A pattern search bound constrained optimization method with a nonmonotone line search strategy

Johanna A. Frau and Elvio A. Pilotta *

June 25, 2018

Abstract

A new pattern search method for bound constrained optimization is introduced. The proposed algorithm employs the coordinate directions, in a suitable way, with a nonmonotone line search for accepting the new iterate, without using derivatives of the objective function. The main global convergence results are strongly based on the relationship between the step length and a stationarity measure. Several numerical experiments are performed using a well known set of test problems. Other line search strategies were tested and compared with the new algorithm.

Key words. pattern search methods, bound constrained optimization, global convergence, numerical experiments.

AMS Subject classifications. 90C30, 90C56, 65K05.

1 Introduction

In this paper, we propose a new algorithm to solve bound constrained optimization problems where the derivatives of the objective function are not available. So, the problem of interest is

\[
\text{minimize } f(x) \quad \text{subject to } x \in \Omega
\]

where \( f: \mathbb{R}^n \to \mathbb{R} \) and \( \Omega = \{ x \in \mathbb{R}^n | l \leq x \leq u \} \) with \(-\infty \leq l < u \leq \infty \). We assume that the objective function is continuously differentiable on \( \Omega \) but the derivative information is unreliable or non-existent. This situation frequently appears in many real world applications where functional values \( f(x) \) require complex simulations or the function contains noise. This kind of situations may arise in applications from molecular geometry [1, 24], medical image registration [27], shape and design optimization [6, 12, 23]. There are many problems where the functional values come from practical experiments, so the explicit formulation of the objective function is not available and Quasi-Newton or finite difference methods are not applicable. Derivative-free optimization has received considerable attention from the optimization community during the last years, including the establishment of solid mathematical foundations for many of the methods considered in practice. Particularly, pattern search methods have succeeded where more elaborate approaches fail. These methods belong to the family of direct search methods, characterized by unsophisticated implementations, the absence of the construction of a model of the objective function and the use of pattern matrices to explore directions around the current iterate. See [4, 5, 15, 16, 28].

Pattern search methods were initially introduced by Hooke and Jeeves [15] for unconstrained optimization problems and lately analyzed and formally presented by Torczon et. al. [16, 28]. More recently, some strategies were adapted from derivative-based methods and incorporated to pattern search methods. For instance, in [10] the authors introduced a global strategy, based in the ideas developed in [13, 18, 21, 22], that uses a nonmonotone line search scheme in a pattern search algorithm.

In [20] Lewis and Torczon extended the pattern search method for the bound constrained case although they did not perform numerical experiments. This problem was also studied in [3] using polynomial interpolation and trust region strategies, which is a quite different approach to pattern search methods.

*Facultad de Matemática, Astronomía y Física, Universidad Nacional de Córdoba, CIEM (CONICET), Medina Allende s/n, Ciudad Universitaria (5000) Córdoba, Argentina. E-mail: {jfrau, pilotta}@famaf.unc.edu.ar
In this article, we propose a pattern search method that includes a nonmonotone line search as globalization strategy for a bound constrained optimization problem \cite{1}. The new algorithm is based on the ideas introduced in \cite{10} for the unconstrained case, however the proofs of the main convergence results of our method use a completely different philosophy. To prove the global convergence, we use the stationarity measure $\chi(x)$ defined in \cite{10}. This measure takes into account the degree to which the directions of the steepest descent point outward with respect to the portion of the feasible region near $x$. In order to validate our algorithm, we perform several numerical experiments and comparisons with other well established algorithm. After that, we extend our benchmark study by incorporating other line search strategies \cite{2, 7, 17, 26, 29}.

This paper is organized as follows: some definitions and preliminary results are given in Section 2. The new algorithm is introduced in Section 3. Convergence results are stated in Section 4. Numerical experiments are presented and analyzed in Section 5. Finally, some conclusions are given in Section 6.

Notation

In this work, $e^{(i)}$ denotes the $i$-th canonical vector in $\mathbb{R}^n$ and $\| \cdot \|$ will be the Euclidean norm. Also, $\text{int}(\Omega)$ denotes the largest open set contained in $\Omega$.

2 Definitions and preliminary results

In this section, we present some definitions and results which are necessary in order to guarantee convergence of the method that we propose to solve problem \cite{1}.

The following two definitions are the basis of the theory of convergence and they are widely used in the context of optimization. The first definition refers to the cone $K$ generated by the set of all nonnegative linear combinations of vectors of a given set. The second definition includes those vectors that make an angle of $90^\circ$ or more with each element of $K$, the polar cone $K^\circ$.

**Definition 1** Let $D = \{v^{(1)}, v^{(2)}, \ldots, v^{(r)}\}$ be a set of $r$ vectors in $\mathbb{R}^n$. The set $D$ generates the cone $K$ if

$$K = \{u | u = \sum_{i=1}^{r} e^{(i)}v^{(i)}, e^{(i)} \geq 0, \text{ for } i = 1, 2, \ldots, r\}.$$

**Definition 2** The polar cone of a cone $K$, denoted by $K^\circ$, is the cone defined by

$$K^\circ = \{w | w^T u \leq 0, \text{ for all } u \in K\}.$$

When we are minimizing a function in a feasible region, we are particularly interested in choosing search directions (descent directions at best) which improve the objective function and remain feasible at the same time. Given $x \in \Omega$, we define $K(x, \epsilon)$ as the cone generated by 0 and the outward pointing normals of the constraints within a distance $\epsilon$ of $x$, namely

$$\{e^{(i)}u^{(i)} - x^{(i)} \leq \epsilon \} \cup \{-e^{(i)}x^{(i)} - l^{(i)} \leq \epsilon \}.$$

In other words, $K(x, \epsilon)$ is generated by the normals to the faces of the feasible region within distance $\epsilon$ of $x$. Observe that if $K(x, \epsilon) = \{0\}$, as in the unconstrained case or when $x$ is well within the interior of the feasible region, $K^\circ(x, \epsilon) = \mathbb{R}^n$. The cone $K(x, \epsilon)$ is important because, for suitable choices of $\epsilon$, its polar $K^\circ(x, \epsilon)$ approximates the feasible region near $x$. So, if $K^\circ(x, \epsilon) \neq \{0\}$, the search can proceed from $x$ along all directions in $K^\circ(x, \epsilon)$ for at least a distance of $\epsilon$ and still remain inside the feasible region. See \cite{10} for more details.

As in the theory of methods based explicitly on derivatives, in derivative free optimization, we need a measure that lets us know how close a point $x$ is to constrained stationarity. In this article, we adopted the following measure of stationarity

$$\chi(x) = \max_{x + \omega \in \Omega, \|\omega\| \leq 1} -\nabla f(x)^T \omega.$$

Roughly speaking, $\chi(x)$ captures the degree to which the direction of steepest descent is outward pointing with respect to the portion of the feasible region near $x$. In \cite{3}, the authors proved that if $\Omega$ is convex, the function $\chi$ has the following properties:
1. \( \chi(x) \) is continuous.

2. \( \chi(x) \geq 0 \).

3. \( \chi(x) = 0 \) if and only if \( x \) is a KKT point for the problem \([1]\).

Thus, showing that \( \chi(x_k) \to 0 \) as \( k \to \infty \) establishes a global first-order convergence result, which will be one of our primary goals on this work.

3 The bound constrained nonmonotone pattern search algorithm \( \text{nmps} \)

We begin this section by introducing the proposed algorithm.

Let \( M \) be a positive integer, which indicates how many previous functional values will be considered on the nonmonotone line search. Let \( \Delta_{\text{tol}} > 0 \) be the tolerance for the convergence criterion. Let \( D_{\oplus} \) be a finite set of \( \mathbb{R}^n \) given by the coordinate directions, that is,

\[
D_{\oplus} = \{ \pm e(i) | i = 1,2,\ldots,n \}.
\]

Assume that \( \{\eta_k\} \) is a sequence chosen such that \( \eta_k > 0 \), for all \( k = 0,1,2,\ldots \), and \( \sum_{k=0}^{\infty} \eta_k = \eta < \infty \) is a convergent series.

Suppose that \( x_0 \in \Omega \) is an initial approximation to the solution and let \( \Delta_0 > 0 \) be the initial value for the step length. Given \( x_k \in \Omega \), \( \Delta_k > \Delta_{\text{tol}} > 0 \), the steps for computing \( x_{k+1} \) are given by the following algorithm.

**Algorithm 1 (nmps)**

**Step 1:**
Compute \( f(x_k) \) and define \( f_{\text{max}}(x_k) \) such that

\[
f_{\text{max}}(x_k) = \max\{f(x_k), \ldots, f(x_{k-\min\{k,M-1\}})\} = \max_{0 \leq j \leq m(k)} \{f(x_{k-j})\}
\]

where \( m(k) = \min\{k, M - 1\} \).

**Step 2:** Backtracking

2.1 Find (if possible) \( d \in D_{\oplus} \) such that \( (x_k + \Delta_k d) \in \Omega \) and the inequality

\[
f(x_k + \Delta_k d) \leq f_{\text{max}}(x_k) + \eta_k - \Delta_k^2
\]

holds. Set \( \Delta_{k+1} \leftarrow 1 \) and \( x_{k+1} \leftarrow x_k + \Delta_k d \).

2.2 If there is no a direction \( d \in D_{\oplus} \) such that \( (x_k + \Delta_k d) \in \Omega \) and (2) holds, then set

\[
\Delta_k \leftarrow \frac{\Delta_k}{2}
\]

and repeat Step 2.1 until a new \( x_{k+1} \) is found.

If \( \Delta_{k+1} < \Delta_{\text{tol}} \) terminate the execution of the algorithm.

4 Theoretical results and convergence analysis

In order to prove our main convergence result we need to demonstrate some auxiliary results. The following proposition concerns the nonmonotonicty of \( \{f(x_k)\} \). It is analogous to Lemma 2.3 in [5].

**Proposition 1** If \( l(k) \) is an integer such that

\[
kM - m(kM) \leq l(k) \leq kM,
\]
with \( m(kM) = \min\{kM, M - 1\} \), and

\[
f(x_{t(k)}) = \max_{0 \leq j \leq m(kM)} \{f(x_{kM-j})\} = \max\{f(x_{kM}), f(x_{kM-1}), \ldots, f(x_{kM-M+1})\}
\]

then

\[
f(x_{t(k+1)}) \leq f(x_{t(k)}) + \eta_{kM} + \ldots + \eta_{kM-M+1} - \Delta_{kM+1}^2
\]

(3)

Proof.

From now on we are going to suppose that the iteration \( k \) is such that \( k \geq M \), therefore \( m(kM) = \min\{kM, M - 1\} = M - 1 \).

Then, by an inductive argument on \( t = 1, 2, \ldots, M \) we will prove that

\[
f(x_{kM+t}) \leq f(x_{t(k)}) + \eta_{kM} + \ldots + \eta_{kM-M+1} - \Delta_{kM+1}^2
\]

(4)

holds for all iteration \( k = 1, 2, \ldots \)

In fact, from (2), we have

\[
f(x_{kM+1}) \leq \max \{f(x_{kM}), f(x_{kM-1}), \ldots, f(x_{kM-M+1})\} + \eta_{kM} - \Delta_{kM}^2
\]

for all \( k \in \mathbb{N} \). Therefore, the inequality (4) holds for \( t = 1 \).

Now, by inductive hypothesis, we suppose that

\[
f(x_{kM+t'}) \leq f(x_{t(k)}) + \eta_{kM} + \ldots + \eta_{kM+t'-1} - \Delta_{kM+t'}^2
\]

for all \( t' = 1, 2, \ldots, t \).

We will prove that (4) holds for \( t + 1 \). Indeed,

\[
f(x_{kM+t+1}) \leq \max \{f(x_{kM}), f(x_{kM-1}), \ldots, f(x_{kM-M+1})\} + \eta_{kM+t} - \Delta_{kM+t}^2
\]

Now by induction step,

\[
\max \{f(x_{kM+1}), \ldots, f(x_{kM+t})\} < f(x_{t(k)}) + \eta_{kM} + \ldots + \eta_{kM+t+1}
\]

Thus,

\[
f(x_{kM+t+1}) \leq f(x_{t(k)}) + \eta_{kM} + \ldots + \eta_{kM+t-1} + \eta_{kM+t} - \Delta_{kM+t}^2
\]

so the inequality (4) holds for \( t + 1 \).

Finally, since \( (k+1)M - M + 1 \leq l(k+1) \leq kM + M \), we have

\[
l(k+1) = kM + t \text{ for some } t \in \{1, 2, \ldots, M\}.
\]

The next proposition is an important tool to prove global convergence of the nmps algorithm where the inequality of Proposition[] is applied iteratively. This idea has also been introduced by Birgin et. al in [5] and we adapted it for our case.
**Proposition 2** If \( \{f(x_k)\}_{k \in \mathbb{N}} \) is bounded below then \( \lim_{k \to \infty} \Delta^2_{\ell(k)-1} = 0 \)

**Proof.** By applying the inequality (3) we have

\[
f(x_{i(k+1)}) \leq f(x_0) + \sum_{k=0}^{\infty} \eta_k - \sum_{k=1}^{\infty} \Delta^2_{\ell(k)-1},
\]
equivalently

\[
\sum_{k=1}^{\infty} \Delta^2_{\ell(k)-1} \leq f(x_0) - f(x_{i(k+1)}) + \sum_{k=0}^{\infty} \eta_k.
\]

Now, since \( f \) is bounded below we have \(-f(x_k) \leq -C \) for all \( k \), and due to summability of the sequence \( \{\eta_k\} \), we obtain

\[
\sum_{k=1}^{\infty} \Delta^2_{\ell(k)-1} < +\infty,
\]
so \( \lim_{k \to \infty} \Delta^2_{\ell(k)-1} = 0 \), as we want to prove. \( \blacksquare \)

In consequence, we observe that

\[
\lim_{k \to \infty} \Delta_{\ell(k)-1} = 0,
\]
since the steps \( \Delta_k \) are small enough and positives.

Now we define set of index

\[
U = \{l(1) - 1, l(2) - 1, l(3) - 1, \ldots\}
\]
where \( \{l(k)\} \) is the sequence of index defined in the Proposition 1.

The following two results have been demonstrated by Kolda, Lewis and Torczon in [10].

**Proposition 3** Let \( x \in \Omega \) and \( \varepsilon \leq 0 \), and let \( K = K(x, \varepsilon) \) and \( K^\circ = K^\circ(x, \varepsilon) \) for the bound constrained problem (7). Let \( G_{K^\circ} \subseteq D_{\mathbb{R}}^\circ \) the set of generators of \( K^\circ \). Then, if [\(-\nabla f(x)\rceil_{K^\circ} = 0 \), there is \( d \in G_{K^\circ} \) such that

\[
\frac{1}{\sqrt{n}}\|[\nabla f(x)\rceil_{K^\circ} \leq -\nabla f(x)^T d.
\]

**Proposition 4** Let \( x \in \Omega \) and \( \varepsilon \geq 0 \), and let \( K^\circ = K^\circ(x, \varepsilon) \) and \( K = K(x, \varepsilon) \) for the bound constrained problem (7). Then

\[
\chi(x) \leq \|[\nabla f(x)\rceil_{K^\circ} + \sqrt{n}\|[\nabla f(x)\rceil_{K^\circ} \varepsilon.
\]

Next, we present the main global convergence result of nmps algorithm.

**Theorem 3** Let \( f : \mathbb{R}^n \to \mathbb{R} \) be continuously differentiable, and suppose \( \nabla f(x) \) is Lipschitz continuous with constant \( L \), \( \|[\nabla f(x)\| \leq \gamma \), for all \( x \in \Omega \) and \( \{f(x_k)\}_{k \in \mathbb{N}} \) bounded below. If \( \{x_k\}_{k \in U} \) is the sequence generated by the nmps algorithm then

\[
\chi(x_k) \leq \sqrt{n}(L + \gamma)\Delta_k \text{ for all } k \in U.
\]

**Proof.** We will consider two cases.

**Case 1.** If \( x_k + \Delta_k d \notin \text{int}(\Omega) \) for all \( d \in D_{\mathbb{R}} \), then \( x_k + \Delta_k d \) is either on the boundary of or outside of \( \Omega \) for all directions \( d \in D_{\mathbb{R}} \).

In other words, if \( l(i) \leq x_k(i) \leq u(i) \) then \( x_k(i) - \Delta_k \leq l(i) \) and \( x_k(i) + \Delta_k \geq u(i) \) for all \( i = 1, 2, \ldots, n \).

The last inequalities imply that if \( x_k + \omega \in \Omega \), the vector \( \omega \) cannot have their components greater than \( \Delta_k \), that is, \( \omega(i) \leq \Delta_k \) for all \( i \). Therefore, \( \|\omega\| \leq \sqrt{n}\Delta_k \).
So,
\[
\chi(x_k) = \max_{x_k + \omega \in \Omega, \|\omega\| \leq 1} -\nabla f(x_k)^T \omega
\leq \max_{x_k + \omega \in \Omega, \|\omega\| \leq 1} \|\nabla f(x_k)\| \|\omega\|
\leq \|\nabla f(x_k)\| \sqrt{\Delta_k}
\leq \sqrt{n} \gamma \Delta_k,
\]
which completes the proof for the Case 1.

**Case 2.** Now we suppose that there is at least \(d \in D \oplus \) such that \(x_k + \Delta_k d \in \text{int}(\Omega)\). Thus, the cone \(K^\circ(x_k, \Delta_k)\) is generated by all the directions \(d \in D \oplus \) such that \(x_k + \Delta_k d \in \text{int}(\Omega)\).

By the mean value theorem, we have that
\[
f(x_k + \Delta_k d_k) - f(x_k) = \Delta_k \nabla f(x_k + \lambda_k \Delta_k d_k)^T d_k,
\]
for some \(\lambda_k \in [0, 1]\).

Since \(k \in U\), this implies
\[
0 \leq f(x_k + \Delta_k d_k) - f_{\max}(x_k) - \eta_k + \Delta_k^2.
\]
Taking into account that \(-f_{\max}(x_k) \leq -f(x_k)\) and \(\eta_k > 0\) for all \(k\), we obtain
\[
0 \leq f(x_k + \Delta_k d_k) - f(x_k) + \Delta_k^2.
\]
Then, we replace (5) in (6)
\[
0 \leq \Delta_k \nabla f(x_k + \lambda_k \Delta_k d_k)^T d_k + \Delta_k^2.
\]
Next, we divide the last inequality by \(\Delta_k\) and adding \(-\nabla f(x_k)^T d_k\), we get the following inequality
\[
-\nabla f(x_k)^T d_k \leq (\nabla f(x_k + \lambda_k \Delta_k d_k) - \nabla f(x_k))^T d_k + \Delta_k.
\]
Using the Proposition 3 we have
\[
\frac{1}{\sqrt{n}} \|[\nabla f(x)]^\circ\| \leq (\nabla f(x_k + \lambda_k \Delta_k d_k) - \nabla f(x_k))^T d_k + \Delta_k.
\]
Then by the Cauchy-Schwarz inequality, the fact that \(\|d_k\| = 1\) for all \(k\) and the boundedness hypothesis of the gradient, we obtain
\[
\frac{1}{\sqrt{n}} \|[\nabla f(x)]^\circ\| \leq \|(\nabla f(x_k + \lambda_k \Delta_k d_k) - \nabla f(x_k))\| + \Delta_k \leq L \Delta_k + \Delta_k.
\]
In consequence,
\[
\|[\nabla f(x)]^\circ\| \leq \sqrt{n} L \Delta_k + \sqrt{n} \Delta_k \leq \sqrt{n} L \Delta_k.
\]
Finally, combining the Proposition 4 with the above result
\[
\chi(x) \leq \|[\nabla f(x)]^\circ\| + \sqrt{n} \|[\nabla f(x)]^\circ\| \varepsilon \leq \|[\nabla f(x)]^\circ\| + \sqrt{n} \gamma \Delta_k \leq \sqrt{n} L \Delta_k + \sqrt{n} \gamma \Delta_k
\]
consequently,
\[
\chi(x) \leq \sqrt{n} (L + \gamma) \Delta_k,
\]
and the proof is complete.

\[\square\]
5 Numerical results

In this section we show and analyse the numerical results obtained using our nonmonotone pattern search bound constrained optimization \texttt{nmps} algorithm. All the numerical experiments were executed on a computer with a 2.3 GHz Intel Core i5–6200u processor (8 GB RAM). We implemented the \texttt{nmps} algorithm in \texttt{matlab} R2016b 64-bit.

To the purpose of carefully analyse the performance of our algorithm we decided to organize our study in two parts. First, we compare the \texttt{nmps} algorithm with the \texttt{patternsearch} routine from \texttt{matlab}‘s optimization toolbox, since both algorithms are based on pattern search methods. Then, we study the performance of \texttt{nmps} algorithm using different line search strategies \cite{7, 17, 25, 29} and the classical Armijo’s rule \cite{2}.

We have selected a set of 63 bound constrained problems from Hock–Schittkowski collection \cite{14}. Since this collection has only 9 bound constrained problems, we have modified other 54 problems with general constraints, extracting the linear and nonlinear constraints from each one of them. The detailed list of these problems and their characteristics is provided in Table 1.

As it is usual in derivative-free optimization articles, we are interested in the number of functional values needed to satisfy the stopping criteria, which are: reaching a sufficiently small step length \((\Delta_k < \Delta_{tol})\), attaining the maximum number of function evaluations \(MaxFE\) or attaining the maximum number of iterations \(MaxIt\). We adopt the convergence test proposed in \cite{25} to measure the ability of an algorithm to improve an initial approximation and to declare that a problem has been solved if the following condition holds

\[ f(x_0) - f(x) \geq (1 - \tau)(f(x_0) - f_L), \tag{7} \]

where \(x_0\) is the initial feasible approximation, \(\tau > 0\) is the level of accuracy and \(f_L\) is the smallest functional value obtained among the considered solvers. We use the performance profile graphs \cite{11, 25} to illustrate the results obtained with \cite{7}.

Given \(P\) the set of problems, \(|P|\) denotes the cardinality of \(P\) and \(S\) the set of considered solvers. The performance profile of a solver \(s \in S\) is defined as the fraction of problems where the performance ratio is at most \(\alpha\), that is, \(\rho_s(\alpha) = \frac{1}{|P|} \text{size} \{p \in P : r_{p,s} \leq \alpha\}\), where \(r_{p,s} = \frac{t_{p,s}}{\text{min}(t_{p,s}, t_{L,s})}\), \(t_{p,s}\) is the number of function evaluations required to satisfy the convergence test \cite{7}.

We used the same initial approximation \(x_0\) as indicated in \cite{14}, projecting onto the bound constraints if the initial approximation was not feasible. After some preliminary tests we adopted \(\eta_k = 1.1^{-k}\) for all \(k\). Finally, the following algorithmic parameters were set: \(\Delta_0 = 1.0\), as the initial step length, \(M = 15\), \(MaxFE = 2500\), \(MaxIt = 5000\) and \(\Delta_{tol} = TOL = 10^{-6}\).

It is worth mentioning three important implementation details of our algorithm. First, the function \(f\) is evaluated in all possible coordinate directions and the accepted new approximation is such that produces the minimum functional value of \(f(x_k + \Delta_k d)\). Second, the step length is updated using the following scheme \(\Delta_{k+1} = \min\{1, 2\Delta_k\}\). Third, the accepted points are stored in a memory for the purpose to avoid revisiting older points without slow down the implementation. See \cite{19}.

5.1 Comparison of performance profiles between \texttt{nmps} and \texttt{patternsearch}

We tested our algorithm \texttt{nmps} using the set of test problems and we compared it with the well established routine \texttt{patternsearch} from \texttt{matlab}. Since both codes are based on a pattern search scheme, we set the same algorithmic parameters. The numerical results are shown in Table 2.

In Figure \cite{7} we show the performance profile pictures using condition \cite{7} with three levels of accuracy: \(\tau = 10^{-1}\), \(\tau = 10^{-3}\) and \(\tau = 10^{-5}\), where a smaller value of \(\tau\) means the satisfaction of condition \cite{7} is more strict. In a performance profile plot, the top curve represents the most efficient method within a factor \(\tau\) of the best measure. When both methods match with the best result, then they are both counted as successful. This means that the sum of the successful percentages may exceed 100%.
| Prob. | N°. HS | n | constraints | objective function |
|-------|--------|---|-------------|-------------------|
| 1     | 1      | 2 | 1           | Generalized polynomial |
| 2     | 2      | 2 | 1           | Generalized polynomial |
| 3     | 3      | 2 | 1           | Generalized polynomial |
| 4     | 4      | 2 | 2           | Generalized polynomial |
| 5     | 5      | 2 | 4           | Generalized polynomial |
| 6     | 25     | 3 | 6           | Sum of squares |
| 7     | 38     | 4 | 8           | Generalized polynomial |
| 8     | 45     | 5 | 10          | Constant |
| 9     | 110    | 10| 20          | General |
| 10    | 13     | 2 | 2           | Quadratic |
| 11    | 15     | 2 | 1           | Generalized polynomial |
| 12    | 16     | 2 | 3           | Generalized polynomial |
| 13    | 17     | 2 | 3           | Generalized polynomial |
| 14    | 18     | 2 | 4           | Quadratic |
| 15    | 19     | 2 | 4           | Generalized polynomial |
| 16    | 20     | 2 | 2           | Generalized polynomial |
| 17    | 21     | 2 | 4           | Quadratic |
| 18    | 23     | 2 | 4           | Quadratic |
| 19    | 24     | 2 | 2           | Generalized polynomial |
| 20    | 30     | 3 | 6           | Quadratic |
| 21    | 31     | 3 | 6           | Quadratic |
| 22    | 32     | 3 | 3           | Quadratic |
| 23    | 33     | 3 | 4           | Generalized polynomial |
| 24    | 34     | 3 | 6           | Linear |
| 25    | 35     | 3 | 3           | Quadratic |
| 26    | 36     | 3 | 6           | Generalized polynomial |
| 27    | 37     | 3 | 6           | Generalized polynomial |
| 28    | 41     | 4 | 8           | Generalized polynomial |
| 29    | 42     | 4 | 2           | Quadratic |
| 30    | 44     | 4 | 4           | Quadratic |
| 31    | 53     | 5 | 10          | Quadratic |
| 32    | 54     | 6 | 12          | General |
| 33    | 55     | 6 | 8           | General |
| 34    | 57     | 2 | 2           | Sum of squares |
| 35    | 59     | 2 | 4           | General |
| 36    | 60     | 3 | 6           | Generalized polynomial |
| 37    | 62     | 3 | 6           | General |
| 38    | 63     | 3 | 3           | Quadratic |
| 39    | 64     | 3 | 3           | Generalized polynomial |
| 40    | 65     | 3 | 6           | Quadratic |
| 41    | 66     | 3 | 6           | Linear |
| 42    | 68     | 4 | 8           | General |
| 43    | 69     | 4 | 8           | General |
| 44    | 71     | 4 | 8           | Generalized polynomial |
| 45    | 72     | 4 | 8           | Linear |
| 46    | 73     | 4 | 4           | Linear |
| 47    | 74     | 4 | 8           | Generalized polynomial |
| 48    | 75     | 4 | 8           | Generalized polynomial |
| 49    | 76     | 4 | 4           | Quadratic |
| 50    | 80     | 5 | 10          | General |
| 51    | 81     | 5 | 10          | General |
| 52    | 83     | 5 | 10          | Quadratic |
| 53    | 84     | 5 | 10          | Quadratic |
| 54    | 86     | 5 | 5           | Generalized polynomial |
| 55    | 93     | 6 | 6           | Generalized polynomial |
| 56    | 101    | 7 | 14          | Generalized polynomial |
| 57    | 102    | 7 | 14          | Generalized polynomial |
| 58    | 103    | 7 | 14          | Generalized polynomial |
| 59    | 104    | 8 | 16          | Generalized polynomial |
| 60    | 106    | 8 | 16          | Linear |
| 61    | 108    | 9 | 2           | Quadratic |
| 62    | 114    | 10| 20          | Quadratic |
| 63    | 119    | 16| 32          | Generalized polynomial |

Table 1: Characteristics of test problems.
Figure 1: Performance profiles of nmps and patternsearch.

In Figure 1a with $\tau = 10^{-1}$, we observe that nmps algorithm is the best solver in the 82% of the set problems while patternsearch does it in 71%. We also see, within a factor of 1.7 of the best solver, both algorithms have a similar behaviour and the performance profile shows these algorithms can solve a problem with a probability of 0.87 with respect to the best solver. Finally, if the goal is to solve efficiently 95% of the problems, the nmps algorithm accomplishes this by using 2.4 times the minimum number of function evaluations while patternsearch needs a factor of 7.5.

Now, when we increase the level of accuracy to $\tau = 10^{-3}$ (Figure 1b), we note that nmps wins in the 79% of the problems in comparison with the 68% of patternsearch. Also, both solvers are equivalent if the solution is required within a factor of 1.7 of the best solver, with a probability of 0.84. Although both solvers can reach the solution in, at most, 95% of the problems, our algorithm was closer than the other since it solved 94% of them, using 3.7 times the minimum number of function evaluations.

Finally, we observe in Figure 1c that the performance of both solvers deteriorate using $\tau = 10^{-5}$ as the level of accuracy. In any case, the algorithm nmps wins in 74% of the problems meanwhile patternsearch wins in 62%. As before, nmps algorithm performs better than patternsearch if you consider a solver that finds the solution using 1.7 times the minimum number of function evaluations, with a probability of 0.8. In such a case, we could expect at most that nmps algorithm solves 87% of the problems while patternsearch solves 81%.

We conclude that, regardless the level of accuracy, our algorithm outperforms the patternsearch routine with the set of test problems considered. In fact, our algorithm always has a probability 10% greater than patternsearch to get the solution.
In the next subsection we will test our algorithm using other line search strategies in order to analyse and understand the advantages of employing different nonmonotone line search procedures.

5.2 Comparison of performance profiles using other line search strategies

Recently, some authors proposed different nonmonotone line search strategies for solving unconstrained minimization problems. In this case, they have considered different patterns as solvers. In this case, the best solver, \texttt{patternsearch}, solving 89% of the problems. Later, within a factor of 2, \texttt{nmps}, with 0\% probability, is the most successful solver with a probability of 0\%.

The first approach, which is called C-line search and we have implemented in \texttt{Cpatternsearch} algorithm, is similar to the nonmonotone line search condition [2], where \( f_{\text{max}}(x_k) \) is replaced by the sequence \( \{C_k\} \) given by

\[
Q_{k+1} = r_k Q_k + 1, \quad C_{k+1} = \frac{r_k Q_k(C_k + \eta_k) + f_{k+1}}{Q_{k+1}}
\]

with \( Q_0 = 1, C_0 = f(x_0), r_k \in [0, 1] \) and \( \{\eta_k\} \) satisfying \( \sum_{k=0}^{\infty} \eta_k = \eta < \infty \) for all \( k = 0, 1, 2, \ldots \).

The second strategy, which is called \( \lambda \)-line search and we have implemented in \texttt{lpatternsearch} algorithm, is also analogous to [2] but in this case \( f_{\text{max}}(x_k) \) is defined by

\[
f_{\text{max}}(x_k) = \max \{ f(x_k), \sum_{r=0}^{m(k)-1} \lambda_k f(x_{k-r}) \}
\]

with \( M \in \mathbb{N}, m(k) = \min \{k, M-1\}, \lambda_k, \lambda \) and \( \sum_{r=0}^{m(k)-1} \lambda_{k+r} = 1 \) for all \( k = 0, 1, 2, \ldots \).

Finally, we have adapted the classical Armijo’s rule to our bound constrained problem [1] at the minimum number of function evaluations, meanwhile \texttt{nmps}.

Again, we chose \( \eta_k = 1.1^{-k} \) for all \( k \), for the three new conditions. Also, we adopted \( M = 15 \) and \( \lambda_k = 1/m(k) \) for all \( r \) in \texttt{lpatternsearch} and \( r_k = 0.85 \) for all \( k \) in \texttt{Cpatternsearch}. Next, we show the performance profiles with convergence test [2] for solving our set of test problems using \texttt{nmps}, \texttt{patternsearch}, \texttt{Cpatternsearch}, \texttt{Apatternsearch} and \texttt{armijo} algorithms. The numerical results are also presented in Table 2.

In Figure 2a, for \( \tau = 10^{-1} \), we see that \texttt{lpatternsearch} attains the best performance in 76% of the problems, followed by \texttt{nmps} with 59\% and \texttt{Cpatternsearch} with 57\%, below we find \texttt{patternsearch} and \texttt{armijo} with 56\% and 51\% respectively. We also observe, within a factor of 2.5 of the best solver, \texttt{lpatternsearch} reaches the greater probability of solving a problem (around 0.92) and \texttt{nmps} follows it with 0.9. Furthermore, in this case, \texttt{patternsearch} is the solver with the lowest performance, with a probability of 0.76. Moreover, with this level of accuracy, we note that \texttt{armijo} solves almost 97\% of the problems using a factor of 4.9 times the minimum number of function evaluations, meanwhile \texttt{nmps} has the same behaviour requiring a factor of 5.75.

Figure 2b displays the performance profile for \( \tau = 10^{-3} \), increasing in this way the level of accuracy. Once more, \texttt{lpatternsearch} is the most successful solver with a probability of 0.63, followed by \texttt{Cpatternsearch} and \texttt{nmps} with a probability of 0.62 and 0.6, respectively. In the last positions, we find \texttt{armijo} and \texttt{patternsearch} with 0.52 and 0.51, respectively. Now, within a factor of 2.5 of the best solver, \texttt{nmps} exhibits the best performance solving 89\% of the problems. Later, \texttt{Cpatternsearch} and \texttt{armijo} get 87\% and 85\% on the resolution of our set of problems. With this level of accuracy, the most we can expect is to solve 94\% of the problems within a factor of 3.8 of the best solver. In the same sense, \texttt{Cpatternsearch} and \texttt{nmps} present a similar behaviour.

Finally, Figure 2c presents the performance profile for \( \tau = 10^{-5} \), requiring in this way a greater descent in the objective function. We observe that \texttt{Cpatternsearch} wins in 65\% of the problems, followed by \texttt{nmps} and \texttt{Apatternsearch} with 57\%, \texttt{armijo} with 49\% and \texttt{patternsearch} with 44\%. We see, within a factor of 2.5 of the best solver, \texttt{Cpatternsearch} obtains the higher probability for solving a problem (0.87), followed by \texttt{nmps} (0.84). In this figure, as opposed to Figures 2a and 2b, we can see how the performance of the different methods are slightly separated from each other. Also, \texttt{Cpatternsearch} algorithm is always on top of all the remaining solvers. In this case, \texttt{Cpatternsearch} achieves the solution in 92\% of the problems within a factor of 2.79.
| Prob. | MaxFE | MaxIt | stop | Cpatternsearch | It. | TOL |
|-------|--------|--------|------|----------------|-----|-----|
| 1     | 0.0000 | 0.0000 | 150  | 100            | 5   | 0.01|
| 2     | 0.0000 | 0.0000 | 100  | 100            | 5   | 0.01|
| 3     | 0.0000 | 0.0000 | 150  | 100            | 5   | 0.01|
| 4     | 0.0000 | 0.0000 | 100  | 100            | 5   | 0.01|
| 5     | 0.0000 | 0.0000 | 150  | 100            | 5   | 0.01|
| 6     | 0.0000 | 0.0000 | 100  | 100            | 5   | 0.01|
| 7     | 0.0000 | 0.0000 | 150  | 100            | 5   | 0.01|
| 8     | 0.0000 | 0.0000 | 100  | 100            | 5   | 0.01|
| 9     | 0.0000 | 0.0000 | 150  | 100            | 5   | 0.01|
| 10    | 0.0000 | 0.0000 | 100  | 100            | 5   | 0.01|

Table 2: Results of numerical experiments
followed by \texttt{nmps} that solves 87\% of the problems using 3.6 times the minimum number of function evaluations. The other solvers can only solve at most 80\% of the problems employing greater factors.

![Performance profile with $\tau=1e-1$](image1)

![Performance profile with $\tau=1e-3$](image2)

![Performance profile with $\tau=1e-5$](image3)

(a) 

(b) 

(c)

Figure 2: Performance profiles for the five strategies

From the analysis of the figures included in Figure 2 we can obtain the following conclusions. First, in most cases the use of a nonmonotone line search of the kind (2), (6) or (9) turns into an advantage in the performance of the algorithms compared to the classical Armijo’s rule (10). In other words, a greater effort devoted to building $f_{\text{max}}(x_k)$ or $C_k$ results in a decrease in the number of function evaluations carried out by the algorithm, which is one of the main goals in derivative-free methods. Second, although the strategies (2) and (9) have similar definitions, the above analysis shows, while $\text{patternsearch}$ reduces its performance as the level of accuracy increases, \texttt{nmps} remains stable for all values of $\tau$. So, if we should choose between this two strategies, \texttt{nmps} would be the most suitable. Third, the \texttt{Cpatternsearch} algorithm could be considered as the solver with the best performance because it is always above the other solvers as the level of accuracy increases. Our algorithm \texttt{nmps}, in its turn, seems to be a good competitor to \texttt{Cpatternsearch}, attaining a similar performance with respect to the latter in several cases. Also, \texttt{nmps} always obtains the second place regarding the probability to solve all the problems. At the end, to our surprise, we observe that the performance of \texttt{patternsearch} algorithm is in many cases below all remaining methods.

6 Conclusions

In this paper, we have proposed a new pattern search algorithm \texttt{nmps} to solve bound constrained optimization problems, which uses a nonmonotone line search strategy for accepting the new iterate. We have proved that,
under mild assumptions, we can guarantee global convergence of our method to a KKT point. This result is strongly based on the relationship between the step length and the stationarity measure (defined by Conn et al. in [8]), and it was proved in Section 4. Furthermore, we have performed several numerical experiments where we have compared the performance of our algorithm to other line search strategies that were implemented in \texttt{Patternsearch}, \texttt{Patternsearch}, \texttt{Patternsearch} and \texttt{Armijo} algorithms. The benchmark results were satisfactory, so we can conclude that the \texttt{nmps} algorithm is competitive, compared to the other solvers, as the numerical experiments reveal. We are currently working on an extension of the \texttt{nmps} algorithm to linearly constrained optimization problems, taking into account the directions generated by the linear constraints.

**Disclosure statement**

No potential conflict of interest was reported by the authors.

**Funding**

This work was partially supported by research project Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) PIP 112–201101–00050, PUE 22920160100134CO and Secretaría de Ciencia y Tecnología (SECyT-UNC).

**References**

[1] P. Alberto, F. Nogueira, H. Rocha, L. Vicente. *Pattern search methods for user-provided points: Application to molecular geometry problems*, SIAM J. Optim., v. 14(4), pp. 1216–1236, 2004.

[2] L. Armijo. *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific J. Math., v. 16, pp 1–3, 1966.

[3] M. Arouxé, N. Echebest and E. A. Pilotta. *Active-set strategy in Powell’s method for optimization without derivatives*, Computational & Applied Mathematics, v. 30 (1), pp. 171–196, 2011.

[4] C. Audet and J. E. Dennis. *Analysis of generalized pattern searches*, SIAM J. Optim., v. 13(3), pp. 889–903, 2003.

[5] E. Birgin, J. M. Martínez and M. Raydán. *Inexact spectral projected gradient methods on convex sets*, IMA Journal of Numerical Analysis, v. 23, pp. 539–559, 2003.

[6] A. J. Booker, J. E. Dennis, P. D. Frank, D. B. Serafini and V. Torczon. *Optimization using surrogate objectives on a helicopter test example*, in Computational Methods for optimal design and control. J. T. Borggaard, J. Burns, E. Cliff and S. Schreck, eds., pp. 49–58, Birkhäuser, Boston, 1998.

[7] W. Cheng. *A derivative-free nonmonotone line search and its application to the spectral residual method*, IMA Journal of Numerical Analysis, v. 29, pp. 814–825, 2009.

[8] A. Conn, N. Gould and P. Toint. *Trust region methods*, MPS/SIAM Ser. Optim. 1, SIAM, Philadelphia, 2000.

[9] A. Conn, K. Scheinberg and L. Vicente. *Introduction to derivative-free optimization*, SIAM, 2009.

[10] M. A. Diniz–Ehrhardt, J. M. Martínez and M. Raydán. *A derivative-free nonmonotone line-search technique for unconstrained optimization*, Journal of Computational and Applied Mathematics, v. 219, pp. 383–397, 2008.

[11] E. Dolan and J. Moré. *Benchmarking optimization software with performance profiles*, Mathematical programming, v. 91, pp. 201–213, 2002.

[12] R. Duvigneau and M. Visonneau. *Hydrodynamics design using a derivative-free method*, Struct. Multidiscip. Optim. v. 28, pp. 195–205, 2004.

[13] L. Grippo, F. Lampariello and S. Lucidi. *A nonmonotone line search technique for Newtons method*, SIAM J. Numer. Anal., v. 23, pp. 707-716, 1986.
[14] W. Hock and K. Schittkowski. Test examples for nonlinear programming codes, Lecture Notes in Economics and Mathematical Systems, v. 187, 1981.

[15] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems, SIAM J. Optim., v. 8(2), pp. 212–229, 1961.

[16] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods, SIAM Review, v. 45(3), pp. 385–482, 2003.

[17] N. Krejić, Z. Lužanin, F. Nikolovski and I. Stojkovska. A nonmonotone line search method for noisy minimization, Optimization Letters, v. 9(7), pp. 1371-1391, 2015.

[18] W. La Cruz, J. M. Martínez and M. Raydán. Spectral residual method without gradient information for solving large-scale nonlinear systems of equations, Math. Comput. v. 75, pp. 1449–1466, 2006.

[19] R. M. Lewis, A. Shepherd and V. Torczon. Implementing generating set search methods for linearly constrained minimization, SIAM Journal on Scientific Computing, v. 29(6), pp. 2507–2530, 2007.

[20] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization, SIAM J. Optim., v. 9(4), pp. 1082–1099, 1999.

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

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

[23] A. L. Marsden, M. Wang, J. E. Dennis and P. Moin. Optimal aeroacoustic shape design using the surrogate management framework, Optim. Eng., v. 5, pp. 235–262, 2004.

[24] J. Meza and M. L. Martínez. On the use of direct search methods for the molecular conformation problem, J. Comput. Chem., v. 15(6), pp. 627-632, 1994.

[25] J. Moré and S. Wild. Benchmarking derivative-free optimization algorithms, SIAM J. Optim., v. 20, pp. 172–191, 2009.

[26] F. Nikolovski and I. Stojkovska. New derivative-free nonmonotone line search methods for unconstrained minimization, Proceedings of the Fifth International Scientific Conference - FMNS2013, v. 1, Mathematics and Informatics, pp. 47–53, 2013.

[27] R. Oeurray and M. Bierlaire. A new derivative-free algorithm for the medical image registration problem, International Journal of Modelling and Simulation, v. 27(2), pp. 115-124, 2007.

[28] V. Torczon. On the convergence of pattern search algorithms. SIAM J. Optim., v. 7(1), pp. 1–25, 1997.

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