Augmented Lagrangian Algorithms based on the Spectral Projected Gradient for solving nonlinear programming problems

M. A. Diniz-Ehrhardt [*] M. A. Gomes-Ruggiero [†] J. M. Martínez [‡]
S. A. Santos [§]

Abstract

The Spectral Projected Gradient method (`SPG`) is an algorithm for large-scale bound-constrained optimization introduced recently by Birgin, Martínez and Raydan. It is based on Raydan's unconstrained generalization of the Barzilai-Borwein method for quadratics. The `SPG` algorithm turned out to be surprisingly effective for solving many large-scale minimization problems with box constraints. Therefore, it is natural to test its performance for solving the subproblems that appear in nonlinear programming methods based on augmented Lagrangians. In this work, augmented Lagrangian methods which use `SPG` as underlying convex-constraint solver are introduced (`ALSPG`), and the methods are tested in two sets of problems. First, a meaningful subset of large-scale nonlinearly constrained problems of the `CUTE` collection is solved and compared with the performance of `LANCELOT`. Second, a family of localization problems in the minimax formulation is solved against the package `FFSQP`.

Key Words: Augmented Lagrangian methods, Projected gradients, nonmonotone line search, large-scale problems, bound-constrained problems, Barzilai-Borwein method.

[*]Associate Professor, Departamento de Matemática Aplicada, IMECC-UNICAMP, CP 6065, 13081-970 Campinas SP, Brazil (cheti@ime.unicamp.br)

[†]Associate Professor, Departamento de Matemática Aplicada, IMECC-UNICAMP, CP 6065, 13081-970 Campinas SP, Brazil (marcia@ime.unicamp.br)

[‡]Professor, Departamento de Matemática Aplicada, IMECC-UNICAMP, CP 6065, 13081-970 Campinas SP, Brazil (martinez@ime.unicamp.br)

[§]Associate Professor, Departamento de Matemática Aplicada, IMECC-UNICAMP, CP 6065, 13081-970 Campinas SP, Brazil (sandra@ime.unicamp.br)

Resumo

O método do gradiente espectral projetado (SPG) é um algoritmo para problemas de minimização em caixas, de grande porte, introduzido recentemente por Birgin, Martínez e Raydan. É baseado na generalização proposta por Raydan, para minimização sem restrições, do método de Barzilai-Borwein para quadráticas. O algoritmo SPG mostrou–se surpreendentemente eficiente para resolver vários problemas de minimização em caixas, de grande porte. Isto nos levou a testar seu desempenho na resolução de subproblemas que aparecem em métodos de programação não–linear baseados em Lagrangeano aumentado. Assim, neste trabalho, introduzimos o algoritmo ALSPG – um método de Lagrangeano aumentado que usa SPG para resolver subproblemas com restrições convexas. Realizamos vários experimentos numéricos, que podem ser divididos em dois conjuntos. O primeiro é um subconjunto significativo de problemas da coleção CUTE, cujos resultados comparamos com os obtidos pelo software LANCELOT. O outro é composto por uma família de problemas de localização, escritos com formulação minimax. O desempenho de ALSPG para esta classe de problemas foi comparado com o obtido pelo pacote FFSQP.

1 Introduction

In a recent paper (Ref. 1), Birgin, Martínez and Raydan introduced the Spectral Projected Gradient method (`SPG`) for continuous optimization with convex constraints. This algorithm is based on Raydan's unconstrained generalization of the Barzilai-Borwein method for quadratics (see Ref. 2, 3, 4). Consider the problem

$$\text{Minimize } F(x) \text{ subject to } \ x \in \Omega, \tag{1}$$

where F has continuous first partial derivatives and Ω is closed and convex. Given the current point $x^k \in \Omega$, `SPG` computes the search direction $d^k \in I\!R^n$ as

$$d^k = P(x^k - \sigma_k \nabla F(x^k)) - x^k, \tag{2}$$

where $P(z)$ is the orthogonal projection of z on Ω and σ_k is the spectral scaling parameter. The new point $x^{k+1} = x^k + t_k d^k$ is chosen in order to satisfy a nonmonotone sufficient descent condition (see Ref. 5). The method `SPG` is especially useful when the projections on Ω are easy to compute, for example, when Ω is a box or a ball. In these cases, this method is extremely easy to code and its memory requirements are minimal. Surprisingly, its numerical performance is very good, when compared with sophisticated trust-region algorithms. `SPG`'s algorithms have been successful in many problems both academic (see Ref. 1, 6, 7) and industrial (see Ref. 8, 9, 10, 11, 12).

The fact that `SPG` is very easy to code is important in practical situations. In some engineering applications the objective function F has been already coded in unusual computer languages, and it is better to write an `SPG` code in that language than to rely on not always effective interface software. The code, on the other hand, is very short and suitable for microcomputers. Finally, it is well known that the main obstacle for the popularization of parallel computer architectures is the necessity of developing specific architecture-oriented software for mathematical problems. Clearly, simple algorithms make this task easier.

This state of facts motivated us to use `SPG` within the augmented Lagrangian framework for solving

$$\text{Minimize } f(x) \text{ subject to } \ h(x) = 0, \ \ x \in \Omega, \tag{3}$$

where $f : I\!R^n \to I\!R$ and $h : I\!R^n \to I\!R^m$ have continuous first derivatives and the set Ω is given by:

$$\Omega = \{x \in I\!R^n \ \mid \ c_i(x) \le 0, i = 1, \ldots, p\},$$

where the functions c_i are continuously differentiable and convex. At each outer iteration of the augmented Lagrangian scheme for solving (3) one finds an approximate solution of

$$\text{Minimize}_x \ \mathcal{L}(x, \lambda, \rho) \ \text{ subject to } \ x \in \Omega, \tag{4}$$

where

$$\mathcal{L}(x, \lambda, \rho) = f(x) + \langle h(x), \lambda \rangle + \frac{\rho}{2} \|h(x)\|^2 \tag{5}$$

is the Augmented Lagrangian function, $\lambda \in I\!R^m$ is an estimate of the vector of Lagrange multipliers, $\rho > 0$ is the penalty parameter, $\langle \cdot, \cdot \rangle$ is the Euclidean inner product and $\| \cdot \|$ is the Euclidean norm (see Ref. 13, 14, 15, 16).

In the present research our proposal is to solve (4) using SPG. We want to assess the performance of this combination using two families of problems: a representative set of large-scale nonlinear programming problems of the CUTE colection (Ref. 17) and localization problems in the minimax formulation.

This paper is organized as follows. In Section 2 we describe our SPG implementation. In Section 3 we present the augmented Lagrangian algorithm, and in Section 4 its global convergence properties are analysed. The numerical experiments are described and discussed in Section 5, where problems from CUTE collection are compared with LANCELOT (Ref. 15), and localization problems are solved also by FFSQP (Ref. 18). Conclusions and perspectives are presented in Section 6.

2 The SPG method

In (Ref. 1) two spectral gradient methods are presented, called SPG1 and SPG2 respectively. The performance of SPG2 turned out to be better than the one of SPG1, so, SPG2 is the algorithm used in our research and it is called SPG here. When applied to (1), SPG generates a sequence of feasible points $x^k \in \Omega$, $k = 0, 1, 2, \ldots$, where x^0 is given. SPG uses the algorithmic parameters M (a positive integer), $\alpha \in (0, 1)$ (the sufficient decrease parameter), and $0 < \sigma_{min} < \sigma_{max} < \infty$ (safeguarding parameters). As in the Introduction, P denotes the orthogonal projection operator on Ω.

Assume that $x^k \in \Omega$ is the current approximation to the solution of (1) and that x^k is not stationary, so $P(x^k - \nabla F(x^k)) - x^k \neq 0$. Assume that $\sigma_{min} \leq \sigma_k \leq \sigma_{max}$. If x^k is stationary, we say that SPG terminates at x^k. The steps for obtaining x^{k+1} are given in the following algorithm.

Algorithm SPG.

Step 1. Compute $d^k = P(x^k - \sigma_k \nabla F(x^k)) - x^k$.
Step 2. Define $\phi_k = \max\{F(x^k), F(x^{k-1}), \ldots, F(x^j)\}$, where $j = \max\{0, k - M + 1\}$. Set $t \leftarrow 1$.
Step 2.1. Test the sufficient descent condition

$$F(x^k + td^k) \leq \phi_k + \alpha t \langle \nabla F(x^k), d^k \rangle. \tag{6}$$

If (6) does not hold, compute $t_{new} \in [0.1t, 0.9t]$; set $t \leftarrow t_{new}$ and repeat the test (6).
Step 2.2. Define

$$x^{k+1} = x^k + td^k,$$

$$s^k = x^{k+1} - x^k, \;\; y^k = \nabla F(x^{k+1}) - \nabla F(x^k).$$

If $\langle s^k, y^k \rangle \leq 0$ define $\sigma_{k+1} = \sigma_{max}$. Otherwise, define

$$\sigma_{k+1} = \max\left\{\min\left\{\frac{\langle s^k, s^k\rangle}{\langle s^k, y^k\rangle}, \sigma_{max}\right\}, \sigma_{min}\right\} \tag{7}$$

and finish the iteration.

When σ_{k+1} is not safeguarded (so $\sigma_{k+1} \in (\sigma_{min}, \sigma_{max})$) we have that

$$\frac{1}{\sigma_{k+1}} = \frac{\langle s^k, y^k \rangle}{\langle s^k, s^k \rangle} = \frac{\langle s^k, \int_0^1 \nabla^2 F(x^k + ts^k) dt \; s^k \rangle}{\langle s^k, s^k \rangle}.$$

So, $\frac{1}{\sigma_{k+1}}$ is a Rayleigh quotient relative to the average Hessian $\int_0^1 \nabla^2 F(x^k + ts^k)dt$ and, in consequence, is between the smallest and the largest eigenvalue of this matrix. Thus, it can be considered that $\frac{1}{\sigma_{k+1}} I$ is the matrix of the form $\frac{1}{\sigma} I$ that better approximates the average Hessian. Therefore, one should develop heuristics for choosing σ_0, since, in general, the necessary information for such estimative is not available at the initial point.

3 The Augmented Lagrangian Algorithm

In this section we consider problem (3). Given $x^{k-1} \in \Omega$, the current approximation to the solution of (3), a bound $L > 0$, $\lambda^k \in I\!R^m$, the current approximation to the multipliers ($\|\lambda^k\| \leq L$) and $\rho_k > 0$, the penalty parameter, the augmented Lagrangian (or outer) iteration that leads to x^k is defined by Algorithm `ALSPG` described below. Assume that ε_k and η_k are two positive sequences that tend to zero.

Algorithm `ALSPG`

Step 1. *Address subproblem*

Solve problem (4) approximately, with $\lambda = \lambda^k$ and $\rho = \rho_k$, using `SPG` with $F(x) = \mathcal{L}(x, \lambda^k, \rho_k)$ and taking x^{k-1} as initial estimate for `SPG` iterations. The approximate solution must be such that

$$\|P(x^k - \nabla \mathcal{L}(x^k)) - x^k\| \leq \varepsilon_k. \tag{8}$$

Define x^k as the approximate solution of (4) so far obtained by `SPG`.

Step 2. *Update the multipliers*

Choose a new vector of multipliers λ^{k+1} such that $\|\lambda^{k+1}\| \leq L$.

Step 3. *Update the penalty parameter*

If

$$\|h(x^k)\|_\infty \leq 0.1 \|h(x^{k-1})\|_\infty \tag{9}$$

or

$$\|h(x^k)\|_\infty \leq \eta_k \tag{10}$$

define

$$\rho_{k+1} = \rho_k. \tag{11}$$

Otherwise, define

$$\rho_{k+1} = 10\rho_k. \tag{12}$$

4 Global Convergence of Algorithm ALSPG

In this section we prove two theoretical results concerning the global convergence of algorithm ALSPG, under regularity assumptions on the feasible set. These results actually do not rely on the SPG inner solver, but solely depend on the approximate minimization criterion (8) and on the updatings of Steps 2 and 3.

Theorem 4.1 *Assume that x^* is a limit point of a sequence generated by algorithm* ALSPG *and $\{\nabla c_i(x) \mid c_i(x) = 0\}$ is linearly independent for all $x \in \Omega$. Then, x^* is a first-order stationary point of the problem*

$$\textit{Minimize } \|h(x)\|^2 \quad \textit{subject to} \quad x \in \Omega. \tag{13}$$

Proof. Let K_1 be an infinite subset of $I\!\!N$ such that $\lim_{k\in K_1} x^k = x^*$. Since $x^k \in \Omega$ for $k = 1, 2, \ldots$ then $x^* \in \Omega$.

Suppose, first, that there exists $k_0 \in I\!\!N$ such that for all $k \geq k_0$, $\rho_{k+1} = \rho_k$. By (9) and (10) this implies that $\lim_{k\in K_1} h(x^k) = 0$. Therefore, $h(x^*) = 0$ and, so, the thesis is true.

Now, suppose that $\lim_{k\to\infty} \rho_k = \infty$. We have

$$\lim_{k\in K_1} \left\| P\left(x^k - \nabla f(x^k) - \sum_{i=1}^m \lambda_i^k \nabla h_i(x^k) - \rho_k h'(x^k)^T h(x^k)\right) - x^k \right\| = 0.$$

Without loss of generality, assume that $q \leq p$,

$$c_i(x^*) = 0 \text{ for } i = 1, \ldots, q,$$

$$c_i(x^*) < 0 \text{ for } i = q+1, \ldots, p,$$

and there exists $k_1 \in K_1$ such that

$$c_i(x^k) < 0 \text{ for } i = q+1, \ldots, p \text{ for all } k \in K_1, k \geq k_1.$$

Define

$$y^k = x^k - \nabla f(x^k) - \sum_{i=1}^m \lambda_i^k \nabla h_i(x^k) - \rho_k h'(x^k)^T h(x^k).$$

Then $P(y^k)$ is a solution of the convex problem

$$\text{Minimize } \|y - y^k\|^2 \quad \text{subject to} \quad y \in \Omega. \tag{14}$$

Since $\lim_{k\in K_1} x^k = x^*$ and $\lim_{k\in K_1} \|P(y^k) - x^k\| = 0$, we have that $\lim_{k\in K_1} \|P(y^k) - x^*\| = 0$. Therefore, without loss of generality, we can assume that

$$\lim_{k\in K_1} c_i(P(y^k)) = 0 \text{ for } i = 1, \ldots, q, \quad \text{and}$$

$$c_i(P(y^k)) < 0 \text{ for } i = q+1, \ldots, p, \text{ for all } k \in K_1, k \geq k_1.$$

Thus, by the KKT conditions of (14), we have that, if $P(y^k) = w^k$, then

$$2(w^k - y^k) + \sum_{i=1}^{q} \beta_i^k \nabla c_i(w^k) = 0 \tag{15}$$

for all $k \in K_1, k \geq k_1$ and for some $\beta_i^k \geq 0$, $i = 1, 2, \dots, q$.

Dividing (15) by ρ_k and since $\{x^k\}$, $\{P(y^k)\}$ and $\{x^k - \nabla f(x^k) - \sum_{i=1}^m \lambda_i^k \nabla h_i(x^k)\}$ are bounded for $k \in K_1$, we obtain that

$$\lim_{k \in K_1} \left[2h'(x^k)^T h(x^k) + \sum_{i=1}^{q} \frac{\beta_i^k}{\rho_k} \nabla c_i(w^k) \right] = 0.$$

Therefore,

$$\lim_{k \in K_1} \left[\nabla \left(\|h(x^k)\|^2 \right) + \sum_{i=1}^{q} \frac{\beta_i^k}{\rho_k} \nabla c_i(w^k) \right] = 0. \tag{16}$$

Let us call $\gamma_i^k = \frac{\beta_i^k}{\rho_k}$ and $\gamma^k = \max\{\gamma_1^k, \dots, \gamma_q^k\}$. If $\lim_{k \in K_1} \gamma^k = 0$, we have that $\nabla \left(\|h(x^*)\|^2 \right) = 0$ and we are done. Otherwise, assume without loss of generality that $\gamma^k \geq \underline{\gamma} > 0$ for all $k \in K_1$. Dividing (16) by γ^k we get

$$\lim_{k \in K_1} \left[\frac{1}{\gamma^k} \nabla \left(\|h(x^k)\|^2 \right) + \sum_{i=1}^{q} \frac{\gamma_i^k}{\gamma^k} \nabla c_i(w^k) \right] = 0. \tag{17}$$

If $\gamma^k \to \infty$ this contradicts linearly independence of $\{\nabla c_i(x^*)\}_{i=1}^q$. Therefore, there exists $\overline{\gamma} > 0$ such that $\gamma^k \leq \overline{\gamma}$ for all $k \in K_1$, after possibly relabelling. Then, taking convergent subsequences of γ_i^k and taking limits in (17) we obtain

$$\nabla \left(\|h(x^*)\|^2 \right) + \sum_{i=1}^{q} \beta_i^* \nabla c_i(x^*) = 0$$

for some $\beta_i^* \geq 0$, $i = 1, 2, \dots, q$. This implies that x^* is a stationary point of (13). $\square$

The proof of Theorem 4.1 depends strongly on the boundedness of the multiplier estimates λ^k. This property is not automatically satisfied by a formula like

$$\lambda^{k+1} = \lambda^k + \rho_k h(x^k), \tag{18}$$

which represents, in the nonlinear programming terminology, a first-order update of Lagrange multipliers. Higher order (more accurate) updates are possible, but they are computationally more expensive. Since our main interest is on large-scale problems, we adopted the first-order update in our algorithm. However, formula (18) must be modified to avoid unboundedness of λ^k.

An alternative assumption to the boundedness of the multipliers would be that the ratio $\frac{\|\lambda^k\|}{\rho_k}$ goes to zero as ρ_k goes to infinity. In (Ref. 14), for example, as a result of the proposed algorithmic models, such ratio is proved to converge to zero as $\rho_k \to \infty$ for any multiplier updating.

To complete the global convergence analysis we prove Theorem 4.2 in the following. Since we already know that limit points of Algorithm `ALSPG` are stationary points of $\|h(x)\|^2$, we can classify them in two families. The first is the class of infeasible stationary points of $\|h(x)\|^2$ on Ω, for which there is nothing to do except to regret their existence. Second, we have the class of feasible limit points. In Theorem 4.2 we prove that if a feasible limit point is regular (the gradients of active constraints are linearly independent), then it is a stationary point of the nonlinear programming problem (3).

Theorem 4.2. *Assume that x^* is a limit point of the sequence generated by algorithm* `ALSPG`*. Suppose that $h(x^*) = 0$ and x^* is regular (the gradients of the active constraints, including those of set Ω, are linearly independent). Then, x^* is a stationary point of (3).*

Proof. Assume, without loss of generality, that

$$c_i(x^*) = 0, \;\; i = 1, \dots, q, \tag{19}$$

$$c_i(x^*) < 0, \;\; i = q+1, \dots, p. \tag{20}$$

By the regularity hypothesis, the columns of $A_* \in \mathbb{R}^{n\times(m+q)}$ are linearly independent, where

$$A_* = (\nabla h_1(x^*), \dots, \nabla h_m(x^*), \nabla c_1(x^*), \dots, \nabla c_q(x^*)). \tag{21}$$

Let K_1 be an infinite subset of $\mathbb{N}$ such that $\lim_{k\in K_1} x^k = x^*$. By (20) and the regularity hypothesis, there exists $k_1 \in \mathbb{N}$ such that

$$c_i(x^k) < 0, \;\; i = q+1, \dots, p, \tag{22}$$

and $A_k \in \mathbb{R}^{n\times(m+p)}$ is full-rank, where

$$A_k = (\nabla h_1(x^k), \dots, \nabla h_m(x^k), \nabla c_1(x^k), \dots, \nabla c_q(x^k)). \tag{23}$$

for all $k \in K_1, k \geq k_1$. Moreover, the Moore-Penrose pseudo-inverse $A_k^\dagger$ is such that

$$\lim_{k\in K_1} A_k^\dagger = A_*^\dagger. \tag{24}$$

For all $k \in \mathbb{N}$ let us define

$$\mu^k = \lambda^k + \rho_k h(x^k). \tag{25}$$

By (8), we have that

$$\left\| P\left(x^k - \nabla(f(x^k) - \sum_{i=1}^m \mu_i^k \nabla h_i(x^k) \right) - x^k \right\| \leq \varepsilon_k. \tag{26}$$

The rest of the proof consists in showing that $\{\mu^k\}$ is bounded, so that we can take limits in (26). For this purpose, define

$$y^k = x^k - \nabla f(x^k) - \sum_{i=1}^m \mu_i^k \nabla h_i(x^k)$$

and

$$w^k = P(y^k).$$

Then, by the KKT conditions associated to the projection, we have that, for $k \in K_1$, $k \geq k_1$,

$$\begin{aligned} 2(w^k - y^k) + \sum_{i=1}^{q} \beta_i^k \nabla c_i(w^k) + \sum_{i=q+1}^{p} \beta_i^k \nabla c_i(w^k) &= 0 \\ \beta_i^k &\geq 0, i = 1, \ldots, p \\ \beta_i^k c_i(w^k) &= 0, i = 1, \ldots, p \\ c_i(w^k) &\leq 0, i = 1, \ldots, p \end{aligned} \tag{27}$$

But $c_i(x^k) < 0$ for $k \in K_1, k \geq k_1, i = q+1, \ldots, p$. So, since $\lim_{k \in K_1}(w^k - x^k) = 0$ we have that there exists $k_2 \geq k_1$ such that

$$c_i(w^k) < 0 \text{ for } k \in K_1, k \geq k_2, i = q+1, \ldots, p.$$

Thus

$$\beta_i^k = 0 \text{ for } k \in K_1, k \geq k_2, i = q+1, \ldots, p.$$

Then, by (27), if $k \in K_1, k \geq k_2$, we have

$$2(w^k - y^k) + \sum_{i=1}^{q} \beta_i^k \nabla c_i(w^k) = 0.$$

Therefore, for $k \in K_1, k \geq k_2$,

$$w^k - x^k + \nabla f(x^k) + \sum_{i=1}^{m} \mu_i^k \nabla h_i(x^k) + \sum_{i=1}^{q} \frac{\beta_i^k}{2} \nabla c_i(w^k) = 0.$$

So,

$$w^k - x^k + \nabla f(x^k) + \widetilde{A}_k \begin{pmatrix} \mu_1^k \\ \vdots \\ \mu_m^k \\ \beta_1^k/2 \\ \vdots \\ \beta_p^k/2 \end{pmatrix} = 0 \tag{28}$$

where $\widetilde{A}_k = \Big(\nabla h_1(x^k), \ldots, \nabla h_m(x^k), \nabla c_1(w^k), \ldots, \nabla c_q(w^k)\Big)$.

Since matrix A_k is full-rank, so is $\widetilde{A}_k$, for $k \in K_1, k \geq k_2$. Therefore, premultiplying (28) by $\widetilde{A}_k^\dagger$ and taking limits, we obtain that $\mu_1^k, \ldots, \mu_m^k$ are convergent to $\mu_1^*, \ldots, \mu_m^*$. This allows us to take limits on both sides of (26), obtaining

$$\left\| P\left(x^* - \nabla f(x^*) - \sum_{i=1}^{m} \mu_i^* \nabla h_i(x^*)\right) - x^* \right\| = 0.$$

This means that x^* solves the problem

$$\text{Minimize}_w \left\| w - x^* + \nabla f(x^*) + \sum_{i=1}^{m} \mu_i^* \nabla h_i(x^*) \right\|^2 \quad \text{subject to} \;\; w \in \Omega. \tag{29}$$

Writing the KKT conditions of problem (29), we obtain the thesis. □

5 Numerical Experiments

5.1 Problems from CUTE collection

In order to assess the performance of algorithm ALSPG, forty nonlinear equality constrained problems with simple bounded variables from the CUTE collection (version May/98) were selected to constitute our first test set. These test problems were divided in small, medium and large ones, with the following features: 60% had both number of variables (n) and number of equality constraints (m) between 1 and 500; 30% had $n \in (500, 3000]$ or $m \in (500, 3000]$ and 10% had $n \in (3000, 5000]$ or $m \in (3000, 5000]$. As a benchmark, these problems were also solved by LANCELOT, an augmented Lagrangian algorithm implemented with a trust-region strategy for addressing the subproblems (see Ref. 15).

The tests were run in Fortran 77 (double precision, −O compiler option), in a Sparc Station Sun Ultra 1. An interface for running ALSPG with the CUTE collection was prepared, assuming that the problems had already been preprocessed, so that nonlinear inequality constraints were turned into equalities by means of additional nonnegative slack variables. Therefore, for choosing our set of test problems, we decided to select problems originally of the form (3).

The initial approximation x^0 was the default of the CUTE set. The algorithmic choices for these tests were: $\lambda^0 = 0 \in I\!R^m$, $\rho_0 = 10$, $\varepsilon_* = 10^{-4}$, $\eta_* = 10^{-4}$, $\eta_0 = 0.1$, $M = 50$, $\alpha = 10^{-4}$, $\sigma_{min} = 10^{-30}$, $\sigma_{max} = 10^{10}$. The initial tolerance ε_0 was chosen as a function of the initial approximation $\|d^0\| = \|P(x^0 - \nabla \mathcal{L}(x^0)) - x^0\|$ and the final tolerance ε_*, computed as follows: given a constant $p_0 \in (0,1)$, we set $\xi = \log_{10} \|d^0\|$, $p = \log_{10} \varepsilon_*$, $\nu = \xi - p_0(\xi - p)$ and defined $\varepsilon_0 = 10^{\nu}$. For generating the sequence of tolerances $\{\varepsilon_k\}$, given a constant $p_1 \in (0,1)$, we compute $\zeta = 10^{p_1(p-\nu)}$ and set $\varepsilon_k = \max\{\zeta \ \varepsilon_{k-1}, \varepsilon_*\}$. In the implementation we used $p_0 = 0.25$ and $p_1 = 0.1$. The sequence of tolerances $\{\eta_k\}$ was updated in Step 3 according to $\eta_{k+1} = \max\left\{\frac{\eta_k}{\rho_{k+1}}, \eta_*\right\}$ if (11) holds or to $\eta_{k+1} = \max\left\{\frac{\eta_0}{\rho_{k+1}}, \eta_*\right\}$ if (12) occurs.

The initial spectral step σ_0 was set by means of an auxiliary initial computation as follows: given x^0, λ^0, ρ_0, we computed $d^0 = P(x^0 - \nabla \mathcal{L}(x^0, \lambda^0, \rho_0)) - x^0$, $\widehat{x} = x^0 + 0.5\frac{\|x^0\|}{\|d^0\|} d^0$, $\widehat{s} = \widehat{x} - x^0$, $\widehat{y} = \nabla \mathcal{L}(\widehat{x}, \lambda^0, \rho_0) - \nabla \mathcal{L}(x^0, \lambda^0, \rho_0)$ and set $\sigma_0 = \langle \widehat{s}, \widehat{s} \rangle / \langle \widehat{s}, \widehat{y} \rangle$.

For LANCELOT we have followed the authors' suggestions (Ref. 15) to define parameters and settings compatible with the ALSPG choices:

- `exact-second-derivatives-used`
- `bandsolver-preconditioned-cg-solver-used 5`
- `exact-Cauchy-point-required`

- `infinity-norm-trust-region-used`
- `gradient-accuracy-required 1.0D−4`
- `constraint-accuracy-required 1.0D−4`
- `trust-region-radius 1.0D+0`
- `maximum-number-of-iterations 1000000`

Results are shown in Table 1, where problems are presented in alphabetical order. For each problem, the figures of the first row correspond to ALSPG and the ones of the second row to LANCELOT. We keep the notation n, m for number of variables and number of equality constraints, respectively. The number of outer iterations of the corresponding augmented Lagrangian algorithm is denoted by `out`; `in` is the number of inner iterations; `feval` the number of functional evaluations and `time` the CPU time in seconds.

Problem	n	m	out	in	feval	fobj	time
ALJAZZAF	3	1	7	112	258	0.7501D+02	0.4865D−02
			6	22	23	0.7501D+02	0.6000D−01
ARGAUSS	3	15	6	7	24	0.0000D+00	0.3275D−02
			1	1	2	0.1180D−07	0.1000D−03
AVION2	49	15	2	675585	1000001	0.1245D+08	0.1413D+03
			9	14013	11549	0.9470D+08	0.5850D+02
BIGBANK	2230	1112	1	787806	1000001	−0.3131D+07	0.1125D+05
			4	49	50	−0.4210D+07	0.1978D+04
BRITGAS	450	360	6	119437	155183	0.0000D+00	0.5849D+03
			6	99	88	0.0000D+00	0.1494D+02
BROYDN3D	5000	5000	15	504	589	0.0000D+00	0.2021D+02
			1	5	6	0.3100D−16	0.7600D+00
BROYDNBD	5000	5000	1	51	60	0.0000D+00	0.3916D+01
			1	31	26	0.1370D-08	0.1224D+02
BT4	3	2	6	81	115	−0.4551D+02	0.3821D−02
			4	22	22	−0.4551D+01	0.6000D−01
CATENARY	501	166	8	183909	239126	−0.3484D+06	0.3798D+03
			4	797	660	−0.3484D+06	0.3058D+02
CLUSTER	2	2	2	17	19	0.0000D+00	0.2341D−02
			1	9	9	0.2820D−06	0.1000D−01
DALLASL	906	667	6	608138	796716	−0.2026D+06	0.1239D+05
			6	81	81	−0.2026D+06	0.1372D+0.3
DITTERT	105	70	6	642	701	−0.1985D+01	0.4784D+00
			9	130	130	−0.2000D+121	0.2470D+01
DIXCHLNV	10	5	7	215	247	0.8226D−10	0.4818D−01
			3	12	12	0.1200D−07	0.5000D−01
DNIEPER	61	24	7	397974	527677	0.1874D+05	0.1276D+03
			6	68	47	0.1870D+05	0.3800D+00
DTOC5	1999	999	7	603180	794567	0.1536D+01	0.6714D+04
			9	21	22	0.1520D+01	0.2580D+01
GILBERT	1000	1	8	41	99	0.4820D+03	0.2848D+00
			5	27	28	0.4820D+03	0.5500D+01

Table 1: Complete comparative results: ALSPG × LANCELOT.

Problem	n	m	out	in	feval	fobj	time
HAGER1	1001	500	6	761763	1000001	0.1133D+01	0.3579D+04
			2	10	11	0.8810D+00	0.8300D+00
HEART6	6	6	2	152959	570414	0.0000D+00	0.1821D+02
			1	1751	1536	0.5660D−05	0.2550D+01
HS111	10	3	6	3257	4093	−0.4776D+02	0.1974D+01
			5	45	44	−0.4776D+02	0.1600D+00
HS41	4	1	6	23	54	0.1926D+01	0.2328D−02
			4	6	7	0.1926D+01	0.3000D−01
HUESTIS	1000	2	11	1045	1083	0.3482D+11	0.4048D+01
			11	151	150	0.3482D+11	0.1128D+03
LCH	600	1	6	1212	1455	−0.4288D+01	0.4030D+01
			5	36	36	−0.4320D+01	0.2570D+01
LEAKNET	156	153	5	761134	1000001	0.7523D+01	0.1389D+04
			11	117	117	0.7960D+01	0.6360D+01
LINSPANH	97	33	6	150	174	−0.7700D+02	0.4247D−01
			5	9	15	−0.7700D+02	0.1700D+00
LOTSCHD	12	7	6	890	1119	0.2398D+04	0.7047D−01
			5	21	22	0.2398D+04	0.1000D+00
METHANB8	31	31	2	749366	1000001	0.0000D+00	0.4768D+03
			1	36	37	0.9370D−05	0.8200D+00
MINC44	1113	1032	6	109	130	0.3158D−03	0.1852D+01
			10	19	20	0.3170D−03	0.5560D+01
MINPERM	583	520	7	152	181	0.9366D−03	0.1155D+01
			9	93	83	0.9450D−03	0.1761D+02
NCVXQP1	100	50	6	1390	1434	−0.7298D+06	0.8129D+00
			7	43	44	−0.7298D+06	0.4200D+00
OPTCNTRL	32	20	6	1310	1469	0.5500D+03	0.1820D+00
			6	21	22	0.5500D+03	0.1100D+00
ORTHRDM2	4003	2000	5	607156	1000001	0.1565D+03	0.2566D+05
			4	129	112	0.1565D+03	0.4318D+02
ORTHREGC	1005	500	6	490278	701241	0.1879D+02	0.5068D+04
			5	43	38	0.1880D+02	0.3580D+01
READING1	202	100	4	773701	1000001	−0.1495D+00	0.1344D+04
			4	741	667	−0.1600D+00	0.2621D+02
SEMICON1	1002	1000	2	716590	1000002	0.0000D+00	0.2004D+05
			1	1401	1195	0.1303D-06	0.3928D+02
SPANHYD	97	33	6	29959	41464	0.2397D+03	0.1068D+02
			4	27	1195	0.2400D+03	0.5600D+00
STEENBRC	540	126	4	872300	1000001	0.3678D+05	0.1637D+04
			8	6446	5240	0.2750D+05	0.1100D+03
TENBARS2	18	8	3	731412	1000002	0.2318D+04	0.7128D+02
			4	346	300	0.2280D+04	0.1590D+01
TRAINF	4008	2002	6	760251	1000001	0.2191D+01	0.1696D+05
			9	58	58	0.2760D+01	0.2103D+03
TRIGGER	7	6	3	999986	1000001	0.0000D+00	0.4449D+02
			1	20	19	0.1020D−06	0.3000D−01
YORKNET	312	256	6	210592	1000001	0.1968D+05	0.7865D+03
			12	236	224	0.1420D+05	0.7071D+02

Table 1 (cont.): Complete comparative results: ALSPG $\times$ LANCELOT.

Twenty-seven out of the forty test problems were successfully solved by algorithm `ALSPG`. For the remaining thirteen the maximum allowed number of functional evaluations (1000000) was exceeded. For two of these thirteen problems (`ORTHRDM2` and `READING1`) the final approximation was nearly optimal (constraint violation less than 10^{-4} and objective function value close to the one obtained by `LANCELOT`). Thus, `ALSPG` had success in 72.5% of the tests (29 in 40). It is worth noticing that the proportion of failures of `ALSPG` followed quite closely the size distribution of the problems: 9% of the large problems, 36% of the medium and 55% of the small ones could not be solved by `ALSPG`. `LANCELOT` performed as follows: thirty four problems were successfully solved (85%) and for five problems it stopped with a too small step (`AVION2`, `DNIEPER`, `HEART6`, `TENBARS2` and `YORKNET`), which correspond to 13% of the tests. For a single problem (`DITTERT`) `LANCELOT` did not converge, that is, no feasible solution could be found. Algorithm `ALSPG` was successful in solving three problems (`DITTERT`, `DNIEPER` and `HEART6`) of the six aforementioned stops of `LANCELOT`.

Summarizing Table 1 using geometric means of the number of the functional evaluations performed and the CPU time spent, we obtained the figures of Table 2. The results of Table 2 allow us to estimate the average time of a single iteration of each algorithm: 0.001 seconds for `ALSPG` and 0.04 seconds for `LANCELOT`. Algorithm `ALSPG` needs to perform approximately 270 times the number of functional evaluations of `LANCELOT`, whereas it takes around 8 times the CPU time spent by the algorithm of Conn, Gould and Toint.

	`feval`	`time`
`ALSPG`	16551.0	16.17
`LANCELOT`	61.608	2.255

Table 2: Summary of comparative results (average values) for problems from `CUTE`.

5.2 The localization problem

Given a family of polytopes $K_i \subset I\!R^2$, $i = 1, \ldots, npol$, the optimal localization problem consists of finding $P \in I\!R^2$ such that its maximum distance to the polytopes is minimum. In other words, P is a solution to

$$\min_{P_i \in K_i} \max\{\|P - P_1\|, \|P - P_2\|, \ldots, \|P - P_{npol}\|\} \tag{30}$$

where $\|\cdot\|$ is the Euclidean norm. A possible optimal configuration with $npol = 4$ is illustrated in Figure 1.

Problem (30) may be rewritten in the format

$$\text{Minimize } z \text{ subject to } \|P - P_i\| \le z, P_i \in K_i, i = 1, \ldots, npol. \tag{31}$$

Introducing positive slack variables, the inequality constraints $\|P - P_i\| \le z$ are turned into equalities $\|P - P_i\| + \theta_i - z = 0$, so that a problem of type (4) is built

$$\begin{gathered}\text{Minimize } z + \sum_{i=1}^{npol} \lambda_i(\|P - P_i\| + \theta_i - z) + \frac{\rho}{2}\sum_{i=1}^{npol}(\|P - P_i\| + \theta_i - z)^2 \\ \text{subject to } \theta_i \ge 0, P_i \in K_i, i = 1, \ldots, npol.\end{gathered} \tag{32}$$

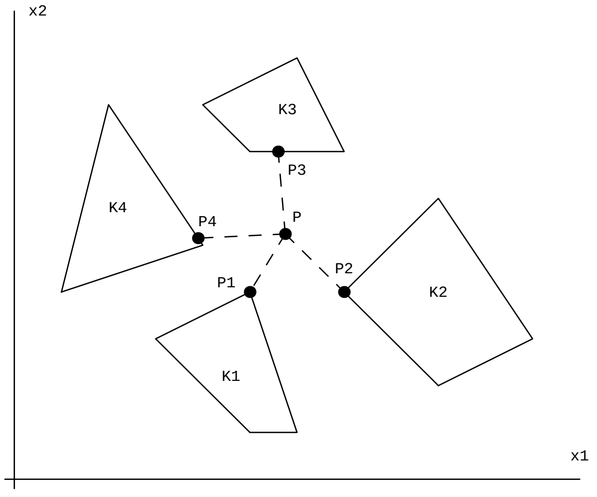

Figure 1: A possible optimal configuration of the localization problem.

In our numerical experiments, we generated a family of twenty medium-size problems of type (30) and compared the performance of solving them by FFSQP (Ref. 18) against solving corresponding problem (31) by Algorithm ALSPG. Code FFSQP is a Fortran implementation of a feasible sequential quadratic programming algorithm for solving constrained nonlinear, possibly minimax, optimization problems. For FFSQP the tolerances were set so that its feasibility and optimality criteria were compatible and comparable with ALSPG choices. The tests were run in Fortran 77 (double precision, −O compiler option), in a Sparc Station Sun Ultra 1.

It is worthwhile noticing that, since the formulation considered by each algorithm is different, ALSPG solves problems with $3npol + 3$ variables ($2npol + 3$ original variables and $npol$ slack ones) and $npol$ constraints whereas FFSQP works with $2npol + 2$ variables and as much constraints as the number of vertices of the problem. The localization problems were randomly generated and were addressed with a different formulation in (Ref. 19).

The initial approximation $(P, P_1, \ldots, P_{npol})$ was obtained by projecting the origin onto an auxiliar centered polygon created during generation of the problem (initial P). The initial values of P_i were set as the projection of such P onto the polygons K_i, $i = i, \ldots, npol$. Variables $\theta_i, i = 1, \ldots, npol$ and z were initially zero. Algorithmic parameter choices for these tests were mostly the same used for the CUTE set of problems, except for $\sigma_{min} = 10^{-6}$, $\sigma_{max} = 10^6$, $\varepsilon_0 = 10^{-2}$ and $\varepsilon_{k+1} = \max\{0.1\varepsilon_k, \varepsilon_*\}$. We also implemented a stopping test to detect lack of progress as follows: we computed $\bar{h}_i = \min\{\|h(x^0)\|, \ldots, \|h(x^i)\|\}$ and stopped if $\bar{h}_{k+1} \geq \bar{h}_k - \max\{10^{-3}\bar{h}_k, 10^{-1}\}$ at fifty consecutive iterations.

Comparative results are presented in Table 3, where column problem provides a number for future reference and the pair (number of polygons, number of vertices) of each problem; for the next columns, the first row corresponds to ALSPG and the second, to FFSQP. We denote by iter the number of outer iterations of each algorithm; feval gives the number of function evaluations; fobj provides the final objective function value (problem (31)) and time gives the CPU time spent in seconds.

Problem	iter	feval	fobj	time
1 (70, 485)	5	18470	30.6309	7.38
	8	560	30.6215	3.29
2 (77, 479)	4	10346	36.7179	5.08
	13	1001	36.7179	7.52
3 (104, 709)	10	11013	37.8149	5.78
	8	835	37.8034	10.75
4 (107, 652)	55	14197	92.1574	8.05
	6	642	72.0387	9.99
5 (116, 717)	6	12446	49.3796	7.88
	7	812	49.3786	13.82
6 (136, 1054)	17	13588	49.1609	11.36
	8	1088	49.1322	21.92
7 (159, 3888)	4	3848	44.7871	8.86
	9	1431	44.7868	86.73
8 (163, 1061)	4	13993	48.3872	13.50
	8	1304	48.3805	35.57
9 (189, 3596)	4	3866	50.0040	10.18
	8	1512	50.0039	92.18
10 (197, 1356)	54	14053	78.9317	14.54
	6	1182	53.8598	50.13
11 (296, 1985)	4	18992	65.8694	30.84
	8	2368	65.8666	237.55
12 (323, 4889)	52	13748	90.1371	42.80
	9	2910	65.7716	436.13
13 (325, 2185)	4	10038	69.8736	25.47
	8	2600	69.8730	295.66
14 (331, 2177)	52	18529	69.5391	31.33
	7	2317	69.5289	291.84
15 (361, 2357)	4	8313	69.4936	20.29
	6	2166	69.7373	336.02
16 (375, 2478)	52	20794	74.8264	40.70
	6	2250	74.8021	374.63
17 (406, 2639)	5	9451	75.2198	25.79
	6	2436	75.2184	468.38
18 (436, 2875)	4	10146	80.4443	30.64
	8	3488	80.4420	786.28
19 (449, 2967)	5	10262	80.2297	31.48
	7	3143	80.2291	706.32
20 (466, 3042)	54	21711	83.7485	54.43
	6	2796	83.7463	695.60

Table 3: Comparative results: `ALSPG` $\times$ `FFSQP`

For six out of the twenty problems of Table 3, algorithm `ALSPG` stopped with lack of progress (problems 4, 10, 12, 14, 16 and 20), which amounts to 70% successful exits for these tests. Algorithm `FFSQP` had two stops with too small step (problems 3 and 12), corresponding to 90% of success. In all these stops, however, a nearly optimal iterate was achieved. For `ALSPG`, the values of $\|h(x^k)\|_\infty$ were $3\times10^{-2}, 2\times10^{-3}, 8\times10^{-2}, 3\times10^{-4}, 4\times10^{-3}$ and 4×10^{-4} at the final approximation for the six aforementioned problems. For `FFSQP`, the norm of the gradient of the Lagrangian at the final iterate was 5×10^{-4} and 2×10^{-4} for problems 3 and 12, respectively.

The results of Table 3 are summarized in Table 4, as we did with Tables 1 and 2. For this set of tests, the estimated average time of a single iteration for each algorithm is 0.001 and 0.06 seconds, for ALSPG and FFSQP, respectively. As a result of the low cost of ALSPG, although our approach needs around eight times the number of functional evaluations taken by FFSQP, algorithm FFSQP needs more than five times as much CPU time as the one spent by ALSPG.

	feval	time
ALSPG	11781.0	16.81
FFSQP	1608.1	92.99

Table 4: Summary of comparative results (average values) for localization problems.

In Table 5 we present results of ALSPG relatively to eight large-scale localization problems, for which FFSQP failed due to memory requirements. The number of variables and of equality nonlinear constraints varied within $n \in [1548, 2817]$ and $m \in [515, 938]$, respectively. The notation is similar to the one of Table 3, except that the outer iterations are given in column out and we also provide the number of inner iterations in column in.

Problem	out	in	feval	fobj	time
21 (515, 14159)	4	6763	9038	85.8409	100.90
22 (640, 4759)	4	12118	16774	92.7432	88.33
23 (646, 12924)	25	15651	21691	102.3918	189.79
24 (677, 5035)	6	11433	15838	96.3327	84.79
25 (734,5442)	4	9733	13595	104.5577	88.46
26 (742, 5519)	3	5984	7951	99.7280	56.78
27 (801, 5955)	5	13107	18495	110.3467	111.92
28 (938, 6985)	23	13268	20006	119.0289	134.15

Table 5: Performance of ALSPG for large-scale localization problems

Analysing Table 5, we defined a measure for the efficiency of the SPG step, given by the arithmetic mean of the values $\texttt{feval}_i/\texttt{in}_i$, $i = 1, 2, \ldots, 8$, which came to 1.4. Ideally, if no rejection occurred at Step 2.1 of algorithm SPG, a single functional evaluation would be done per inner iteration. For this family of tests, in average, less than two functional evaluations are necessary per inner iteration, which indicates a good performance of the SPG step.

6 Final Remarks

We have introduced an augmented Lagrangian algorithm (ALSPG) for which the spectral projected gradient is the tool for tackling the underlying subproblems. Our motivation was the SPG effectiveness for minimization with simple bounds, so we wanted to assess its performance within the augmented Lagrangian framework. For the proposed algorithm we proved global convergence results in the sense that the generated limit points are stationary provided they are regular and feasible with respect to the nonlinear equality constraints.

Two families of test problems were addressed. First, forty nonlinear equality constrained problems with simple bounded variables from the CUTE collection were solved and compared against LANCELOT. Both augmented Lagrangian algorithms had a quite robust performance for such problems: 72.5% were successfully solved by ALSPG and 85% by LANCELOT.

The second set of tests was formed by twenty medium-size localization problems in minimax formulation. After turned into the nonlinear programming equivalent format, with auxiliary and slack variables, they were solved by ALSPG. For comparative purposes, they were solved, in the original formulation, by the code FFSQP. Both strategies performed quite well (70% of reported success for ALSPG and 90% for FFSQP). An additional family of eight large-scale localization problems were solved solely by ALSPG, since FFSQP could not address their large number of variables and/or constraints. For this large-scale set, we observed the efficiency of the SPG inner step, with an average of 1.4 functional evaluations per iteration.

Summing up, the results summarized in Tables 2 and 4 corroborate the low cost features of ALSPG iterations. The first-order features of algorithm ALSPG might require a large number of functional evaluations. Its iterations, however, are very cheap. This easy-to-code algorithm, with minimal memory requirements, available upon request to the authors, might be a worthwhile alternative provided the problem does not have too expensive functional evaluations.

Acknowledgements: We are thankful to E. G. Birgin for making his polygon generator available for our numerical tests with localization problems. We also thank A. R. Conn, N. I. M. Gould and Ph. L. Toint for the package LANCELOT and J. L. Zhou, A. L. Tits and C. T. Lawrence for the code FFSQP.

References

[1] E. G. Birgin, J. M. Martínez and M. Raydan, *Nonmonotone spectral projected gradient methods on convex sets*, SIAM Journal of Optimization, Vol. 10, pp. 1196-1211, 2000.

[2] J. Barzilai and J. M. Borwein, *Two point step size gradient methods*, IMA Journal of Numerical Analysis, Vol. 8, pp. 141-148, 1988.

[3] M. Raydan, *On the Barzilai and Borwein choice of steplength for the gradient method*, IMA Journal of Numerical Analysis, Vol. 13, pp. 321-326, 1993.

[4] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM Journal on Optimization, Vol. 7, pp. 26-33, 1997.

[5] L. Grippo, F. Lampariello and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM Journal on Numerical Analysis, Vol. 23, pp. 707-716, 1986.

[6] B. Molina, S. Petiton and M. Raydan, *An assessment of the preconditioned gradient method with retards for parallel computers*, Computational and Applied Mathematics, to appear.

[7] F. Luengo, W. Glunt, T. L. Hayden and M. Raydan, *Preconditioned spectral gradient method*, SIAM Journal on Scientific Computing, to appear.

[8] E. G. Birgin, R. Biloti, M. Tygel and L. T. Santos, *Restricted optimization: a clue to a fast and accurate implementation of the common reflection surface method*, Journal of Applied Geophysics, Vol. 42, pp. 143-155, 1999.

[9] E. G. Birgin, I. Chambouleyron and J. M. Martínez, *Estimation of optical constants of thin films using unconstrained optimization*, Journal of Computational Physics, Vol. 151, pp. 862-880, 1999.

[10] E. G. Birgin, and Y. G. Evtushenko, *Automatic differentiation and spectral projected gradient methods for optimal control problems*, Optimization Methods and Software, Vol. 10, pp. 125-146, 1998.

[11] Z. Castillo, D. Cores and M. Raydan, *Low cost optimization techniques for solving the nonlinear seismic reflection tomography problem*, Optimization and Engineering, Vol. 1, pp. 155-169, 2000.

[12] M. Mulato, I. Chambouleyron, E. G. Birgin and J. M. Martínez, *Determination of thickness and optical constants of a-Si:H films from transmittance data*, Applied Physics Letters, Vol. 77, pp. 2133-2135, 2000.

[13] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*, Academic Press, New York, 1982.

[14] 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, Vol. 28, pp. 545-572, 1991.

[15] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A), Springer Series in Computational Mathematics 17*, Springer-Verlag, New York, Berlin and Heidelberg, 1992.

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

[17] I. Bongartz, A. R. Conn, N. I. M. Gould and Ph. L. Toint, *CUTE: Constrained and Unconstrained Testing Environment*, ACM Transactions on Mathematical Software, Vol. 21, pp. 123-160, 1995.

[18] J. L. Zhou, A. L. Tits and C. T. Lawrence, *User's Guide for FFSQP Version 3.7: a Fortran code for solving optimization programs, possibly minimax, with general inequality constraints and linear equality constraints, generating feasible iterates*, Technical Report SRC-TR-92-107r5, Institute for Systems Research, University of Maryland, USA, 1997.

[19] E. G. Birgin, J. M. Martínez and M. Raydan, *SPG: software for convex constrained optimization*, Technical Report, Institute of Mathematics, University of Campinas, Brazil, 2000.