Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Lecturas en Ciencias de la Computación
ISSN 1316-6239

A derivative-free nonmonotone line search technique for unconstrained optimization

M.A. Diniz-Ehrhardt, J.M. Martínez y Marcos Raydan

RT-2006-10

Centro de Cálculo Científico y Técnológico
CCCT
Caracas, Octubre 2006

A derivative–free nonmonotone line search technique for unconstrained optimization

M. A. Diniz-Ehrhardt * J. M. Martínez † M. Raydan ‡

October 30, 2006

Dedicated with friendship to Claude Brezinski at the occasion of his retirement

Abstract

A tolerant derivative–free nonmonotone line search technique is proposed and analyzed. Several consecutive increases in the objective function and also non descent directions are allowed for unconstrained minimization. To exemplify the power of this new line search we describe a direct search algorithm in which the directions are chosen randomly. The convergence properties of this random method rely exclusively on the line search technique. We present numerical experiments, to illustrate the advantages of using a derivative-free nonmonotone globalization strategy, with approximated-gradient type methods and also with the inverse SR1 update that could produce non descent directions. In all cases we use a local variation finite differences approximation to the gradient.

Keywords: unconstrained minimization, derivative-free methods, nonmonotone line-search schemes, SR1 updates.

1 Introduction

We propose and analyze a new tolerant and nonmonotone derivative–free line search globalization strategy for the unconstrained minimization problem

$$\text{Minimize } f(x) \quad \text{subject to} \quad x \in I\!R^n, \tag{1}$$

where $f : I\!R^n \to I\!R$ has continuous partial derivatives which are not available.

The optimization problem (1) appears in industrial applications because, quite frequently, the objective function is evaluated through a computer simulation process, and therefore derivatives cannot be evaluated. For example, shape optimization in fluid-dynamics problems has received remarkable attention in recent years (e.g., [2, 18, 22] and references therein). Due to the availability of very efficient commercial and public-domain Computational Fluid Dynamics (CFD) solvers, shape optimization strategies that treat the solver as a black box offer a strong potential. For this type of methods, the CFD solver simply participates during the evaluation of the objective function. In a typical fluid dynamics problem, the values of the pressure drop, the outlet velocity, etc., depend on the fluid properties, the boundary conditions and the boundary shape. In a shape optimization problem, the fluid properties and the boundary conditions are already set, thus, the objective function, represented by a predetermined combination of the above-mentioned fluid-dynamics parameters,

*Department of Applied Mathematics IMECC-UNICAMP, University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. This author was supported by PRONEX-Optimization 76.79.1008-00, FAPESP (Grant 90-3724-6), and CNPq. E-mail: cheti@ime.unicamp.br

†Department of Applied Mathematics IMECC-UNICAMP, University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. This author was supported by PRONEX-Optimization 76.79.1008-00, FAPESP (Grant 90-3724-6), CNPq and FAEP-UNICAMP. E-mail: martinez@ime.unicamp.br

‡Departamento de Computación, Facultad de Ciencias, Universidad Central de Venezuela, Ap. 47002, Caracas 1041-A, Venezuela. Sponsored by the Center of Scientific Computing at UCV. E-mail: mraydan@kuaimare.ciens.ucv.ve

depends on the boundary shape only. Geometry is the input to the black box (CFD solver), the value of the objective function is the output, and derivative information is very hard (or even impossible) to obtain from this computational simulation.

The globalization strategy that we present combines and extends the Grippo, Lampariello, and Lucidi (GLL) [17], the Lucidi and Sciandrone (LSc) [21], and the Li and Fukushima (LF) [20] line search techniques. It also extends similar nonmonotone line search schemes recently proposed [19], for solving large-scale nonlinear systems of equations. The GLL strategy accepts significant consecutive increases in the objective function (nonmonotone behavior), but requires exact gradient information and descent directions to guarantee global convergence. On the other hand, the LF scheme tolerates non descent directions but little or insufficient nonmonotone behavior. Finally, the LSc line search is a monotonic strategy that accepts several directions to be explored simultaneously. For some well-known and also some new numerical methods for unconstrained minimization, the three aspects (non descent directions, nonmonotone behavior, and several directions explored simultaneously) could be of great help and important for good numerical performance. Our new line search scheme, that will be described in Section 2, has these three features.

To illustrate the power of this new line search we describe, in Section 3, a direct search algorithm in which the directions are chosen randomly. The convergence properties of this random method rely exclusively on the line search technique. Some small size numerical experiments are also presented for this case. We present numerical experiments with approximated-gradient type methods (Section 4), and also with the inverse SR1 update [13, 16] (Section 5) that could produce non descent directions. In both cases we report numerical results using a local variation finite differences approximation to the gradient, as discussed and used in [14]. In Section 6 we discuss the observed results, in particular the advantages of using a tolerant derivative-free nonmonotone globalization strategy.

Notation

Throughout the paper $\|\cdot\|$ will be the Euclidian norm although in some cases it can be replaced by an arbitrary norm.

2 Model line-search algorithm and convergence

We denote $g(x) = \nabla f(x)$ for all $x \in I\!R^n$. Let τ_{min}, τ_{max} be such that $0 < \tau_{min} < \tau_{max} < 1$. Let M be a positive integer. Assume that $\{\eta_k\}$ is a sequence such that

$$\eta_k > 0, \quad \text{for all } k = 0, 1, 2, \ldots, \ \sum_{k=0}^{\infty} \eta_k = \eta < \infty$$

and that $\{\beta_k\}$ is a bounded sequence such that $\beta_k > 0$ for all $k \in I\!N$ with the property that, for all infinite subset of indices $K \subset I\!N$,

$$\lim_{k \in K} \beta_k = 0 \Rightarrow \lim_{k \in K} g(x_k) = 0. \tag{2}$$

(Observe that the choice $\beta_k \equiv 1$ is admissible.) Assume that $x_0 \in I\!R^n$ is a given initial point. If $x_k \in I\!R^n$ is the $k-$th iterate computed by the algorithm, the steps for computing x_{k+1} are given below.

Algorithm 1. (Model algorithm)

Step 1. *Compute the directions*

Compute D_k a finite set of $I\!R^n$. (The cardinality of D_k does not need to be constant.)

Define

$$\bar{f}_k = \max\{f(x_k), \ldots, f(x_{\max\{k-M+1,0\}}\}.$$

Step 2. *Backtracking*

Step 2.1. For all $d \in D_k$ set $\alpha(d) \leftarrow 1$.

Step 2.2. Find (if possible) $d \in D_k$ such that the inequality

$$f(x_k + \alpha(d)d) \leq \bar{f}_k + \eta_k - \alpha(d)^2 \beta_k \tag{3}$$

holds. If some d satisfying (3) is found, set $\alpha_k = \alpha(d)$, choose x_{k+1} such that

$$f(x_{k+1}) \leq f(x_k + \alpha_k d), \tag{4}$$

and finish the iteration.

If (3) fails for all $d \in D_k$, compute, for all $d \in D_k$, $\alpha_{new}(d) \in [\tau_{min}\alpha(d), \tau_{max}\alpha(d)]$, set $\alpha(d) \leftarrow \alpha_{new}(d)$ and repeat Step 2.2.

The fact that an iteration of the algorithm is well defined is trivial in this case, because $\eta_k > 0$ guarantees that (3) holds if $\alpha(d)$ is sufficiently small. Observe that the choice $x_{k+1} = x_k + \alpha_k d$ is admissible in (4). We allow the algorithm to make a different choice of x_{k+1} in order to try extrapolation steps.

For proving convergence of the algorithm, we need some definitions to cope with the nonmonotonicity of $f(x_k)$. A similar construction has been proposed in [5] for proving convergence of the inexact SPG method.

Define, for all $\ell = 1, 2, 3, \ldots$, $V_\ell = \max\{f(x_{(\ell-1)M+1}), \ldots, f(x_{\ell M})\}$ and $\nu(\ell) \in \{(\ell-1)M+1, \ldots, \ell M\}$ such that $f(x_{\nu(\ell)}) = V_\ell$.

Clearly,

$$\begin{aligned} f(x_{\ell M+1}) &\leq \max\{f(x_{(\ell-1)M+1}), \ldots, f(x_{\ell M})\} + \eta_{\ell M} - \alpha_{\ell M}^2 \beta_{\ell M} \\ &= V_\ell + \eta_{\ell M} - \alpha_{\ell M}^2 \beta_{\ell M} \leq V_\ell + \eta_{\ell M}, \\ f(x_{\ell M+2}) &\leq \max\{V_\ell, f(x_{\ell M+1})\} + \eta_{\ell M+1} - \alpha_{\ell M+1}^2 \beta_{\ell M+1} \\ &\leq V_\ell + \eta_{\ell M} + \eta_{\ell M+1} - \alpha_{\ell M+1}^2 \beta_{\ell M+1} \leq V_\ell + \eta_{\ell M} + \eta_{\ell M+1}. \end{aligned}$$

So, by an inductive argument, $f(x_{\ell M+j}) \leq V_\ell + \eta_{\ell M} + \ldots + \eta_{\ell M+j-1} - \alpha_{\ell M+j-1}^2 \beta_{\ell M+j-1}$ for all $j = 1, 2, \ldots, M$. But $\nu(\ell+1) \in \{\ell M + 1, \ldots, \ell M + M\}$, therefore,

$$V_{\ell+1} = f(x_{\nu(\ell+1)}) \leq V_\ell + (\eta_{\ell M} + \ldots + \eta_{\ell M+M-1}) - \alpha_{\nu(\ell+1)-1}^2 \beta_{\nu(\ell+1)-1}.$$

So, for all $\ell = 1, 2, \ldots$ we have that

$$f(x_{\nu(\ell+1)}) \leq f(x_{\nu(\ell)}) + (\eta_{\ell M} + \ldots + \eta_{\ell M+M-1}) - \alpha_{\nu(\ell+1)-1}^2 \beta_{\nu(\ell+1)-1}. \tag{5}$$

Using this inequality, we can prove the following proposition:

Proposition 1. *Assume that $\{f(x_k)\}_{k\in\mathbb{N}}$ is bounded below. Then*

$$\lim_{\ell\to\infty} \alpha_{\nu(\ell)-1}^2 \beta_{\nu(\ell)-1} = 0.$$

Proof. It follows from (5) using the summability of η_k and the fact that f is bounded below. □

From now on we define:

$$K = \{\nu(1) - 1, \nu(2) - 1, \nu(3) - 1, \ldots\}. \tag{6}$$

Theorem 1. *Assume that $\{x_k\}_{k\in\mathbb{N}}$ is generated by Algorithm 1 and $\{f(x_k)\}_{k\in\mathbb{N}}$ is bounded below. Assume, moreover, that $d_k \in D_k$ for all $k \in \mathbb{N}$ and (x_*, d) is a limit point of the subsequence $\{(x_k, d_k)\}_{k\in K}$. Then*

$$\langle g(x_*), d\rangle \geq 0. \tag{7}$$

Proof. Let K_1 be an infinite subset of K such that $\lim_{k\in K_1} x_k = x_*$ and $\lim_{k\in K_1} d_k = d$.

By Proposition 1, we have that

$$\lim_{k\in K_1} \alpha_k^2 \beta_k = 0.$$

If some subsequence of $\{\beta_k\}$ converges to zero, then $g(x_*) = 0$ and we are done. Otherwise, we have that $\lim_{k\in K_1} \alpha_k = 0$. Let us analyze this situation. Therefore, for $k \in K_1$ large enough, we have that $\alpha_k < 1$.

Without loss of generality let us assume that $\alpha_k < 1$ for all $k \in K_1$. By the initial choice of $\alpha(d)$ and (3) we have that for all $k \in K_1$ and for all $d \in D_k$, there exists $\alpha'_k(d)$ such that

$$\lim_{k \in K_1} \alpha'_k(d) = 0 \tag{8}$$

and

$$f(x_k + \alpha'_k(d)d) > \bar{f}_k + \eta_k - (\alpha'_k(d))^2 \beta_k. \tag{9}$$

In particular, (9) holds for $d = d_k$. Let us write, for simplicity $\alpha'_k = \alpha'_k(d_k)$. Therefore, since $\bar{f}_k \geq f(x_k)$,

$$\frac{f(x_k + \alpha'_k d_k) - f(x_k)}{\alpha'_k} \geq -\alpha'_k \beta_k$$

for all $k \in K_1$. By the Mean Value Theorem, for all $k \in K_1$ there exists $\xi_k \in [0, 1]$ such that

$$\langle g(x_k + \xi_k \alpha'_k d_k), d_k \rangle \geq -\alpha'_k \beta_k.$$

Therefore, for all $k \in K_1$,

$$\langle g(x_k + \xi_k \alpha'_k d_k) - g(x_k), d_k \rangle + \langle g(x_k), d_k \rangle \geq -\alpha'_k \beta_k.$$

So, for all $k \in K_1$,

$$\langle g(x_k), d_k \rangle \geq -\alpha'_k \beta_k - \|g(x_k + \xi_k \alpha'_k d_k) - g(x_k)\| \|d_k\|.$$

Define $\beta'_k = \alpha'_k \beta_k + \|g(x_k + \xi_k \alpha'_k d_k) - g(x_k)\| \|d_k\| > 0$. Since $\|d_k\|$ and β_k are bounded and $\alpha'_k \to 0$ we have that

$$\lim_{k \in K_1} \beta'_k = 0 \tag{10}$$

and

$$\langle g(x_k), d_k \rangle \geq -\beta'_k$$

By (10), taking limits in both sides of this inequality, we obtain the desired result. □

Remark 1. Observe that the algorithm does not have a stopping criterion. The iterations continue even when $g(x_k) = 0$. However the proof is correct even when this occurs, since the sequence is always infinite.

Remark 2. The role of the parameter η_k is to guarantee that the iteration is well defined, even when d_k is not a descent direction.

Remark 3. The condition (2) is satisfied if $\beta_k = 1$ for all k and, many times, this is the only reasonable choice. In some situations, however, different alternatives are more reasonable. For example, if $f(x) \geq 0$ for all x and a solution with null (or almost null) objective function value can be expected (i.e. least-squares problems) it is sensible to choose $\beta_k = \min\{c_1, c_2 f(x_k)\}$ where $c_1, c_2 > 0$ are suitable scaling parameters.

Corollary 1. *Assume that x_k and D_k are as in Theorem 1, $0 < \theta < 1$, and $0 < \Delta_{min} < \Delta_{max} < \infty$. Suppose that the level set $\{x \in \mathbb{R}^n \mid f(x) \leq f(x_0) + \eta\}$ is bounded and that K_1 is an infinite subset of K such that for all $k \in K_1$ there exists $d_k \in D_k$ satisfying*

$$\|d_k\| \in [\Delta_{min}, \Delta_{max}] \quad \textit{and} \quad \langle d_k, g(x_k) \rangle \leq -\theta \|g(x_k)\| \|d_k\|. \tag{11}$$

Then, for all $\varepsilon > 0$, there exists $k \in \mathbb{N}$ such that $\|g(x_k)\| \leq \varepsilon$.

Proof. Since, by the definition of the algorithm, $f(x_k) \leq f(x_0) + \eta$ for all $k \in \mathbb{N}$, the sequence $\{x_k\}$ is bounded. Then, by (11), there exists an infinite subsequence $K_2 \subset K_1$ such that

$$\lim_{k \in K_2} x_k = x_*, \quad \lim_{k \in K_2} d_k = d,$$

for some $x_* \in \mathbb{R}^n$ and $d \neq 0$. By (11), $\langle g(x_k), d_k \rangle \leq 0$ for all k. Then, by Theorem 1, $\langle g(x_*), d \rangle = 0$.

Therefore, $\lim_{k \in K_2} \langle d_k, g(x_k) \rangle = 0$.

Then, by (11), $\lim_{k \in K_2} \|g(x_k)\| \|d_k\| = 0$. Since $\|d_k\| \geq \Delta_{min} > 0$ for all k, this implies that $\lim_{k \in K_2} \|g(x_k)\| = 0$. So, for $k \in K_2$ large enough, $\|g(x_k)\| \leq \varepsilon$, as we wanted to prove. □

Corollary 1 says that, under the assumption (11), stationary points up to any arbitrary precision can be found by Algorithm 1. Now, strictly speaking, the fulfillment of the second part of (11) depends on knowing $g(x_k)$, which is beyond our possibilities if we want to devise truly derivative-free methods. We may circumvent this difficulty by means of the occasional choice of a random direction. Roughly speaking, the condition that must be satisfied by a random direction d_k is that the probability of (11) must be greater than a fixed probability $p > 0$. This requirement is easy to satisfy due to the geometrical meaning of (11). With some abuse of language, the convergence properties of this "occasionally random" version of Algorithm 1 are given in Theorem 2.

Theorem 2. *Assume that $\{x_k\}_{k \in \mathbb{N}}$ is generated by Algorithm 1 with the condition that, for all $k \in \mathbb{N}$, a direction $d_k \in D_k$ is chosen randomly in such a way that:*

1. *$d_0, d_1, d_2, \ldots$ are independent n-dimensional random variables;*
2. *There exist $\theta \in (0,1), p \in (0,1)$, $0 < \Delta_{min} < \Delta_{max} < \infty$ such that, for all $k \in \mathbb{N}$, the probability of the event defined by (11) is greater than p.*

Assume that $\{x \in \mathbb{R}^n \mid f(x) \leq f(x_0) + \eta\}$ is bounded and $\varepsilon > 0$. Then, with probability 1, there exists $k \in \mathbb{N}$ such that $\|g(x_k)\| \leq \varepsilon$.

Proof. By the definition of Algorithm 2, for all $\ell \in \mathbb{N}$ the probability of the event defined by

$$\langle g(x_{(\ell-1)M}), d_{(\ell-1)M} \rangle \leq -\theta \|g(x_{(\ell-1)M})\| \|d_{(\ell-1)M}\|, \ldots, \langle g(x_{\ell M-1}), d_{\ell M-1} \rangle \leq -\theta \|g(x_{\ell M-1})\| \|d_{\ell M-1}\| \quad (12)$$

and

$$\Delta_{min} \leq \|d_{(\ell-1)M}\| \leq \Delta_{max}, \ldots, \Delta_{min} \leq \|d_{\ell M-1}\| \leq \Delta_{max} \quad (13)$$

is greater than $p^M > 0$. Therefore, the probability of the existence of a sequence $K_1 \subset \mathbb{N}$ such that (12-13) holds for all $k \in K_1$ is equal to 1.

Now, in each set of indices of the form $\{(\ell-1)M, \ldots, \ell M - 1\}$ necessarily one of them is of the form $\nu(\ell) - 1$. Therefore, the probability of the existence of a subsequence $K_1 \subset K$ such that (11) holds for all $k \in K_1$ is equal to 1. Therefore, by Corollary 1, the probability of the existence of k such that $\|g(x_k)\| \leq \varepsilon$ is equal to 1, as we wanted to prove. □

The choice (4) allows one to employ extrapolation steps. Roughly speaking, after finding an acceptable point $x_k + \alpha_k d_k$ one tries to find an even better point $x_k + c\alpha_k d$ for some $c > 1$. This may be quite useful far from the solution. Assume that $c_{max} > 1$. A simple Extrapolation algorithm is given below. However, there is a large field for extrapolation improvement using the theory of sequence transformations [6, 7].

Algorithm 2. (Extrapolation)

Step 1. Set $c = 1$.
Step 2. If $2c > c_{max}$ set $x_{k+1} = x_k + c\alpha_k d$ and finish the iteration.
Step 3. If $f(x_k + 2c\alpha_k d) > f(x_k + c\alpha_k d)$, set $x_{k+1} = x_k + c\alpha_k d$ and finish the iteration.
Step 4. Set $c \leftarrow 2c$ and go to Step 2.

2.1 Discussion

One should be very cautious in the interpretation of Theorem 2. Algorithms based on random choices for minimization usually converge to global minimizers with probability 1. On the other hand, Theorem 2 guarantees

a weaker property. So, why should we use this random choice of directions in a practical algorithm instead of any standard global optimization procedure based on random points?

Moreover, theorems that say that random algorithms converge to global minimizers with probability 1 usually give very little information about the practical behavior of the method. Isn't this the case of our Theorem 2? In other words, assume that we define an algorithm based on a reasonable (say, quasi-Newton) choice of the directions with the contribution of occasional random directions, satisfying the assumptions of Theorem 2: Should this be more efficient than merely using the "reasonable choices" with no random direction at all?

There is still a third question: with the assumptions of Theorem 2, is it possible to prove convergence to global minimizers?

The third question is merely theoretical and its answer is *No*. Let us give a one-variable counter-example. Assume that $f : I\!R \to I\!R$ is such that $\lim_{x\to-\infty} f(x) = \infty$, f is strictly decreasing in $(-\infty, 1]$, strictly increasing in $[1, 3]$, strictly decreasing in $[3, 5]$ and strictly increasing in $[5, \infty)$ with $\lim_{x\to\infty} f(x) = \infty$. Assume that $f(1) = 1$, $f(3) = 4$, $f(5) = 0$. Therefore, 1 is a local minimizer and 5 is a global minimizer. Assume that $\eta = 1$, $f(x_0) = 2$, $\Delta_{max} = 1$. Assume that the level set defined by $f(x) \leq 3$ has two connected components $[0.5, 1.5]$ and $[4, 6]$. Finally, assume that $D_k = \{d_k\}$ and that $\|d_k\|$ is always not greater than Δ_{max}. Then, all the iterates belong to $[0.5, 1.5]$ and, therefore, the probability of convergence to the global minimizer is zero.

Let us go now to the first question. We wish to compare a naive implementation of Algorithm 1 (which *do not* exhibits convergence to global minimizers) with a naive random-point algorithm which possess the property of finding global minimizers with an arbitrary precision. Both algorithms needs decisions about the distribution of the random variables that define the directions (in the case of Algorithm 1) and the random points (in the case of the competitor). In the case of Algorithm 1, let us use $D_k = \{d_k\}$, choosing d_k with all its random components uniformly distributed between -1 and 1. In the case of the "competitor" we need to define the distribution of the search points x_k. This is a hard decision, so, we are going to give this algorithm an additional advantage: we will generate the random points uniformly in a box where the global minimizer is known to be. Finally, in Algorithm 1 we will use $\eta_k = 1.1^{-k}$, $\beta_k = 1$ for all k, $\tau_{min} = \tau_{max} = 0.5$ and $M = 1$.

We wish to minimize $f(x) = \sum_{i=1}^{n} x_i^2/i$. In Algorithm 1 the initial point x_0 is chosen with all its components randomly distributed in $[-50, 50]$. In the case of the Competitor, the random trial points are always chosen uniformly in $[-50, 50]^n$. The results for $n = 10$ were the following: after more than 2 million functional evaluations and five minutes of execution time, the Competitor obtained a best function value of 48.38. The simple implementation on Algorithm 1, on the other hand, obtained a functional value smaller than 10^{-6} in 921 iterations with 16012 functional evaluations and using less than 1 second.

The second question remains. Is there any practical advantage in adding, from time to time, random directions to set D_k, if the other (one or more than one) directions in D_k are (say) approximate quasi-Newton or gradient type directions? The answer to this question is in our numerical experiments.

3 A random search algorithm

To illustrate the flexibility of our model algorithm (Algorithm 1), and the potentiality of the theory developed in Section 2, we present the following algorithm that uses randomly generated search directions at every iteration.

Assume that f, τ_{min}, τ_{max}, $\{\eta_k\}$, and x_0 are as in Algorithm 1 and that c_{max} is as in Algorithm 2. Assume that $\Delta_{min}, \Delta_{max}$ are such that $0 < \Delta_{min} < \Delta_{max} < \infty$.

Given $x_k \in I\!R^n$, the steps for computing x_{k+1} are the following:

Algorithm 3. (Random line-search algorithm)

Step 1. *Compute a random direction*

Compute a random direction $d \in I\!R^n$ such that $\Delta_{min} \leq \|d\| \leq \Delta_{max}$.

Define $\bar{f}_k = \max\{f(x_k), \ldots, f(x_{\max\{k-M+1,0\}}\}$.

Step 2. *Trying unitary step*

If

$$f(x_k + d) \leq f(x_k) + \eta_k - \beta_k, \tag{14}$$

set $\alpha_k = 1$, $d_k = d$ and go to Step 5 (Extrapolation).

If

$$f(x_k - d) \leq f(x_k) + \eta_k - \beta_k, \tag{15}$$

set $\alpha_k = 1$, $d_k = -d$ and go to Step 5 (Extrapolation).

Step 3. *Quadratic interpolation*

Compute $\tilde{\alpha}$, the minimizer of the parabola that interpolates the points

$$(-1, f(x_k - d)), (0, f(x_k)), (1, f(x_k + d)).$$

If $\tilde{\alpha}$ exists and belongs to $[\tau_{min}, \tau_{max}]$, set $d_k = d$ and go to Step 4 (Backtracking).
If $\tilde{\alpha}$ exists and belongs to $[-\tau_{max}, -\tau_{min}]$, set $d_k = -d$, $\tilde{\alpha} = -\tilde{\alpha}$ and go to Step 4 (Backtracking).
If $f(x_k + d) \leq f(x_k - d)$, set $d_k = d$, $\tilde{\alpha} = 1/2$ and go to Step 4 (Backtracking).
If $f(x_k + d) > f(x_k - d)$, set $d_k = -d$, $\tilde{\alpha} = 1/2$ and go to Step 4 (Backtracking).

Step 4. *Backtracking*

Step 4.1. Set

$$\alpha \leftarrow \tilde{\alpha}. \tag{16}$$

Step 4.2. If

$$f(x_k + \alpha d_k) \leq \bar{f}_k + \eta_k - \alpha^2 \beta_k, \tag{17}$$

set $\alpha_k = \alpha$, $x_{k+1} = x_k + \alpha_k d_k$ and finish the iteration.

If (17) does not hold, compute $\alpha_{new} \in [\tau_{min}\alpha, \tau_{max}\alpha]$ using safeguarded quadratic interpolation, set $\alpha \leftarrow \alpha_{new}$ and repeat the test (17).

Step 5. *Extrapolation*

Use Algorithm 2 to obtain $c \geq 1$ and set $x_{k+1} = x_k + c\alpha_k d_k$.

Clearly, this algorithm is a particular case of Algorithm 2, although the rigorous verification is rather tedious.

In order to assess the performance of this algorithm, the twenty first problems from Moré, Garbow and Hillstrom collection [23] were selected to constitute our test set. The tests were run in Fortran 77, double precision. The initial approximation x^0 was the default proposed in [23]. We also used $\tau_{min} = 0.1, \tau_{max} = 0.9, c_{max} = 10, \Delta_{min} = -2, \Delta_{max} = 2$ and $M = 15$.

The sequences in the line search, η_k and β_k, are defined as:

$$\eta_k = \frac{|f(x_0)|}{k^{1.1}} \text{ and } \beta_k \equiv 1, \ k = 0, 1, \ldots$$

The algorithm is interrupted if:

$$f(x_k) \leq 10^{-9}, \tag{18}$$

since all of these tests are least–squares problems. However, as the algorithm may find any critical point of problem (1), we also adopted the stopping criterion

$$\|x_{k+1} - x_k\| \leq tol. \tag{19}$$

We say that the sequence $\{x_k\}$ does not converge if a maximum number of function evaluations in the main algorithm (500000) or in the line search (1000) was exceeded, or a maximum number of iterations (5000) was attained. The results are shown in Table 1, where problems are presented according to the order of [23]. We used the following notation:

- n - denotes the number of variables;

- *Conv=1* indicates that the stopping criterion (18) was satisfied at an approximation x_k;
- *Conv=2* means that the algorithm was interrupted because (19), with $tol = 10^{-7}$, was occurred;
- *Conv=3* indicates that the maximum number of function evaluations in the line search was exceeded;
- *Conv=4* means that the maximum number of function evaluations in the main algorithm has been attained;
- *Conv=5* denotes that the maximum number of iterations was exceeded;
- *NC* means nonconvergence since a non numeric value (NaN) was attained;
- *It* and *InterIt* denotes, respectively, the number of iterations in the main algorithm and the number of line search iterations;
- *Searches* represent the number of times that the line search procedure was necessary;
- f is the value of the function at the solution obtained by the algorithm;
- *difx* is equal to the value of $\|x_{k+1} - x_k\|$.

Prob	n	Conv	It	InterIt	Searches	Evalf	f	difx
1	2	2	430	377	98	1619	4.91D-05	8.96D-08
2	2	5	5000	5470	1528	22243	4.91D+01	7.83DE-05
3	2	2	23	12	1	78	1.352D-01	2.55D-08
4	2	2	341	0	0	342	10.00D+11	4.09D-08
5	2	2	849	2634	531	5122	2.55D-03	6.11D-08
6	2	2	3769	38	22	15026	1.24D+02	7.68D-10
7	3	NC	134	0	0	137	NaN	NaN
8	3	2	1453	1627	544	5905	8.52D-03	4.03D-08
9	3	2	4	29	4	42	1.01D-06	4.97D-08
10	3	2	2	0	0	6	7.13D+08	4.26D-11
11	3	2	1010	2826	585	5832	6.67D+00	9.35D-08
12	3	2	682	98	43	1740	1.28D-04	2.47D-08
13	4	2	487	20	487	1174	3.52D-05	3.86D-08
14	4	5	5000	358	173	12636	1.78D+00	8.76D-02
15	4	2	22	126	19	193	2.92D-03	5.94D-08
16	4	5	5000	0	0	5001	6.35D+06	2.55D-06
17	4	3	356	620	216	1689	6.57D-02	6.67D-08
18	6	2	1556	1356	5516	10185	3.44D-02	9.51D-08
19	11	2	407	174	77	1386	7.65D+00	6.81D-09
20	6	2	1587	1234	517	5967	2.00D-02	1.36D-08

Table 1: Random search algorithm

We must recall that the directions generated by this algorithm are always chosen randomly. In spite of this, comparing the results in Table 1 with the exact solutions exhibited in [23], we can see that two of the twenty test problems were successfully solved (Problems 6 and 8) and for six of the problems $(2, 9, 16, 18, 19$ and $20)$ the objective function value was close to the one reported in [23]. On the negative size, the number of function evaluations is clearly large even for small dimensions. For medium size or large scale problems we will discuss in the next sections additional options, that nevertheless, will take advantage of the theoretical and practical features of random directions.

4 Discrete gradient type algorithms

In this section we present an algorithm that combines the idea of a random direction with a gradient type direction, according to our discussion of subsection 2.1.

Assume that f, τ_{min}, τ_{max}, $\{\eta_k\}$, and x_0 are as in Algorithm 1. Assume, as in Algorithm 2, that $c_{max} > 1$. Let $\{\varepsilon_k\}$ be a sequence such that $\varepsilon_k > 0$ for all $k \in I\!N$. Assume that $0 < \sigma_{min} < \sigma_0 < \sigma_{max} < \infty$. In this algorithm we define g_k as a discrete approximation of the gradient vector at x_k, as discussed and used in [14].

Given $x_k \in I\!R^n$, the steps for computing x_{k+1} are the following:

Algorithm 4. (Discrete gradient type algorithm)

Step 1. *Decide wether or not to use a random direction*
Compute a random real number $0 < z < 1$. If $z \leq p$ then compute a random direction $d_k \in I\!R^n$ such that

$$\Delta_{min} \leq \|d_k\| \leq \Delta_{max},$$

and go to Step 3. Else ($z > p$) go to Step 2.

Step 2. *Compute the discrete gradient at iteration 0*
If $k > 0$ go to Step 2.
Step 2.1. Set $y \leftarrow x_0$.
Step 2.2. For $j = 1, \ldots, n$, execute Steps 1.3–1.5.
Step 2.3. Set $h \leftarrow \varepsilon_k \; sign([y]_j)$.
Step 2.4. Set $z = y + h \; e_j$.
Step 2.5. Set $[g_0]_j = [f(z) - f(y)]/h$.
Step 2.6. If $f(z) < f(y)$, set $y \leftarrow z$.
Step 2.7. Re-define $x_0 \leftarrow y$.

Step 3. *Compute the search direction*
Compute $d_k = -g_k/\sigma_k$.

Step 4. *Backtracking*
Step 4.1. Set

$$\alpha \leftarrow 1. \tag{20}$$

Step 4.2. If (17) holds set $\alpha_k = \alpha$. If $\alpha_k = 1$, go to Step 5 (Extrapolation).
If (17) holds and $\alpha_k < 1$, set $y \leftarrow x_k + \alpha_k d_k$ and go to Step 6.
If (17) does not hold, compute $\alpha_{new} \in [\tau_{min}\alpha, \tau_{max}\alpha]$ using safeguarded quadratic interpolation, set $\alpha \leftarrow \alpha_{new}$ and repeat the test (17).

Step 5. *Extrapolation*
Use Algorithm 2 to obtain $c \geq 1$, set $y = x_k + c\alpha_k d_k$, and go to Step 6.

Step 6. *Compute the new discrete gradient*
Step 6.1. For $j = 1, \ldots, n$, execute Steps 5.2–5.5.
Step 6.2. Set $h \leftarrow \varepsilon_k$. If $[y]_j < [x_k]_j$, set $h \leftarrow -h$.
Step 6.3. Set $z = y + h \;\; e_j$.
Step 6.4. Set $[g_{k+1}]_j = [f(z) - f(y)]/h$.
Step 6.5. If $f(z) < f(y)$, set $y \leftarrow z$.

Step 7. *Compute the new iterate*
Set $x_{k+1} \leftarrow y$.

Step 8. *Compute the inverse of the next step length*
Choose $\sigma_{k+1} > 0$ using your favorite gradient type method.

Remark 5. At Steps 1 and 5 of Algorithm 4, the discrete gradient is computed. When, at an auxiliary point, it is detected that the functional value decreases, the auxiliary point is taken as central point of the

approximation and the new increment is computed starting from it. When one tries to exploit parallelism, this is not the best decision, being better to keep the same central point throughout the gradient estimation process and computing the auxiliary evaluations in parallel. The remark that follows holds for both versions of the algorithm.

Remark 6. Algorithm 4 can be viewed as a particular case of Algorithm 1. Hence, from Corollary 1, the following argument is obtained. If for infinitely many iterations condition (11) holds, then stationary points will be found up to any arbitrary precision. The assumption concerning the angle between d_k and the exact negative gradient at x_k seems to be quite reasonable when the directions d_k are built using a discrete approximation of the exact gradient vector with sufficiently small values of ε_k. In practice, for all k, $\varepsilon_k = 10^{-8}\|x_0\|_\infty$ (if $\|x_0\|_\infty = 0$, then we chose $\varepsilon_k = 10^{-8}$).

We present numerical results obtained with algorithm 4, where we choose for the step length, at Step 8,

$$\sigma_{k+1} = \max\{\sigma_{min}, \min\{\sigma_{max}, \frac{\langle g_{k+1} - g_k, x_{k+1} - x_k\rangle}{\|x_{k+1} - x_k\|^2}\}\},$$

i. e., we are using the *spectral gradient method* [3, 26].

Now our test set is composed by the fifteen (from 21 to 35) problems from Moré, Garbow and Hillstrom collection [23] that can be run for different values of n. They were also run in Fortran 77, double precision. Algorithmic parameter choices for these tests were mostly the same used for the Algorithm 3 implementation, except for $\sigma_{min} = 10^{-10}, \sigma_{max} = 10^{10}$, $tol = 10^{-6}$ and $kmax = 1500$. The results are shown in Tables 2–4, according to the different values of p. The notation of these tables is similar to the one of Table 1, except that the number of non descent directions generated during the process is given in column "AscDir" and we also provide the norm of the discrete gradient at the solution obtained by the algorithm in column "normg".

Prob	n	Conv	It	InterIt	Searches	Evalf	AscDir	f	difx	normg
21	100	2	60	65	8	6226	0	4.29E-07	7.11E-07	5.94E-04
22	100	2	468	5	2	47374	0	3.66E-06	9.98E-07	1.53E-04
23	100	3	1	1001	2	1204	0	1.14E+11	5.00E-01	2.93E+22
24	100	2	243	0	0	24644	0	9.71E+04	9.16E-07	1.82E-02
25	100	1	254	1374	227	27130	0	7.14E-10	2.22E-05	2.29E-01
26	100	2	175	7	3	17783	0	1.84E-06	8.86E-07	4.92E-06
27	100	2	4	0	0	505	0	4.02E-09	8.54E-08	6.51E-05
28	100	2	99	41	14	10141	0	1.08E-06	8.89E-07	2.68E-05
29	100	1	5	0	0	607	0	4.91E-11	1.39E-03	1.20E-04
30	100	1	24	0	0	2526	0	3.02E-10	1.04E-04	7.18E-04
31	100	1	16	0	0	1718	0	1.13E-10	3.13E-05	2.23E-03
32	100	1	2	0	0	304	0	2.56E-15	1.90E+01	1.76E-02
33	100	5	1500	5137	1499	156738	0	3.43E+09	2.66E-01	3.96E+10
34	100	5	1500	4854	1499	156455	0	2.75E+10	6.35E-01	1.07E+11
35	100	2	334	35	18	33870	0	9.48E-03	9.58E-07	1.39E-03

Table 2: Discrete spectral gradient algorithm - $p = 0$.

Prob	n	Conv	It	InterIt	Searches	Evalf	AscDir	f	difx	normg
21	100	2	167	92	21	17060	3	2.37E-07	5.40E-07	4.45E-04
22	100	5	1500	173	105	151774	28	4.687E-05	3.00E-04	5.19E-02
23	100	NC	1	0	0	202	0	NaN	NaN	NaN
24	100	2	1451	150	35	146802	33	9.71E+04	8.80E-07	2.49E-02
25	100	1	21	6	3	1129	0	8.55E-12	1.78E-04	9.46E-01
26	100	2	132	25	5	13458	1	2.24E-06	8.87E-07	9.07E-06
27	100	2	4	0	0	505	0	4.02E-09	8.54E-08	6.51E-05
28	100	2	349	282	65	35632	2	1.07E-06	9.28E-07	2.53E-05
29	100	1	5	0	0	607	0	4.91E-11	1.39E-03	1.20E-04
30	100	1	24	0	0	2526	0	3.02E-10	1.04E-04	7.18E-04
31	100	1	16	0	0	1718	0	1.13E-10	3.13E-05	2.23E-03
32	100	1	2	0	0	304	0	2.56E-15	1.90E+01	1.76E-02
33	100	2	10	28	8	1139	0	1.16E+12	6.93E-08	7.30E+11
34	100	2	10	26	8	1137	1	4.26E+12	6.86E-08	1.34E+12
35	100	2	799	359	104	81159	20	9.48E-03	8.29E-07	1.17E-03

Table 3: Discrete spectral gradient algorithm - $p = 0.05$.

Prob	n	Conv	It	InterIt	Searches	Evalf	AscDir	f	difx	normg
21	100	2	129	95	18	13225	7	7.41E-07	9.33E-07	7.78E-04
22	100	5	1500	424	213	152025	80	2.28E-03	1.35E-04	2.72E-02
23	100	NC	1	0	0	202	0	NaN	NaN	NaN
24	100	5	1500	422	90	152023	74	9.71E+04	2.14E-02	5.00E+00
25	50	2	37	6	3	1944	1	1.15E-05	1.47E-07	6.77E-03
26	100	2	388	191	34	39480	13	1.37E-06	9.76E-07	4.66E-06
27	100	2	4	0	0	505	0	4.02E-09	8.54E-08	6.51E-05
28	100	2	187	101	31	19089	0	1.08E-06	9.79E-07	2.83E-05
29	100	1	5	0	0	607	0	4.91E-11	1.39E-03	1.20E-04
30	100	1	131	28	10	13361	7	6.41E-10	8.22E-06	3.05E-04
31	100	1	40	1	1	4143	2	8.43E-11	3.62E-05	1.41E-03
32	100	1	2	0	0	304	0	2.56E-15	1.90E+01	1.76E-02
33	100	2	8	23	6	932	2	3.54E+12	7.07E-08	1.27E+12
34	100	2	2	0	0	303	1	6.10E+12	7.14E-08	1.60E+12
35	100	2	1140	757	203	115998	48	9.48E-03	8.05E-07	1.17E-03

Table 4: Discrete spectral gradient algorithm - $p = 0.1$.

To analyze the performance of this discrete gradient type algorithm, for different values of p, we summarized Tables 2–4, obtaining the figures of Table 5, where the second column shows the number of problems that obtained "Conv=1", and the last column contains the tests with final value of f close to f_* (up to 10^{-6}).

p	Conv=1	$\approx f_*$
0	5	2
0.05	5	2
0.1	4	2

Table 5: Summary of Tables 2–4

First we observe that using a suitable choice of step length, Algorithm 4 shows in general an effective performance for different values of p. We also observe that, for some tests, the final value of the objective function was less than 10^{-4}. This occurred for three problems with each value of p. Notice that we set $n = 100$ in problem 35, while in [23] results are reported for this problem with, at most, ten variables. As f_* must be $\approx 6.50 * 10^{-3}$, if $n = 10$, we considered that the results obtained by algorithm 4, respect to problem 35, is valuable. In summary, we can say that the discrete gradient type algorithm failed in four problems when $p = 0$ and $p = 0.05$ and in five problems with $p = 0.1$, representing, respectively, success in 73% and 67% of the tests.

In Table 6 we present results of Algorithm 4 to four large–scale problems, out of the same test set used above. For these tests, the best results were obtained with $M = 5$.

Prob	n	Conv	It	InterIt	Searches	Evalf	AscDir	f	difx	normg
21	5000	2	36	15	5	185052	0	2.95D-05	5.86D-06	4.93D-03
26	5000	2	48	74	16	245123	2	9.44D-06	7.07D-06	1.28D-01
27	5000	2	4	0	0	25005	0	7.98D-03	5.01D-06	8.90D+00
31	5000	2	18	0	0	95019	0	2.094D-09	9.80D-06	8.69D-04

Table 6: Discrete spectral gradient algorithm (large-scale problems) - $M = 5$.

All of these tests were interrupted with "Conv=2", but the values of f are close to f_*. We verify that good results can be obtained combining the proposed nonmonotone line search strategy with a discrete gradient type algorithm, specially if n is large.

5 Discrete inverse SR1 update

The symmetric rank one method (SR1) has been considered in the last few years as a serious quasi-Newton competitor with the BFGS and the DFP methods for unconstrained optimization. At the k-th iteration, a symmetric matrix B_k is given to approximate the Hessian of f, and a search direction is computed by

$$d_k = -B_k^{-1} g_k.$$

The SR1 update for the next iteration is given by

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k},$$

where the vector $y_k = g_{k+1} - g_k$, $s_k = x_{k+1} - x_k$, and g_k is the exact gradient at x_k. In this work, we propose to use the approximated gradient vectors g_k, without derivative information, as in the discrete gradient type method (Section 4). This update, using the exact gradient, was first suggested independently by Broyden [8], Davidon [12], and Fiacco and McCormick [15].

By the well-known Sherman-Morrison formula, we can also obtain the associated update for the inverse of the approximated Hessian H_k:

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k},$$

where, once again, we can use the approximated gradient vectors to build the matrices H_k.

An important characteristic of the SR1 update is that even if B_k is positive definite, then B_{k+1} may be indefinite. The same is true for H_k. Indeed, the denominator in both cases could be negative even when the function is a convex quadratic [13, 16] and, so, the eigenvalues might be shifted to the negative side. Moreover, the denominator could be zero or *numerically* zero, which could lead to numerical instability. However, in practice, the SR1 updates are surprisingly good (see e.g. [9]). To explain this behavior, some theoretical properties have been found [10, 16]. In particular, a very interesting property is the finite termination of the method, under mild assumptions, for convex quadratic functions. In this case, the sequence of SR1 matrices terminates at the exact Hessian (or the inverse) at iteration $n+1$ (see e.g. [16]). Moreover, for general functions, the matrices generated by the SR1 formulas tend to be very good approximations of the Hessian matrix (or the inverse), frequently better than the DFP and the BFGS matrices [10].

Concerning the drawback of the denominator being close to zero, a simple safeguard prevents the possible breakdown and the presence of numerical instabilities. In practice, it has been observed that SR1 methods perform well simply by skipping the update if the denominator is close to zero. To be precise, the update is applied only if

$$|(y_k - B_k s_k)^T s_k| \geq \rho \|s_k\| \, \|y_k - B_k s_k\|, \tag{21}$$

where $0 < \rho < 1$ (typically $\rho \approx 10^{-7}$). If (21) does not hold, we set $B_{k+1} = B_k$. A similar safeguard strategy is designed when dealing with the matrices H_k.

When a non positive definite matrix is built (either B_{k+1} or H_{k+1}), we might end up with an ascent direction. For that reason, the combination of SR1 updates with line search globalization strategies has been historically avoided. The presence of ascent directions, as discussed before, is totally acceptable by our line search scheme, and the global convergence is guaranteed. We now present an algorithm for the discrete gradient inverse SR1 update in which, based on our discussion of subsection 2.1, we use every once in a while a random direction. Similar versions can be easily obtained for the discrete gradient direct SR1 update, and also for the exact gradient SR1 updates.

Assume that f, τ_{min}, τ_{max}, $\{\eta_k\}$, and x_0 are as in Algorithm 1. Assume that $0 < \rho < 1$ and $0 < \sigma_{min} < \sigma_0 < \sigma_{max} < \infty$. Assume, as in Algorithm 2, that $c_{max} > 1$. Assume that Δ_{min} and Δ_{max} are as in Algorithm 3. Assume that H_0 is a given initial symmetric and positive definite matrix, that g_0 is the discrete gradient at x_0 obtained using Step 1 in Algorithm 4, and that $0 \leq p < 1$ is a real number. In this algorithm we define $\beta_k = \max\{\delta, \|g_k\|\}$, where $0 < \delta \ll 1$ is a fixed number and g_k is the discrete approximation of the gradient vector at x_k described in Algorithm 4.

Given $x_k \in I\!R^n$, the steps for computing x_{k+1} are the following:

Algorithm 5. (Discrete-gradient inverse SR1 update)

Step 1. *Decide wether or not to use a random direction*

Compute a random real number $0 < z < 1$. If $z \leq p$ then compute a random direction $d_k \in I\!R^n$ such that

$$\Delta_{min} \leq \|d_k\| \leq \Delta_{max},$$

and go to Step 3. Else ($z > p$) go to Step 2.

Step 2. *Compute the SR1 search direction.*
Compute $d_k = -H_k\, g_k$.

Step 3. *Backtracking*
Step 3.1. Set

$$\alpha \leftarrow 1. \tag{22}$$

Step 3.2. If (17) holds set $\alpha_k = \alpha$. If $\alpha_k = 1$, go to Step 4 (Extrapolation). If (17) holds and $\alpha_k < 1$, set $y \leftarrow x_k + \alpha_k d_k$ and go to Step 5.

If (17) does not hold, compute $\alpha_{new} \in [\tau_{min}\alpha, \tau_{max}\alpha]$ using safeguarded quadratic interpolation, set $\alpha \leftarrow \alpha_{new}$ and repeat the test (17).

Step 4. *Extrapolation*
Use Algorithm 2 to obtain $c \geq 1$, set $y = x_k + c\alpha_k d_k$, and go to Step 5.

Step 5. *Compute the new iterate*
Set $x_{k+1} \leftarrow y$.

Step 6. *Compute the vector y_k*
Step 6.1. Using Step 5 in Algorithm 4, compute the new discrete gradient g_{k+1}.
Step 6.2. Set $y_k = g_{k+1} - g_k$.

Step 7. *Compute the new matrix H_{k+1}*
Step 7.1. Set $s_k = x_{k+1} - x_k$.
Step 7.2. If $|(s_k - H_k y_k)^T y_k| \leq \rho \|y_k\|\, \|s_k - H_k y_k\|$ then set $H_{k+1} = H_k$
Else

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}$$

Remark 7. Algorithm 5 can be viewed as a particular case of Algorithm 1. Hence, from Theorem 2, with probability 1, stationary points will be found up to any arbitrary precision.

We now present numerical results obtained with algorithm 5, using the same test set used in the previous section, with $n = 100$, which is a medium-size dimension suitable for secant type methods. The experiments were also run in Fortran 77, double precision and the required parameters were the same ones used for Algorithm 4. The results are shown in Tables 7–9, according to the different values of p. The notation of these tables is identical to the one used for Tables 2–4.

Prob	Conv	It	InterIt	Searches	Evalf	AscDir	f	difx	normg
21	2	46	1029	12	5777	7	1.37D+03	1.67D+00	1.88D+03
22	2	94	5	3	9600	11	2.74D-07	5.71D-06	2.02D-04
23	3	0	1000	1	2002	0	1.11D+17	1.00D+00	1.58D+29
24	2	256	8	3	25965	44	9.71D+04	2.73D-06	2.64D-01
25	1	31	3	1	3235	0	2.72D-09	1.08D-06	2.30D-02
26	2	113	8	3	11522	5	2.42D-06	1.75D-07	8.80D-05
27	2	3	5	1	409	0	3.86D-07	1.49D-06	1.43D-03
28	2	12	4	3	11417	0	1.23D-06	2.24D-07	1.38D-04
29	1	6	0	0	707	0	2.87D-13	4.85D-06	1.12D-06
30	2	248	258	72	25407	50	1.12D+01	2.74D-06	7.09D-04
31	2	230	238	80	23569	77	1.18D+01	9.15D-06	3.51D-03
32	1	3	1	1	405	0	2.50D-15	9.34D-08	2.00D-07
33	2	3	3	1	407	0	2.46D+01	8.19D-08	2.99D+02
34	2	3	3	1	407	0	2.61D+01	9.92D-07	7.37D+03
35	2	280	26	10	28407	53	9.50D-03	2.50D-07	7.62D-03

Table 7: Discrete inverse SR1 algorithm for $n = 100$, $M = 15$, and $p = 0$.

Prob	Conv	It	InterIt	Searches	Evalf	AscDir	f	difx	normg
21	2	980	422	120	101043	220	0.4D-07	1.9D-07	3.4D-04
22	2	152	19	9	15712	19	2.2D-07	4.81D-07	1.8D-04
23	3	0	1000	1	1102	0	1.11D+12	1.00D+00	1.58D+21
24	2	249	34	15	24663	38	9.1D+04	4.1D-07	7.1D-01
25	2	45	3	1	3235	0	1.2D-06	5.08D-09	2.30D-03
26	2	158	121	13	16122	12	1.3D-06	8.1D-07	1.40D-05
27	1	11	5	1	1217	0	6.0D-09	1.2D-08	9.7D-05
28	2	41	16	1	4258	0	8.1D-07	8.2D-07	2.2D-04
29	1	7	0	0	808	0	0.5D-15	5.3D-07	1.7D-07
30	2	276	165	61	17407	51	5.1D+00	8.9D-07	4.2D-05
31	2	262	270	84	26569	47	1.2D+01	7.9D-07	3.51D-04
32	1	3	1	1	405	0	2.50D-15	9.34D-08	2.00D-07
33	2	3	3	1	407	0	2.46D+01	8.19D-08	2.99D+02
34	2	3	3	1	407	0	2.61D+01	9.92D-07	7.37D+03
35	2	148	126	17	15175	30	8.1D-03	3.1D-07	2.5D-02

Table 8: Discrete inverse SR1 algorithm for $n = 100$, $M = 15$, and $p = 0.05$.

Prob	Conv	It	InterIt	Searches	Evalf	AscDir	f	difx	normg
21	2	1273	968	246	129642	291	0.4D-06	1.5D-07	1.1D-03
22	2	409	118	46	41528	48	4.0D-07	2.81D-07	1.4D-04
23	3	0	1000	1	1102	0	1.11D+12	1.00D+00	1.58D+21
24	2	215	44	17	21860	29	1.1D+05	2.9D-07	9.0D-01
25	2	288	3	1	29192	62	3.1D+01	9.7D-07	11.4D+00
26	2	181	183	22	18565	16	1.8D-06	5.6D-07	1.10D-04
27	1	9	5	1	1015	0	4.0D-09	1.9D-07	6.8D-05
28	2	9	15	1	1025	1	1.1D-06	2.0D-07	3.1D-04
29	1	9	3	1	1013	1	0.1D-15	4.8D-08	4.3D-08
30	2	319	351	110	32671	80	4.1D+00	4.1D-07	1.3D-04
31	2	396	351	97	40448	72	5.2D+02	8.1D-07	3.1D-04
32	1	3	1	1	405	0	4.50D-17	4.8D-08	6.2D-08
33	2	4	3	1	508	0	3.1D+00	4.3D-08	5.2D-01
34	2	3	3	1	407	0	4.1D+00	3.7D-07	7.0D-01
35	2	284	296	37	29081	40	1.1D-02	1.3D-07	4.5D-03

Table 9: Discrete inverse SR1 algorithm for $n = 100$, $M = 15$, and $p = 0.1$.

We observe, as expected from a secant type method, that algorithm 5 requires, quite frequently, fewer iterations, and hence fewer function evaluations, than algorithm 4 for the same problems. In many cases, the algorithm solves the problem even though ascent directions are being generated, and this is clearly an advantage of the new line search strategy. In general, when $p = 0$ the method requires less iterations although there are cases for which convergence was not observed. On the other hand, when p increases, the number of iterations increases but also the number of successful results also increases. Based on this results, the value of $p = 0.05$ seems to be a compromise between efficiency and theoretical robustness. This remark also applies to our gradient type scheme (algorithm 4). Finally, we would like to mention that our discrete and globalized SR1 update is an interesting candidate to be extended for constrained problems. In that setting, it may not be possible to impose the curvature condition $y_k^t s_k > 0$, and thus the best known updates (e.g. BFGS) are

not recommended. For constrained problems, a line search SR1 update method could be a more flexible and suitable option, and so it deserves further investigation.

As a final remark, we conjecture that the new line search should be useful when coupled with novel derivative-free optimization algorithms [1, 4, 11], in particular those based on interpolatory quadratic models [24, 25].

References

[1] P. Alberto, F. Nogueira, H. Rocha and L. N. Vicente [2004], Pattern search methods for user-provided points: Application to molecular geometry problems, *SIAM J. Opt.* 14, pp. 1216-1236.

[2] G. Allaire [2001]. *Shape Optimization by the Homogenization Method*. Springer Verlag, New York.

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

[4] F. Van den Berghen and H. Bersini [2005], CONDOR, a new parallel, constrained extension of Powell's UOBYQA algorithm: Experimental results and comparison with the DFO algorithm, *Journal of Computational and Applied Mathematics* 181, pp. 157-175.

[5] E. G. Birgin, J. M. Martínez and M. Raydan [2003], Inexact spectral gradient method for convex-constrained optimization, *IMA Journal on Numerical Analysis* 23, pp. 539-559.

[6] C. Brezinski and A. C. Matos [1996], A derivation of extrapolation algorithms based on error estimates, *Journal of Computational and Applied Mathematics* 66, pp. 5-26.

[7] C. Brezinski and M. Redivo Zaglia [1991], *Extrapolation Methods. Theory and Practice*, North-Holland, Amsterdam.

[8] C. G. Broyden [1967], Quasi-Newton methods and their application to function minimization, *Mathematics of Computation* 21, pp. 368-381.

[9] R. H. Byrd, H. F. Khalfan and R. B. Schnabel [1996], Analysis of a rank-one trust region method, *SIAM J. Opt.* 6, pp. 1025-1039.

[10] A. R. Conn, N. I. M. Gould and P. L. Toint [1991], Convergence of quasi-Newton matrices generated by the symmetric rank one update, *Math. Prog.* 50, pp. 177-195.

[11] A. R. Conn, K. Scheinberg and Ph.L. Toint [1997], Recent progress in unconstrained nonlinear optimization without derivatives, *Math. Prog.* 79, pp. 397-414.

[12] W. C. Davidon [1968], Variance algorithms for minimization, *Computer J.* 10, pp. 406-410.

[13] J. E. Dennis Jr. and R. B. Schnabel [1983]. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ.

[14] M. A. Diniz-Ehrhardt, M. A. Gomes-Ruggiero, V. L. R. Lopes and J. M. Martínez [2003], Discrete Newton's method with local variations and global convergence for nonlinear equations, *Optimization* 52, pp. 417-440.

[15] A. V. Fiacco and G. P. McCormick [1968]. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley and Sons, NY.

[16] R. Fletcher [1987]. *Practical Methods of Optimization*. John Wiley & Sons, New York.

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

[18] J. Haslinger and R. A. E. Mäckinen [2003]. *Introduction to Shape Optimization: Theory, Approximation, and Computation*. SIAM.

[19] W. La Cruz, J. M. Martínez and M. Raydan [2006], Spectral residual method without gradient information for solving large-scale nonlinear systems of equations, *Mathematics of Computation* 75, pp. 1449-1466.

[20] D.H. Li and M. Fukushima [2000], A derivative-free line search and global convergence of Broyden-like method for nonlinear equations. *Optimization Methods and Software.*, 13, pp. 181-201.

[21] S. Lucidi and M. Sciandrone [2002], On the global convergence of derivative–free methods for unconstrained optimization, *SIAM J. Opt.* 13, pp. 97-116.

[22] B. Mohammadi and O. Pironneau [2001]. *Applied Shape Optimization for Fluids*. Clarendon Press, Oxford.

[23] J. J. Moré, B. S. Garbow and K. E. Hillstrom [1981], Testing unconstrained optimization software, *ACM Transactions on Mathematical Software*, 7, pp 17-41.

[24] M. J. D. Powell [2003], On trust region methods for unconstrained minimization without derivatives, *Math. Prog.*, 97, pp 605-623.

[25] M. J. D. Powell [2004], On the use of quadratic models in unconstrained minimization without derivatives, *Optimization Methods and Software*, 19, pp 399-411.

[26] M. Raydan [1997], The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM J. Opt.*, 7, pp. 26-33.