	14(0.01)	14(0.01)	14(0.01)	13(0.01)
HS35	10(0.01)	9(0.01)	10(0.01)	10(0.01)	10(0.01)
HS35MOD	9(0.01)	10(0.01)	9(0.01)	9(0.01)	9(0.01)
HS36	1(0.00)	1(0.00)	1(0.00)	1(0.00)	1(0.00)
HS37	6(0.01)	6(0.01)	6(0.01)	6(0.01)	6(0.01)
HS41	1(0.00)	1(0.00)	1(0.00)	1(0.00)	1(0.00)
HS44NEW	3(0.00)	3(0.00)	3(0.00)	3(0.01)	3(0.00)
HS44	4(0.00)	4(0.01)	4(0.00)	4(0.01)	4(0.00)
HS48	7(0.01)	7(0.00)	7(0.01)	7(0.00)	7(0.01)
HS49	42(0.02)	72(0.04)	50(0.03)	42(0.02)	60(0.04)
HS50	13(0.01)	13(0.01)	13(0.01)	13(0.01)	13(0.01)
HS51	5(0.00)	5(0.00)	5(0.00)	5(0.00)	5(0.01)
HS52	4(0.01)	4(0.00)	4(0.00)	4(0.00)	4(0.00)
HS53	4(0.00)	4(0.00)	4(0.00)	4(0.00)	4(0.01)
HS55	3(0.00)	3(0.00)	3(0.00)	3(0.00)	3(0.00)
HS62	44(0.03)	83(0.07)	54(0.04)	62(0.04)	81(0.07)

(*continued on next page*)

Table 2 (*continued*)

Problem	n. iterations (CPU time)				
	$\eta = 0$	$\eta = 0.1$	$\eta = 0.5$	$\eta = 0.9$	*etavar*
HS76	6(0.01)	6(0.01)	6(0.01)	6(0.01)	6(0.01)
HS86	10(0.01)	10(0.01)	10(0.01)	10(0.01)	10(0.01)
HS9	51(0.03)	51(0.03)	51(0.03)	51(0.03)	51(0.03)
HUBFIT	9(0.01)	9(0.01)	9(0.01)	9(0.00)	9(0.01)
KSIP	1(0.54)	1(0.56)	1(0.53)	1(0.54)	1(0.53)
LEUVEN7	1(5.67)	594(408.90)	422(266.06)	1(5.63)	1(5.69)
LISWET10	1(0.06)	1(0.06)	1(0.06)	1(0.06)	1(0.06)
LISWET11	1(0.31)	1(0.31)	1(0.31)	1(0.31)	1(0.31)
LISWET12	16(2.39)	16(2.43)	16(2.40)	16(2.39)	16(2.41)
LISWET1	1(0.01)	1(0.01)	1(0.01)	1(0.01)	1(0.01)
LISWET2	1(0.38)	1(0.38)	1(0.38)	1(0.38)	1(0.38)
LISWET3	1(0.16)	1(0.16)	1(0.16)	1(0.16)	1(0.16)
LISWET4	1(0.16)	1(0.17)	1(0.17)	1(0.16)	1(0.17)
LISWET5	1(0.17)	1(0.17)	1(0.17)	1(0.17)	1(0.17)
LISWET6	1(0.18)	1(0.19)	1(0.18)	1(0.19)	1(0.19)
LISWET7	1(0.59)	1(0.59)	1(0.60)	1(0.59)	1(0.59)
LISWET8	1(0.73)	1(0.72)	1(0.73)	1(0.71)	1(0.72)
LISWET9	32(6.11)	32(6.07)	32(6.02)	32(6.03)	32(5.98)
LOTSCHD	5(0.00)	12(0.01)	12(0.01)	12(0.01)	9(0.01)
LSQFIT	7(0.01)	7(0.01)	7(0.01)	7(0.01)	7(0.01)
MOSARQP1	8(0.25)	8(0.25)	8(0.25)	8(0.26)	8(0.25)
MOSARQP2	20(0.50)	23(0.74)	20(0.51)	20(0.49)	24(0.68)
NCVXQP1	3(0.02)	3(0.02)	3(0.02)	3(0.02)	3(0.02)
NCVXQP2	4(0.02)	4(0.02)	4(0.02)	4(0.02)	4(0.02)
NCVXQP3	13(0.11)	9(0.13)	15(0.13)	13(0.11)	18(0.15)
NCVXQP4	4(0.06)	4(0.06)	4(0.06)	4(0.06)	4(0.06)
NCVXQP5	3(0.02)	3(0.02)	3(0.02)	3(0.02)	3(0.02)
NCVXQP6	15(0.05)	15(0.11)	16(0.11)	15(0.05)	17(0.11)
NCVXQP7	2(0.02)	2(0.02)	2(0.02)	2(0.01)	2(0.02)
NCVXQP8	2(0.04)	2(0.04)	2(0.04)	2(0.04)	2(0.04)
NCVXQP9	10(0.03)	4(0.04)	10(0.03)	10(0.03)	10(0.03)
OSORIO	14(4.50)	7(4.33)	7(4.35)	7(4.31)	6(4.33)
PORTFL1	36(0.03)	36(0.03)	36(0.03)	36(0.03)	36(0.03)
PORTFL2	24(0.02)	24(0.02)	24(0.02)	24(0.02)	24(0.02)
PORTFL3	37(0.03)	37(0.03)	37(0.03)	37(0.03)	37(0.03)
PORTFL4	33(0.03)	33(0.02)	33(0.03)	33(0.03)	33(0.03)
PORTFL6	31(0.03)	31(0.02)	31(0.02)	31(0.02)	31(0.03)
PORTSNQP	2(0.01)	2(0.02)	2(0.02)	2(0.02)	2(0.02)
PORTSQP	2(0.01)	2(0.01)	2(0.01)	2(0.01)	2(0.01)
POWELL20	1(0.00)	1(0.00)	1(0.00)	1(0.00)	1(0.00)
PRIMAL1	1(0.11)	1(0.11)	1(0.11)	1(0.11)	1(0.11)
PRIMAL2	1(0.16)	1(0.15)	1(0.15)	1(0.15)	1(0.15)
PRIMAL3	3(1.51)	3(1.52)	3(1.49)	3(1.48)	3(1.48)
PRIMAL4	4(0.89)	4(0.89)	4(0.89)	4(0.89)	4(0.90)
PRIMALC5	329(4.12)	329(4.04)	329(4.08)	329(4.09)	329(4.11)
QPCBOEI1	40(3.97)	53(3.86)	53(3.85)	53(3.86)	27(3.86)
QPNBAND	6(2.52)	11(3.95)	11(3.96)	11(3.96)	11(3.96)
RDW2D52U	280(0.11)	280(0.11)	280(0.12)	280(0.11)	280(0.12)
SOSQP2	10(0.02)	10(0.02)	10(0.02)	10(0.02)	10(0.03)
STCQP1	1(19.77)	1(19.04)	1(18.83)	1(18.76)	1(18.89)
STCQP2	129(7.84)	109(9.59)	153(12.47)	142(9.58)	123(10.42)
STNQP1	5(2.12)	3(3.13)	5(2.04)	5(2.03)	5(2.03)
STNQP2	5(0.77)	3(1.71)	5(1.72)	5(0.76)	5(0.75)
TABLE7	33(0.61)	48(1.20)	99(2.30)	320(6.97)	65(3.08)
TABLE8	22(0.37)	10(0.14)	10(0.14)	10(0.14)	8(0.14)
TARGU	31(0.11)	40(0.14)	91(0.31)	285(0.89)	14(0.41)
TFI3	3(0.04)	3(0.04)	3(0.04)	3(0.04)	3(0.04)
ZECEVIC2	3(0.00)	3(0.00)	3(0.00)	3(0.00)	3(0.00)

For this set of problems, we can see from the results that both non-monotone strategies are clearly superior to the monotone one. Among the non-monotone strategies, however, that of Grippo et al. [18] appears to perform slightly better than the strategy of [35]. It is worth mentioning, however, that both nonmonotone strategies are expected to have similar behavior for most problems, which justifies our formulation as an alternative for the nonmonotone strategy used in [12].

4.1. Application to large-scale orthogonal Procrustes problem

Let $A \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times p}$. In this section we apply Algorithm 2 to solve the Orthogonal Procrustes Problem (OPP):

$$\min \|AX - B\|_F^2, \text{ s.t. } X^TX = I \text{ and } X \in \mathbb{R}^{m\times p}. \tag{22}$$

Our decision to do so is motivated by numerical experiments reported in [12] where it is shown that this kind of problems can be solved successfully using non-monotone strategies. Some practical applications of this problem appear in factor analysis, structural identification, global positioning systems and others [5,17].

Since we are concerned with large-scale problems, our strategy here is use the block Lanczos bidiagonalization method (BLDM) for solving large-scale problems. The BLDM can be described as follows:

Algorithm 3 *(BLDM).* Starting from matrix B of (22):

```
Compute $U_1$, $V_1$ with orthonormal columns such that $U_1B_1 = B$ and $V_1A_1 = A^TU_1$ (reduced QR factorization of $B$ and $A^TU_1$).
For $i = 1, 2, \dots, k$
    $U_{i+1}B_{i+1} = AV_i - U_iA_i^T$ (reduced QR factorization)
    $V_{i+1}A_{i+1} = A^TU_{i+1} - V_i^TB_{i+1}^T$ (reduced QR factorization)
```

Notice that $U_i, V_i \in \mathbb{R}^{m\times p}$ are orthogonal matrices and $B_i, A_i \in \mathbb{R}^{p\times p}$ are upper triangular matrices. Further, by defining,

$$\bar{U}_k \equiv [U_1U_2\cdots U_k] \in \mathbb{R}^{m\times kp}, \quad \bar{V}_k \equiv [V_1V_2\cdots V_k] \in \mathbb{R}^{m\times kp} \text{ and}$$

$$T_k \equiv \begin{bmatrix} A_1^T & & & \\ B_2 & A_2^T & & \\ & \ddots & \ddots & \\ & & B_k & A_k^T \\ & & & B_{k+1} \end{bmatrix} \in \mathbb{R}^{(k+1)p\times kp},$$

after k BLDM steps the following recurrence relations hold true:

$$\bar{U}_{k+1}E_1B_1 = B; \tag{23}$$

$$A\bar{V}_k = \bar{U}_{k+1}T_k, \tag{24}$$

$$A^T\bar{U}_{k+1} = \bar{V}_kT_k^T + V_{k+1}A_{k+1}E_{k+1}^T, \tag{25}$$

where $E_i \in \mathbb{R}^{m\times p}$ is zero except for the lines from $(i-1)p+1$ to ip, which are the $p \times p$ identity matrix. In addition, $\bar{V}_k^T\bar{V}_k = I$ and $\bar{U}_{k+1}^T\bar{U}_{k+1} = I$.

Block Lanczos bidiagonalization has been applied to efficiently solve large-scale linear systems and eigenvalue problems. For some applications, mainly for ill-conditioned large-scale problems, re-orthogonalization strategies in the columns of U_k and V_k are required. For further details on Block Lanczos Bidiagonalization the reader is referred to [2,3,21].

Regarding the construction of approximate solutions to the OPP, after k BLDM iterations, our strategy is to construct approximate solutions defined by $X^k = \bar{V}_kY$. To describe how this is made notice that (23), (24) and (25) show that $\|AX^k - B\|_F = \|T_kY - \bar{U}_{k+1}^TB\|_F$. Furthermore, since $X^TX = I$ if and only if $Y^TY = I$, the solution of (22) can be approximated by $X_k = \bar{V}_kY_k$, where Y_k is solution of

$$\min \|T_kY - \bar{U}_{k+1}^TB\|_F^2, \text{ s.t. } Y^TY = I \text{ and } Y \in \mathbb{R}^{kp\times p}. \tag{26}$$

Note that as k increases, the dimension of the column subspace of $\bar{V}_k$ increases and hence, thanks to the excellent convergence approximation properties of BLDM [14,21], the approximation X_k gets better. We expect to obtain convergence (within a proper tolerance) in a few BLDM iterations. Furthermore, in order to enhance performance, instead of (26) we solve

$$\min \|\Sigma_k\bar{Y} - P_k^T\bar{U}_{k+1}^TB\|_F^2, \text{ s.t. } \bar{Y}^TY = I \text{ and } \bar{Y} \in \mathbb{R}^{kp\times p}$$

where $T_k = P_k\Sigma_kQ_k^T$ is the SVD of matrix T_k and $\bar{Y} = Q_k^TY$.

In order to overcome difficulties associated with the dimension of the problem, in this part of the work we propose a method for solving (22) by combining both Algorithms 2 and 3. Therefore, at each step k of BLDM we apply the Non-monotone Spectral Projected Gradient version (Algorithm 2) for solving (26). This scheme will be called hereafter as PBLDM. At each iteration, the feasible initial guess for solving (26) is taken to be $\begin{bmatrix} Y_{k-1}^T & 0 \end{bmatrix}^T \in \mathbb{R}^{kp\times p}$, except for $k = 1$ where we use the left singular vector of T_1. It is worth mentioning that, for this class of problems, subproblem (20) can be exactly solved by computing a SVD decomposition of T_k (see [14] for further details).

The matrices A and B used in the numerical tests for the OPP were generated as follows. We considered $A = PSR^T$, where both P and R are randomly generated orthogonal matrices and S is diagonal. Three examples are considered:

Table 3
Results for Example 1.

Size ($p = 10$)	PBLDM						PSVD	
	CPU	Residual	Blk(k)	T_k	MaxI	MinI	CPU	Its.
$m = 500$	0.14	4.4×10^{-10}	6	60	8	6	0.72	12
$m = 1000$	0.57	1.6×10^{-10}	6	60	7	6	5.28	12
$m = 5000$	14.25	1.1×10^{-10}	6	60	9	6	592.49	12

Table 4
Results for Example 2.

Size ($p = 5$)	PBLDM						PSVD	
	CPU	Residual	Blk(k)	T_k	MaxI	MinI	CPU	Its.
$m = 100$	0.76	4.1×10^{-13}	20	100	2000	42	0.19	782
$m = 500$	57.51	1.4×10^{-10}	99	495	2000	28	5.65	1234
$m = 1000$	214.43	1.8×10^{-10}	152	760	2000	41	31.05	1484

Table 5
Results for Example 3.

Size ($p = 5$)	PBLDM						PSVD	
	CPU	Residual	Blk(k)	T_k	MaxI	MinI	CPU	Its.
$m = 50$	0.03	1.2×10^{-8}	10	50	30	6	0.02	127
$m = 95$	0.07	2.3×10^{-4}	17	85	44	6	0.11	484
$m = 500$	0.54	3.4×10^{-5}	21	105	30	6	4.69	1001

Example 1. The elements on the main diagonal of S are randomly and uniformly distributed in the interval $[10, 12]$. This is a well-conditioned problem.

Example 2. In this case, $S_{ii} = 1 + \frac{99(i-1)}{(m-1)} + 2r_i$ and r_i are random numbers chosen from a uniform distribution on the interval $[0, 1]$.

Example 3. Matrix S is defined using the MATLAB functions `ones` and `rand`, such that

```
diag(S) = [10*ones(1,m1)+rand(1,m1), 5*ones(1,m2)+rand(1,m2),
           2*ones(1,m3)+rand(1,m3), rand(1,m4)/1000]
```

with $m1 + m2 + m3 + m4 = m$. Thus, A has several small singular values and it is ill-conditioned.

The above test problems were solved for several dimensions with $\eta = 0.85$. Tables 3, 4 and 5 display the results obtained with PBLDM (including re-orthogonalizations in U_k and V_k). Also, for the sake of comparison, we display results obtained with the original algorithm, i.e., with Algorithm 2 applied to (22) with no Lanczos scheme, which we will refer to as PSVD. For Example 3 we use three different values of m, namely, $m = 50$ ($m_1 = 15$, $m_2 = 15$, $m_3 = 12$, $m_4 = 8$), $m = 95$ ($m_1 = 30$, $m_2 = 30$, $m_3 = 30$, $m_4 = 5$) and $m = 500$ ($m_1 = 160$, $m_2 = 160$, $m_3 = 160$, $m_4 = 20$). In all cases the iterative process stops when the first order optimality condition is satisfied with tolerance 10^{-3} with respect to Frobenius norm (see [12] for details on stopping criteria). The tables also include the CPU time (in seconds) as well as the final residuals obtained for each instance, i.e. the value of $\|AX - B\|_F^2$. In the PSVD case we list the total number of iterations performed, but in the PBLDM case we chose to list the number of BLDM steps needed to solve the problem (labeled as Blk). Column T_k lists the number of columns of matrix T_k of (26) at the last PBLDM iteration. Further, for every test problem we display both the maximum (MaxI) and the minimum (MinI) number of iterations of Algorithm 2 for solving (26) (in this case the tolerance parameter to declare convergence was set to 10^{-4}). Regarding Algorithm 3, we note that the lower k is, the lower the order of matrix Y and therefore the lower the computational effort required to solve the problem (26).

From the tables it is seen that the performance of our approach improves significantly as the number of variables (m) gets larger. In this regard, it is worth observing that the number of BLDM steps required for convergence is closely related to the singular value distribution of matrix A, and an interesting conclusion drawn from our numerical experiments is that PBLDM works well on problems where the singular values of matrix A are clustered. Fig. 2(a) shows the singular value distribution (normalized by the maximum singular value) and Fig. 2(b) displays the performance of PBLDM for each example tested with $m = 500$ (y-axis shows the Frobenius norm of the first-order optimality condition). In Example 2 the singular values are uniformly distributed and, in fact, in this case PBLDM does not work well; on the other hand, in Example 1 and Example 3, for which matrix A has clustered singular values (Example 1 with one cluster and Example 3 with four clusters), our approach performs well for large problems. We observe that the presence of ill-conditioning in Example 3 (about 10^5) does not harm convergence of PBLDM; this is not the case with PSVD whose convergence speed is deteriorated, as seen from the number of iterations required for convergence. The close relationship between the number of steps of the PBLDM

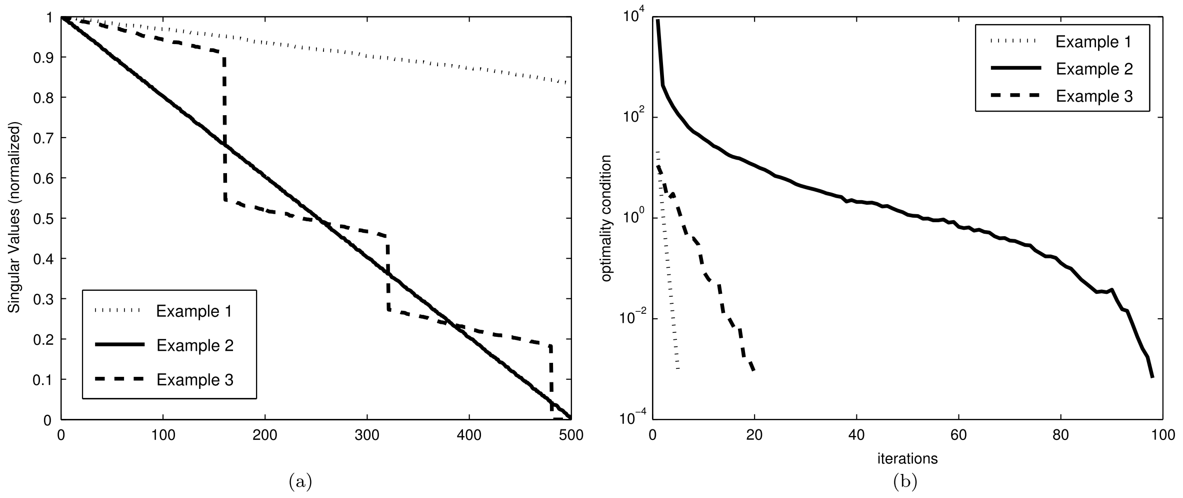


Fig. 2. Singular values distributions versus performance of Algorithm PBLDM with $m = 500$. (Condition numbers of A. Example 1: $k_2(A) = 1.99$. Example 2: $k_2(A) = 5.33 \times 10$. Example 3: $k_2(A) = 1.38 \times 10^6$.)

and the number of clusters is illustrated in Fig. 2(b). Summarizing, the conclusion drawn from the numerical experiments is that in general the block strategy accelerates convergence and uses less computational resources. By way of illustration, for Example 1 with $m = 500$, while PSVD deals with 50×10^4 variables, the last PBLDM iteration deals only with 6×10^2, which results in a reduction in the number of variables of about 99%. In Example 2 with $m = 500$, the reduction is about 79%. However, this is not the case with Example 3 for which significantly more blocks are required until useful information about the problem is captured. In conclusion, the block strategy appears to work better than the original full strategy (PSVD) for a number of problems, which we feel justifies this investigation and might contribute to the development of faster and more efficient methods for OPPs.

5. Conclusions

In this paper we proposed a non-monotone algorithm to minimize a continuous differentiable function over an arbitrary closed set. Our strategy combines regularization techniques and the non-monotone approach of Zhang and Hager [35]. Under some usual nonlinear programming assumptions we have proved globally convergence to stationary points, irrespective of the initial guess. As a particular case of the proposed algorithm, we obtained a Non-monotone Spectral Projected Algorithm for minimization over closed sets. It is worthwhile mentioning that the non-monotone strategy was essential to reduce the number of subproblems resolutions (projection onto feasible set for the especial case) as well as to avoid local minima. Numerical results confirmed the theoretical properties on a class of problems from CUTEst collection. Furthermore, a numerical technique for large-scale orthogonal Procrustes Problem based on Block Lanczos bidiagonalization has been proposed with promising perspective as shown by preliminary numerical results.

Acknowledgement

The authors would like to thank the reviewers for their thorough consideration of our article and for their insightful observations.

References

[1] T.A. Arias, A. Edelman, S.T. Smith, The geometry of algorithms with orthogonality constraints, SIAM J. Matrix Anal. Appl. 20 (1998) 303–353.
[2] J. Baglama, L. Reichel, Restarted block Lanczos bidiagonalization methods, Numer. Algorithms 43 (2006) 251–272.
[3] J. Baglama, L. Reichel, An implicitly restarted block Lanczos bidiagonalization method using Leja shifts, BIT Numer. Math. 53 (2013) 285–310.
[4] J. Barzilai, J.M. Borwein, Two point step size gradient methods, IMA J. Numer. Anal. 8 (1988) 141–148.
[5] T. Bell, Global positioning system-based attitude determination and the orthogonal Procrustes problem, J. Guid. Control Dyn. 26 (2003) 820–822.
[6] E.G. Birgin, J.M. Martínez, M. Raydan, Inexact spectral projected gradient methods on convex sets, IMA J. Numer. Anal. 23 (2003) 539–559.
[7] A.W. Bojanczyk, A. Lutoborski, The Procrustes problem for orthogonal Stiefel matrices, SIAM J. Sci. Comput. 21 (1999) 1291–1304.
[8] M.T. Chu, N.T. Trendafilov, The orthogonally constrained regression revisited, J. Comput. Graph. Stat. 10 (2001) 746–771.
[9] Z. Cui, B. Wu, A new modified nonmonotone adaptive trust region method for unconstrained optimization, Comput. Optim. Appl. 53 (2012) 795–806.
[10] Y.H. Dai, On the nonmonotone line search, J. Optim. Theory Appl. 112 (2002) 315–330.
[11] E.D. Dolan, J.J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2002) 201–213.
[12] J.B. Francisco, F.S. Viloche Bazán, Nonmonotone algorithm for minimization on closed sets with application to minimization on Stiefel manifolds, J. Comput. Appl. Math. 236 (2012) 2717–2727.
[13] J. Fu, W. Sun, Nonmonotone adaptive trust-region method for unconstrained optimization problems, Appl. Math. Comput. 163 (2005) 489–504.
[14] G.H. Golub, C.F. Van Loan, Matrix Computations, third edition, The Johns Hopkins University Press, London, 1996.

[15] N.I.M. Gould, D. Orban, P.L. Toint, Galahad, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization, ACM Trans. Math. Softw. 29 (2004) 353–372.

[16] N.I.M. Gould, D. Orban, P.L. Toint, CUTEst: a constrained and unconstrained testing environment with safe threads, Technical Report RAL-TR-2013-005, Rutherford Appleton Laboratory, 2013.

[17] J. Gower, Orthogonal and projection Procrustes analysis, in: W. Krzanowski (Ed.), Recent Advances in Descriptive Multivariate Statistics, Oxford University Press, Oxford, 1995, pp. 113–134.

[18] L. Grippo, F. Lampariello, S. Lucidi, A nonmonotone line search technique for Newton's method, SIAM J. Numer. Anal. 23 (1986) 707–716.

[19] L. Grippo, F. Lampariello, S. Lucidi, A class of nonmonotone stabilization methods in unconstrained optimization, Numer. Math. 59 (1991) 779–805.

[20] N.J. Higham, The symmetric Procrustes problem, BIT Numer. Math. 28 (1988) 133–143.

[21] S. Karimi, F. Toutounian, The block least squares method for solving nonsymmetric linear systems with multiples right-hand sides, Appl. Math. Comput. 177 (2006) 852–862.

[22] J. Mo, C. Liu, S. Yan, A nonmonotone trust region method based on nonincreasing technique of weighted average of successive function values, J. Comput. Appl. Math. 209 (2007) 97–108.

[23] A. Nemirovski, Sums of random symmetric matrices and quadratic optimization under orthogonality constraints, Math. Program. 109 (2007) 283–317.

[24] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.

[25] T. Rapcsák, On minimization on Stiefel manifolds, Eur. J. Oper. Res. 143 (2002) 365–376.

[26] M. Raydan, The Barzilai and Borwein gradient method for large scale unconstrained minimization problem, SIAM J. Optim. 7 (1997) 26–33.

[27] A. Shapiro, F. Al-Khayyal, First-order conditions for isolated locally optimal solutions, J. Optim. Theory Appl. 77 (1993) 189–196.

[28] W. Sun, Nonmonotone trust region method for solving optimization problems, Appl. Math. Comput. 156 (2004) 159–174.

[29] W.Y. Sun, J.Y. Han, J. Sun, Global convergence of non-monotone descent methods for unconstrained optimization problems, J. Comput. Appl. Math. 146 (2002) 89–98.

[30] P.L. Toint, An assessment of non-monotone linesearch techniques for unconstrained optimization, SIAM J. Sci. Comput. 17 (1996) 725–739.

[31] P.L. Toint, Non-monotone trust region algorithm for nonlinear optimization subject to convex constraints, Math. Program. 77 (1997) 69–94.

[32] N.T. Trendedafilov, On the l_1 Procrustes problem, Future Gener. Comput. Syst. 19 (2003) 1177–1186.

[33] Z. Wen, W. Yin, A feasible method for optimization with orthogonality constraints, Technical Report TR10-26, Rice University CAAM, Rice University, 2010.

[34] Z. Yu, Solving bound constrained optimization via a new nonmonotone spectral projected gradient method, Appl. Numer. Math. 58 (2008) 1340–1348.

[35] H. Zhang, W. Hager, A nonmonotone line search technique and its application to unconstrained optimization, SIAM J. Optim. 14 (2004) 1043–1056.

[36] Z. Zhang, K. Du, Successive projection method for solving the unbalanced Procrustes problem, Sci. China Ser. A 49 (2006) 971–986.

[37] Q. Zhou, D. Hang, Nonmonotone adaptive trust region method with line search based on new diagonal updating, Appl. Numer. Math. 91 (2015) 75–88.