CURVILINEAR STABILIZATION TECHNIQUES FOR TRUNCATED NEWTON METHODS IN LARGE SCALE UNCONSTRAINED OPTIMIZATION

STEFANO LUCIDI†, FRANCESCO ROCETICH†, AND MASSIMO ROMA†

Abstract. The aim of this paper is to define a new class of minimization algorithms for solving large scale unconstrained problems. In particular we describe a stabilization framework, based on a curvilinear linesearch, which uses a combination of a Newton-type direction and a negative curvature direction. The motivation for using negative curvature direction is that of taking into account local nonconvexity of the objective function. On the basis of this framework, we propose an algorithm which uses the Lanczos method for determining at each iteration both a Newton-type direction and an effective negative curvature direction. The results of extensive numerical testing are reported together with a comparison with the LANCELOT package. These results show that the algorithm is very competitive, which seems to indicate that the proposed approach is promising.

Key words. large scale unconstrained optimization, Newton-type method, negative curvature direction, curvilinear linesearch, Lanczos method

AMS subject classifications. 90C30, 65K05

PII. S1052623495295250

1. Introduction. In this work, we deal with the definition of new efficient unconstrained minimization algorithms for solving large scale problems. Namely, we consider the minimization problem

\[ \min_{x \in \mathbb{R}^n} f(x), \]

where \( f \) is a real valued function on \( \mathbb{R}^n \) and \( n \) is large. We assume throughout that both the gradient \( g(x) = \nabla f(x) \) and the Hessian matrix \( H(x) = \nabla^2 f(x) \) of \( f \) exist and are continuous.

Many algorithms have been proposed for solving such problems; in particular, we focus our attention on truncated Newton-type algorithms since they retain the good Newton-like convergence rate properties.

Most of the Newton-type algorithms for large scale minimization problems perform quite well on test problems which present small regions where the objective function is nonconvex, but they have trouble in minimizing test functions where these regions are large. This behavior can be clearly evidenced by running these algorithms on the CUTE collection [2], which includes a variety of functions with large nonconvexity regions. As far as the authors are aware, the only Newton-type algorithm which performs sufficiently well on all the CUTE test problems is the truncated Newton algorithm implemented in the LANCELOT software package [3]. In our opinion this is due to the fact that, unlike the other ones, this algorithm is able to take the local nonconvexity of the objective function into account. The LANCELOT algorithm is a trust region method, and at each iteration \( k \), its idea is to compute a step using a truncated conjugate gradient method to approximately minimize the quadratic model of
the objective function within a trust region. However, whenever the conjugate gradient method detects a nonascent negative curvature direction (i.e., a direction $d_k$ such that $g(x_k)^T d_k \leq 0$ and $d_k^T H(x_k) d_k < 0$), the minimization of the model is stopped and the new point is obtained by performing a significant movement along this direction. Since the direction $d_k$ is one in which the function decreases and its directional derivative also is decreasing (at least locally), this movement can contribute significantly to hasten the search for a region in which the objective function is convex so that the properties of a Newton-type method may apply.

The previous remarks seem to indicate that the use of the negative curvature directions could be very beneficial in solving large scale minimization problems. This encourages further investigation in defining new algorithms based on the attempt to exploit more deeply the information about the curvature of $f$, which can be obtained by means of negative curvature directions. In fact, although the strategy of the LANCELOT algorithm to compute a negative curvature direction is very simple and efficient, some of its aspects can be improved. In particular, even if we are in a region where the Hessian is not positive semidefinite, the conjugate gradient scheme of the LANCELOT algorithm can be truncated before finding a negative curvature direction. Furthermore, even if such a direction is detected, it could convey poor information about the negative curvature of $f$, and hence, it could be unable to influence the behavior of the algorithm significantly. Therefore, it could be worthwhile, from the computational point of view, to define new algorithms that, without an excessive computational burden, try to use negative curvature directions more effective than the simple ones produced in the LANCELOT algorithm.

Negative curvature directions which can be considered the best from the theoretical point of view are those directions that guarantee the skill of the minimization algorithm in escaping regions where the objective function is strictly nonconvex. Such directions are the basis for defining unconstrained minimization algorithms which converge toward stationary points where the Hessian matrix is positive semidefinite. This second order convergence can be ensured by exploiting, to a great extent, the negative curvature information of the objective function contained in the second order derivatives. Such information is obtained by requiring that the negative curvature direction has some resemblance to an eigenvector of the Hessian associated with its minimum eigenvalue. More in particular, it is usually needed to compute a direction of negative curvature $d_k$ such that (see, for example, [15, 16, 17, 24])

\[
\begin{align*}
&d_k^T H(x_k) d_k \to 0 \implies \min [0, \lambda_{\min}(H(x_k))] \to 0, \\
&(1.2)
\end{align*}
\]

where $\lambda_{\min}(H(x_k))$ is the minimum eigenvalue of the Hessian matrix.

Algorithms converging toward stationary points where the Hessian matrix is positive semidefinite have been defined both in a trust region approach (see, for example, [9, 24]) and in a linesearch approach (see, for example, [15, 16, 17]). In particular, the second approach originated from [15] and [17], which propose a minimization method based on a monotone curvilinear linesearch of the following form:

\[
(1.3) \quad x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k,
\]

where $\alpha_k$ is a step size computed by an Armijo-type technique, $s_k$ is a Newton-type direction which ensures global convergence and a superlinear rate of convergence, and $d_k$ is a negative curvature direction which guarantees convergence toward points where the Hessian matrix is positive semidefinite.
In [8] the approach proposed in [15, 17] has been embedded in a nonmonotone stabilization strategy by defining a new general algorithmic model of the type of [13]. The motivation of this extension arose from the fact that substantial computational benefits can be obtained by abandoning the enforcement of generating monotonically decreasing objective function values. These benefits are particularly noticeable in solving minimization problems with highly nonlinear or ill-conditioned objective functions. The effectiveness of using a nonmonotone strategy was shown both in the linesearch approach (see, for example, [11, 13, 27]) and in the trust region approach (see, for example, [7, 28, 29]). In [8] a sample algorithm, based on the nonmonotone curvilinear algorithmic model proposed in this paper, has been implemented and shown to outperform other standard Newton-type algorithms, in particular the LANCELOT algorithm, on a large set of small and medium dimension test problems. These numerical results seem to suggest that the combined use of a nonmonotone stabilization technique and a good negative curvature direction can be the basis for defining new efficient unconstrained minimization algorithms.

The aim of this work is to extend the approach proposed in [8] to the case of large scale problems. The main difficulty in tackling such minimization problems is the fact that the use of condition (1.2) as a criterion for selecting a good negative curvature direction is not practical. In fact, the computation of a direction which satisfies (1.2) requires heavy and expensive matrix manipulations, which limits its applicability when the dimension of the problems becomes large.

In section 2 of this paper, as a first step to overcome this difficulty, we prove that the conditions required on the directions of negative curvature to guarantee the convergence toward stationary points where the Hessian matrix is positive semidefinite can be weakened. In fact, we show that the convergence properties of the general stabilization framework of [8] (hence of the algorithms proposed in [15, 17]) still hold under weaker assumptions on the negative curvature directions used. In particular, these new conditions imply that it is sufficient to compute directions of negative curvature \( d_k \) which satisfy

\[
\begin{align*}
\langle d_k, H(x_k) d_k \rangle & \to 0 \\
g(x_k) & \to 0
\end{align*}
\]

implies \( \min \{ 0, \lambda_{\min}(H(x_k)) \} \to 0 \).

This result, besides being a step toward the definition of large scale minimization algorithms which converge toward stationary points where the Hessian matrix is positive semidefinite, gives useful hints on how to compute efficiently a good direction of negative curvature. In fact, roughly speaking, condition (1.4) indicates that an algorithm based on the curvilinear search (1.3) is able to leave a region where the objective function is nonconvex and its gradient is not too small by exploiting only the properties of the direction \( s_k \). Therefore, an expensive computation of a negative curvature direction \( d_k \) which satisfies a condition similar to (1.2) is not necessary and justified when the gradient is sufficiently large. On the other hand, the contribution of a direction \( d_k \) which has a quite strict connection with an eigenvector of the Hessian corresponding to the most negative eigenvalue is essential to go away from a region of nonconvexity of the objective function in which the gradient is small.

In section 3 of this paper, drawing our inspiration from the previous result, we define a new minimization algorithm for large scale unconstrained problems which, at each iteration, computes both a truncated Newton direction and a negative curvature direction using the Lanczos method to approximately solve the Newton equation.
The choice of using the Lanczos algorithm as the iterative method for computing the search directions is due to the fact that some of its properties are very suitable for the approach described in this work. First of all, it does not break down if the Hessian matrix is indefinite. This implies that a good approximation of the Newton direction can be always calculated and, furthermore, that the process of gaining information on the negative curvature of the objective function is not stopped if the algorithm meets a negative curvature direction. Another interesting feature of the Lanczos algorithm which will be exploited in this paper is the possibility of obtaining good estimates of the smallest eigenvalue of the Hessian matrix and of one of the associated eigenvectors. In fact, if enough storage room is available to store a sequence of vectors produced by the Lanczos algorithm and if suitable assumptions are satisfied, then a negative curvature direction satisfying the new condition described in this work can be determined and, hence, the convergence of the algorithm toward stationary points where the Hessian matrix is positive semidefinite can be ensured.

In practice, in the case of large scale problems, this storage requirement cannot be satisfied and only a small number of vectors produced by the algorithm can be stored. However, also in this case, the theory and the computational experiences guarantee that the Lanczos algorithm can provide efficiently a negative curvature direction that, even if it does not satisfy condition (1.4), can be considered a good approximation of an eigenvector of the Hessian corresponding to the most negative eigenvalue. Therefore, we can conclude that the Lanczos algorithm allows us to compute a negative curvature direction which is a compromise between

- a negative curvature direction which satisfies condition (1.4) but which could be too expensive to compute when the gradient of $f$ becomes small,
- a negative curvature direction (like the one used by the LANCELOT algorithm) which is easy to compute but which could have no resemblance to any eigenvector of the Hessian corresponding to the most negative eigenvalue.

Finally, in section 4, we report the results of a large computational experimentation. One of the purposes of these numerical experiments has been to investigate the effect of the use of the negative curvature direction produced by the Lanczos method and the effect of the use of the nonmonotone stabilization strategy. To this aim, we have implemented four different algorithms belonging to the class described by the proposed algorithmic model: a monotone algorithm which does not use negative curvature directions, a monotone algorithm which uses negative curvature directions, a nonmonotone algorithm which does not use negative curvature directions and a nonmonotone algorithm which uses negative curvature directions. We have tested and compared them on a large set of standard large scale test problems available from the CUTE collection [2]. The results obtained show that both the use of the negative curvature direction and the nonmonotone stabilization technique play a significant role for improving the performance of the minimization method and indicate clearly that the most efficient algorithm is the one which uses both the negative curvature directions and the nonmonotone stabilization strategy.

Furthermore, in order to have a feel of the effectiveness of our approach, we have compared the results obtained by our best algorithm with those obtained by the default version and the version without preconditioner of the LANCELOT package [3]. The results of these comparisons indicate that the proposed approach is very promising and show that our algorithm is competitive with the LANCELOT algorithms. Even if no final conclusion can be drawn, these numerical results encourage further research.
on new large scale minimization algorithms based on the combined use of “sufficiently good” negative curvature directions and nonmonotone stabilization techniques.

2. Globalization algorithm. The aim of this section is to prove that the convergence properties of the nonmonotone stabilization algorithmic model proposed in [8] still hold under weaker assumptions on the negative curvature direction used. We refer to [8, 13, 15, 17] for a detailed discussion of the rationale behind the algorithm. Here we just recall that the approach is based on the following observations:

- The search directions $s_k$ and $d_k$ convey much more information on the minimization problem than single function evaluations, and under standard assumptions, the decrease of their magnitude can be used as an indication that the algorithm is converging.
- The widely used globalization techniques of enforcing the monotone decrease of the objective function values by controlling the magnitude of the step length can deteriorate the behavior of a Newton-type minimization algorithm (see, for example, [13]).

Therefore, the basic idea of the proposed algorithm is to produce the new point by automatically accepting unit step size ($\alpha_k = 1$) in (1.3) without checking the objective function value whenever the magnitude of $\|s_k\| + \|d_k\|$ is decreasing. Otherwise, the algorithm resorts to a nonmonotone linesearch technique to determine the step length.

More in particular, here we describe the steps of the algorithm while we detail the computation of the directions $s_k$ and $d_k$ in the next section.

**NONMONOTONE CURVILINEAR STABILIZATION ALGORITHM (NMC).**

Data: $x_0$, $\Delta_0 > 0$, $\delta \in (0, 1)$, $N \geq 1$, $M \geq 0$, $\sigma \in (0, 1)$ and $\gamma \in (0, \frac{1}{2})$.

Step 1: **Initialization**

Set $k = \ell = j = 0$, $\Delta = \Delta_0$; compute $f(x_0)$ and set $f_0^R = f_0 = f(x_0)$.

Step 2: **Test for convergence**

If $\|g(x_k)\| = 0$ and $H(x_k)$ is positive semidefinite stop.

Step 3: **Tests for automatic step acceptances**

If $k \neq \ell + N$ compute directions $s_k$ and $d_k$; then

(a) if $\|s_k\| + \|d_k\| \leq \Delta$, set $x_{k+1} = x_k + s_k + d_k$, $k = k + 1$, $\Delta = \delta \Delta$ and go to Step 2;

(b) if $\|s_k\| + \|d_k\| > \Delta$, compute $f(x_k)$ if necessary; then:

- if $f(x_k) \geq f_j^R$, backtrack to $x_\ell$;

- otherwise, if $k \neq \ell$ set $\ell = k$, $j = j + 1$, $f_j = f(x_k)$ and update $f_j^R$ according to

\[
(2.1) \quad f_j^R = \max_{0 \leq i \leq m(j)} f_{j-i}, \quad \text{where} \quad m(j) = \min[j, M];
\]

and go to Step 5.

Step 4: **Control after $N$ automatic step acceptances**

If $k = \ell + N$ compute $f(x_k)$ if necessary; then

(a) if $f(x_k) \geq f_j^R$, backtrack to $x_\ell$ and go to Step 5;

(b) if $f(x_k) < f_j^R$, set $\ell = k$, $j = j + 1$, $f_j = f(x_k)$ and update $f_j^R$ according to (2.1); compute directions $s_k$ and $d_k$; then:

- if $\|s_k\| + \|d_k\| \leq \Delta$, set $x_{k+1} = x_k + s_k + d_k$, $k = k + 1$, $\Delta = \delta \Delta$ and go to Step 2;

- otherwise go to Step 5.
Step 5: Nonmonotone linesearch

Compute $\alpha_k = \sigma^h$ where $h$ is the smallest nonnegative integer such that

$$f(x_k + \alpha_k^2 s_k + \alpha_k d_k) \leq f_j^R + \gamma \alpha_k^2 \left[ g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k \right],$$

set

$$x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k, \quad k = k + 1, \quad \ell = k, \quad j = j + 1, \quad f_j = f(x_k),$$

update $f_j^R$ according to (2.1) and go to Step 2.

Remark 2.1. We note that, for the sake of simplicity, in the algorithmic scheme we have omitted the explicit indication concerning the necessity to store the point $x_\ell$ and the directions $s_\ell$ and $d_\ell$ whenever we set $\ell = k$.

This algorithmic model derives from the combination of the nonmonotone globalization approach described in [8] and the curvilinear linesearch strategy introduced in [15] and [17]. In particular, if we set $d_k = 0$ for all $k$ in Algorithm NMC, we obtain exactly the algorithmic model of [13], while if we set $\Delta = 0$ and $M = 0$ we obtain the modified Newton methods of [15] and [17].

As we said before, in the algorithm the objective function is not necessarily evaluated at each iteration. We denote by $\ell$ the index of the last point produced by the algorithm where the objective function has been evaluated; we store the computed objective function values in the sequence $\{f_j\}$ while $\{f_j^R\}$ represent the sequence of reference values whose elements are given, at each $j$, by the maximum over a prefixed number $M$ of previous function values (see (2.1)).

In more detail, in Step 3 we compare the magnitude of $\|s_k\| + \|d_k\|$ with a decreasing sequence of upper bounds in order to validate that the iterates are converging. If this test fails then convergence is enforced by the nonmonotone linesearch procedure. The particular form of this control is motivated by the fact that whenever the sequence $\{x_k\}$ is superlinearly convergent towards a minimizer then $\|s_k\| \to 0$ and for every $\theta > 0$ and $\delta \in (0, 1)$, the inequality $\|d_k\| \leq \theta \delta^k$ hold for sufficiently large values of $k$ (see [13]). This implies that, eventually, the test at Step 3 is satisfied.

After a predetermined number $N$ of iterations in Step 4, the objective function is computed and compared with the actual reference value in order to evaluate if the new iterate leads to regions where the function is poorly behaved; in this case the algorithm is restarted from the iterate corresponding to the point $x_\ell$.

In Step 5 the nonmonotone linesearch procedure is based on the Armijo-type step size rule proposed in [15] with the distinguishing feature that in the right-hand side of (2.2), $f(x_k)$ is replaced by the reference value $f_j^R$ in order to still try to accept the unit step size. For further detail on the nonmonotone stabilization strategy we refer to paper [13].

Now we show that the properties of Algorithm NMC proved in [8] still hold under weaker requirements on the negative curvature direction $d_k$.

In particular first of all we assume that the directions $\{s_k\}$ and $\{d_k\}$ are bounded.

Then we introduce the following conditions:

**CONDITION 1.** The directions $\{s_k\}$ are such that $g(x_k)^T s_k \leq 0$ and

$$g(x_k)^T s_k \to 0 \implies g(x_k) \to 0 \quad \text{and} \quad s_k \to 0.$$
**Condition 2.** The directions \( \{ s_k \} \) are such that

\[
s_k \to 0 \quad \text{implies} \quad g(x_k) \to 0.
\]

**Condition 3.** The directions \( \{ d_k \} \) are such that

\[
g(x_k)^T d_k \leq 0, \quad d_k^T H(x_k) d_k \leq 0,
\]

\begin{equation}
\begin{cases}
  d_k^T H(x_k) d_k \to 0, \\
  g(x_k) \to 0
\end{cases}
\end{equation}

implies

\[
\begin{cases}
  \min \{ 0, \lambda_{\min}(H(x_k)) \} \to 0, \\
  d_k \to 0,
\end{cases}
\]

where \( \lambda_{\min}(H(x_k)) \) is the minimum eigenvalue of the Hessian matrix.

Condition 1 is the classical assumption on \( s_k \) used in literature to force the global convergence towards a stationary point. In Algorithm NMC (as in the algorithm of [13]) Condition 1 can be replaced with the weaker Condition 2 whenever a new point is accepted without using the linesearch procedure.

Condition 3 concerns the direction \( d_k \) and, as Theorem 2.1 will show, is able to ensure the convergence of the algorithm towards stationary points where the Hessian is positive semidefinite. This condition is weaker than those previously used in literature in order to avoid that an algorithm converges toward a point where the Hessian is not positive semidefinite. In fact (see [8, 15, 17, 24]) they require, at least, that (2.3) of Condition 3 be replaced by

\begin{equation}
d_k^T H(x_k) d_k \to 0 \quad \text{implies} \quad \begin{cases}
  \min \{ 0, \lambda_{\min}(H(x_k)) \} \to 0, \\
  d_k \to 0.
\end{cases}
\end{equation}

This difference could appear negligible, but as we said in the introduction, this new condition confirms the fact that the computation of a direction of negative curvature which is a good approximation of an eigenvector corresponding to the most negative eigenvalue is not strongly necessary when we are far from a stationary point.

The following theorem shows that the convergence properties of Algorithm NMC proved in [8] still hold under the weaker conditions stated above. The proof of this theorem is reported in the appendix.

**Theorem 2.1.** Let \( f \) be twice continuously differentiable, let \( x_0 \) be given, and suppose that the level set \( \Omega_0 = \{ x \in \mathbb{R}^n \mid f(x) \leq f(x_0) \} \) is compact. Assume that the directions \( s_k \) and \( d_k \) satisfy Conditions 2 and 3 and that, when linesearch is performed, the direction \( s_k \) satisfies also Condition 1. Let \( x_k, k = 0, 1, \ldots, \) be the points produced by Algorithm NMC. Then, either the algorithm terminates at some \( x_p \) such that \( g(x_p) = 0 \) and \( H(x_p) \) is positive semidefinite or it produces an infinite sequence such that

(a) the sequence \( \{ x_k \} \) remains in a compact set, and every limit point \( x_* \) belongs to \( \Omega_0 \) and satisfies \( g(x_*) = 0 \). Further, \( H(x_*) \) is positive semidefinite and no limit point of \( \{ x_k \} \) is a local maximum of \( f \);

(b) if the number of stationary points of \( f \) in \( \Omega_0 \) is finite, then the sequence \( \{ x_k \} \) converges to a stationary point where \( H \) is positive semidefinite;

(c) if there exists a limit point where \( H \) is nonsingular, the sequence \( \{ x_k \} \) converges to a local minimum point.

We note that, for simplicity, we stated the preceding theorem under the standard Condition 2; however, the same convergence results can be proved even if we replace Condition 2 with the following weaker condition:
CONDITION 2. The directions \( \{ s_k \} \) and \( \{ d_k \} \) are such that
\[
\| s_k \| + \| d_k \| \rightarrow 0 \quad \text{implies} \quad g(x_k) \rightarrow 0.
\]

Moreover, the proof can be easily modified in a way that the previous convergence results of Theorem 2.1 are ensured under Condition 2 (or Condition 2') and the following slight modifications of Condition 1 and Condition 3.

CONDITION 1'. The directions \( \{ s_k \} \) and \( \{ d_k \} \) are bounded and such that
\[
g(x_k)^T s_k \leq 0, \quad g(x_k)^T d_k \leq 0,
\]
\[
g(x_k)^T s_k \rightarrow 0 \quad \text{implies} \quad s_k \rightarrow 0,
\]
\[
g(x_k)^T (s_k + d_k) \rightarrow 0 \quad \text{implies} \quad g(x_k) \rightarrow 0.
\]

CONDITION 3'. The directions \( \{ d_k \} \) are bounded and such that
\[
g(x_k)^T d_k \leq 0, \quad d_k^T H(x_k) d_k \leq 0,
\]
\[
\begin{cases}
d_k^T H(x_k) d_k \rightarrow 0, \\
g(x_k)^T d_k \rightarrow 0
\end{cases}
\quad \text{implies} \quad \begin{cases}
\min [0, \lambda_{\min}(H(x_k))] \rightarrow 0, \\
d_k \rightarrow 0.
\end{cases}
\]

We remark that Condition 1' is weaker than Condition 1, while Condition 3' is stronger than Condition 3.

3. Computation of the search directions. In this section we consider the computation of the search directions \( s_k \) and \( d_k \) used by Algorithm NMC.

The direction \( s_k \) should play a double role; in fact, in addition to enforcing the global convergence of the algorithm, it should also guarantee, under standard assumptions, the superlinear convergence rate. As we are tackling large scale problems, this fact led us to compute the direction \( s_k \) by solving approximately the Newton equation
\[
(3.1) \quad H(x_k)s = -g(x_k).
\]

The most popular approach used to find an approximate solution of (3.1) which satisfies (or can be easily modified to satisfy) Conditions 1 and 2 is the use of the conjugate gradient method (see, for example, [1, 5, 6, 12, 23, 26]).

A related approach to solve approximately (3.1) is the use of the Lanczos algorithm; in fact, in [18, 19], Nash showed that a truncated scheme based on the Lanczos algorithm allows us to obtain an effective Newton-type direction.

As regards the direction \( d_k \), it should help the algorithm to escape from the region where the objective function is nonconvex. Furthermore, Condition 3 indicates that its contribution is more necessary as the gradient of the objective function is smaller in this region. Therefore, the direction \( d_k \) should be a nonascent direction of negative curvature such that it has a resemblance with an eigenvector of the Hessian matrix corresponding to the most negative eigenvalue, and this resemblance should be stronger as the gradient is smaller.

As indicated in [24] (see also [22]), a “good” approximation of the eigenvector corresponding to the smallest eigenvalue of the Hessian matrix can be efficiently obtained by using the Lanczos algorithm.
Therefore, on the basis of these considerations, a natural choice for us is that of using a truncated scheme based on the Lanczos algorithm for computing both search directions in our algorithm.

For the sake of completeness, now we describe the general scheme of the Lanczos algorithm with starting vector $q_1 = -g(x_k)$ (see, for example, [4, 10, 22]).

**Lanczos Algorithm.**

Data: $q_1 = -g(x_k)$

Step 1: $i = 1$, $v_0 = 0$, $\beta_i = \|q_i\|$.

Step 2: $v_i = \frac{1}{\beta_i} q_i$, $\alpha_i = v_i^T H(x_k) v_i$, $q_{i+1} = H(x_k) v_i - \alpha_i v_i - \beta_i v_{i-1}$, $\beta_{i+1} = \|q_{i+1}\|$

Step 3: If $\|\beta_{i+1}\| = 0$ stop; otherwise $i = i + 1$ and go to Step 2.

At the $i$th iteration the following matrices can be defined:

$$V_i = \begin{pmatrix} v_1 & v_2 & \vdots & v_i \end{pmatrix}, \quad T_i = \begin{pmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 \\ \beta_2 & \alpha_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & 0 & \ddots & \alpha_{i-1} & \beta_i \\ 0 & \cdots & 0 & \alpha_i & \beta_i \end{pmatrix}.$$

Now we recall some important results concerning the Lanczos algorithm which are summarized in the following theorem (see [4, 10, 22]).

**Theorem 3.1.**

(i) If $g(x_k)$ has nonzero projections only on $p$ eigenvectors of $H(x_k)$ and if the distinct eigenvalues corresponding to these $p$ eigenvectors are $m \leq p$, then the Lanczos algorithm terminates after $m$ steps with $\beta_{m+1} = 0$ and $\beta_i \neq 0$, $i = 1, \ldots, m$.

(ii) Suppose that $\tilde{y}_i$ is the solution of the system $T_i y = -V_i^T g(x_k)$, and set

$$s_i = V_i \tilde{y}_i.$$

Then, if $\beta_{i+1} = 0$, the vector $s_i$ is a solution of the system $H(x_k)s = -g(x_k)$; otherwise, if $\beta_{i+1} \neq 0$, the vector $s_i$ is the solution of the system $V_i^T (H(x_k)s + g(x_k)) = 0$.

(iii) Assume that $H(x_k)$ has $m$ distinct eigenvalues and that $g(x_k)$ has nonzero projections on $m$ eigenvectors of $H(x_k)$ which correspond to distinct eigenvalues. Let $\mu_m$ be the minimum eigenvalue of the tridiagonal matrix $T_m$ produced by the Lanczos algorithm, and let $w_m$ be the corresponding eigenvector. Then $\mu_m$ is also the minimum eigenvalue of $H(x_k)$, and its corresponding eigenvector is given by

$$d_m = V_m w_m.$$

(iv) If $\mu_i$ is the minimum eigenvalue of the tridiagonal matrix $T_i$, then there exists an eigenvalue $\lambda$ of $H(x_k)$ such that $\|\lambda - \mu_i\| \leq \beta_{i+1}$.

(v) Assume that $\lambda$ is the smallest eigenvalue of $H(x_k)$ and $\mu_i$ is the smallest eigenvalue of the tridiagonal matrix $T_i$. Then

$$\lambda \leq \mu_i \leq \lambda + \frac{C}{\phi_{i-1}^2},$$

where $C$ is a positive scalar and $\phi_{i-1}$ is the $i - 1$ degree Chebyshev polynomial.

Property (i) of Theorem 3.1 ensures that the algorithm terminates in a finite number of steps with $\beta_{m+1} = 0$. Property (ii) shows that the Lanczos algorithm can be successfully used for approximately solving the Newton equation (3.1).
Properties (iii), (iv), and (v) of Theorem 3.1 show that the Lanczos algorithm can be a useful tool for efficiently computing a good approximation of the most negative eigenvalue and the corresponding eigenvectors of the Hessian matrix $H(x_k)$. In particular, property (iii) shows that in the case of termination with $\beta_{m+1} = 0$ the minimum eigenvalue and the corresponding eigenvector of the Hessian matrix $H(x_k)$ can be obtained by computing the minimum eigenvalue and the corresponding eigenvector of the tridiagonal matrix $T_m$ and, as is well known, this last computation can be done in a very efficient way by taking the particular structure of the matrix $T_m$ into account. Property (iv) provides a simple bound on the accuracy of the estimate of the smallest eigenvalue of the Hessian matrix obtained at the $i$th iteration of Lanczos procedure. Finally, property (v) shows that the smallest eigenvalue of the tridiagonal matrix $T_i$ approaches the smallest eigenvalue of the Hessian matrix very quickly.

These considerations show that, at each iteration, the Lanczos algorithm produces two vectors $\tilde{s}_i$ and $\tilde{d}_i = V_i w_i$ which are approximations of a Newton-type direction and of an eigenvector corresponding to the smallest eigenvalue of the Hessian matrix $H(x_k)$, respectively. Furthermore, we can easily check the accuracy of these approximations. However, the use of (3.2) and (3.3) for the computation of $\tilde{s}_i$ and $\tilde{d}_i$ requires the storage of the matrix $V_i$, which limits their applications when $i$ is large. Fortunately, this drawback can be efficiently overcome in the computation of the vector $\tilde{s}_i$. In fact, in [21] it is shown that the properties of the Lanczos algorithm, the particular choice of the starting vector $q_1 = -g(x_k)$, and the particular structure of the tridiagonal matrix $T_i$ allow us to compute $\tilde{s}_i$ from the estimate $\tilde{s}_{i-1}$ obtained at the previous iteration by using update formulae which do not require any storage and matrix computations. In particular in our algorithm we use the SYMMLQ method proposed in [21], which is considered the most theoretically and numerically satisfactory method for solving linear systems involving indefinite symmetric matrices. Another feature of this solver of linear systems is that in the SYMMLQ algorithm, the Lanczos iterates are not interrupted even if the system is not consistent. This is particularly appealing in our approach since, also in this case, we can obtain useful information for computing a “good” direction of negative curvature $d_k$. In fact, in particular, we could compute the vector $\tilde{d}_i$ which is closely related to the eigenvector corresponding to the minimum eigenvalue of the Hessian matrix and use it as direction $d_k$. Unfortunately, the use of the matrix $V_i$ whose columns are the Lanczos vectors $v_1, \ldots, v_i$ cannot be avoided in the computation of $\tilde{d}_i$ given by (3.3). Therefore, due to the requirement of limited storage room, we store only a fixed number $L$ of Lanczos vectors in order to use a matrix $V_L$ of reasonable dimensions. Property (v) of Theorem 3.1 seems to indicate that few Lanczos vectors are needed to ensure that the direction $\tilde{d}_L$ has enough resemblance to the eigenvector corresponding to the minimum eigenvalue of the Hessian matrix.

Taking the previous considerations into account, we have based the computation of the directions $d_k$ and $s_k$ used in our algorithm NMC on the SYMMLQ routine by Paige and Saunders [21]. In particular we can roughly summarize the scheme that we have used for computing both search directions in the following steps:

- **Lanczos iterations.**
  The SYMMLQ routine has been slightly modified in a way that it continues to perform Lanczos iterations until the $i$th iteration, where both
  - one of the stopping criteria of the original version of SYMMLQ is satisfied,
  - and $|\beta_{i+1}| < \epsilon_k$ or $i > L$.
- **Computation of direction $s_k$.**
If the system (3.1) is solvable and the vector $\tilde{s}_i$ produced by SYMMLQ routine satisfies Conditions 1 and 2, then set $s_k = \tilde{s}_i$.

Otherwise, set $s_k = -g(x_k)$.

- Computation of direction $d_k$.

Compute the most negative eigenvalue $\mu_i$ of the tridiagonal matrix $T_i$;

- If $\mu_i < 0$, then compute the corresponding eigenvector $w_i$, and set

$$d_k = -\rho_k \text{sgn} \left[ g(x_k)^T \tilde{d}_i \right] \tilde{d}_i,$$

where $\tilde{d}_i = V_i w_i$ and $\rho_k$ is a suitable scaling factor.

- Otherwise, set $d_k = 0$.

Remark 3.1. We note that if enough storage room was available, we could set $L = n$, and hence, the previous scheme could compute a direction of negative curvature $d_k$ that ensures the convergence of the algorithm NMC toward stationary points where the Hessian matrix is positive semidefinite. More precisely, when enough storage is available and the assumption required at part (iii) of Theorem 3.1 is fulfilled, then Condition 3 is satisfied and Theorem 2.1 applies; otherwise, it is not ensured.

Remark 3.2. We point out that some of the choices concerning the search directions are dictated by the need of simplifying the analysis and to concentrate on the issues related to the main topic of the paper, i.e., the study of the possibility as well as the importance of exploiting the information on the local nonconvexity of the objective function contained in the second order derivatives. In particular, the use of the steepest descent direction as “alternative” choice of the direction $s_k$ is the simplest one. More efficient choices are certainly possible; for example, the global convergence can be ensured also by using more sophisticated directions obtained by minimizing the quadratic model of the objective function over the subspace spanned by the positive curvature directions produced by the Lanczos algorithm. As regards the negative curvature direction used in the algorithm, as already pointed out in the introduction, our choice has been motivated by its theoretical properties which guarantee a good resemblance to the eigenvector corresponding to the smallest eigenvalue of the Hessian matrix. Actually, other negative curvature directions can be obtained as a simple by-product of the Lanczos algorithm; these directions, even if very simple to compute, need further investigation to understand if they act sufficiently like the eigenvector corresponding to the smallest eigenvalue of the Hessian matrix. All these possibilities must be investigated and exploited in order to design an efficient code. This is beyond the aim of this paper and will be the subject of future work.

4. Numerical experiences. In this section we report the results of extensive numerical testing performed on all the unconstrained large scale problems available from the CUTE collection [2]. This test set consists of 98 functions covering all the classical test problems along with a large number of nonlinear optimization problems of various difficulty. The problems we have solved range from problems with 1000 variables to problems with 10,000 variables.

We made all the runs on an IBM RISC System/6000 375 using Fortran in double precision with the default optimization compiling option. For all the algorithms considered, the termination criterion is $\|g\| \leq 10^{-5}$. In the sequel, when we compare the results of different algorithms we consider all the test problems where the algorithms converge to the same point; moreover, in these comparisons the results of two runs are reputed equal if they differ by at most 5%.

The first purpose of these numerical experiences is to investigate the influence of
the presence of the negative curvature directions and the influence of the nonmonotone stabilization technique. To this aim, we have implemented four different algorithms belonging to the proposed class:

1. a monotone algorithm which does not use negative curvature directions (Mon);
2. a monotone algorithm which uses negative curvature directions (MonNC);
3. a nonmonotone algorithm which does not use negative curvature directions (NMon);
4. a nonmonotone algorithm which uses negative curvature directions (NMon-NC).

The second purpose is to have an idea of the potentiality of the approach described in this work in terms of efficiency and robustness. In particular, we have compared the results of our best algorithm with those obtained by the default band preconditioner version and the version without preconditioner of the LANCELOT package [3].

Of course, an efficient implementation of an algorithm belonging to the proposed class should be based on extensive empirical tuning of all the parameters which appear in the algorithm. Since the definition of an efficient code is not a goal of this paper, we have not adjusted all the parameters very accurately. In most of the cases, we have adopted (or slightly modified) choices already proposed in literature. The accurate analysis of the sensitivity of the algorithm as the parameters vary will be the subject of future work.

In more detail, as regards the stabilization scheme of the two nonmonotone algorithms, we adopt essentially the same parameters as in [13] with some modifications in order to take the fact that we are considering large scale problems into account; in particular, unlike [13], we use the following values: \( M = 20 \), \( N = 20 \), and \( \Delta_0 = 10^3 \). For the two monotone algorithms we set \( M = 0 \) and \( \Delta_0 = 0 \). The only difference with respect to the implementation described in [13] is that, following [28], we choose to reduce the freedom of the algorithm whenever a backtracking is performed; in fact, we set \( \Delta = 10^{-1} \Delta \) and \( M = M/5 + 1 \) whenever the algorithm backtracks to the last accepted point where the function has been evaluated.

In the scheme of the computation of the search directions we have set the user-specified tolerance required by the SYMMLQ routine to the value

\[
rtol = \tau \min \left\{ 1, \|g(x_k)\| \max \left( \frac{1}{k + 1}, \exp \left( \frac{1}{\tau n} \right) \right) \right\},
\]

where \( \tau = 10^{-1} \). The presence of the term \( 1/\exp \left( \frac{k}{\tau n} \right) \) reflects the fact that we are tackling large scale problems.

In the two algorithms which use negative curvature directions, we have set

\[
\epsilon_k = 10^3 \max \left\{ \|g(x_k)\|, \frac{1}{k + 1} \right\}
\]

and

\[
\rho_k = \sqrt{-\mu_i} \min \left\{ 1, \frac{1}{\|g_k\|} \right\},
\]

where \( \mu_i \) is the smallest eigenvalue of the matrix \( T_i \) produced by the Lanczos algorithm. The rationale behind the choice (4.1) is to force the parameter \( \epsilon_k \) to follow the behavior of the magnitude of the gradient only when \( k \) is large. The particular choice
(4.2) of the scaling factor $\rho_k$ derives from the one proposed by Moré and Sorensen in [17], where we added the term $\min\{1, 1/\|g_k\|\}$ since numerical experiences have indicated that it is better not to perturb the Newton-type direction too much when the iterates are far from a stationary point.

The complete results of our numerical experiences are reported in [14], and here, for sake of brevity, we report some summaries of this extensive numerical testing.

First we focused our attention more on the parameter $L$, which plays a crucial role for the applicability of our approach. We recall that this parameter represents the total number of Lanczos basis vectors stored; this number must be fixed to a small prescribed value in order to limit the storage requirements and, on the other hand, it influences the “goodness” of the negative curvature direction. Of course, this parameter appears only in the algorithms MonNC and NMonNC, which use negative curvature directions.

Since the effect in these two algorithms is quite similar, for the sake of simplicity, in Table 4.1 we report the cumulative results (that is the total number of iterations, function and gradient evaluations, and total CPU time, in seconds) obtained by algorithm NMonNC with different values $L$ on the whole test set of 98 test problems.

On the basis of these results, both the values $L = 50$ and $L = 100$ seem to be acceptable; in fact, for values $L \geq 50$ the influence of the negative curvature directions can be observed, but after this value an increase of CPU time is clearly shown without any substantial improvement of the efficiency of the algorithm. Hence, in what follows we have chosen as default value $L = 50$.

Another significant investigation concerns how expensive the computations of the most negative eigenvalue $\mu_i$ of the tridiagonal matrix $T_i$ and of its corresponding eigenvector $w_i$ are. In the algorithms NMonNC and MonNC, which require the computation of $\mu_i$ and $w_i$ in order to determine the directions of negative curvature $d_k$, we have computed “exactly” $\mu_i$ and $w_i$ by using a QR factorization of the tridiagonal matrix $T_i$ which exploits the particular structure of this matrix. The numerical results show that the computational burden for the algorithm NMonNC (similar situation is for MonNC) for determining the eigenvalue $\mu_i$ and the corresponding eigenvector $w_i$ is negligible with respect to the time needed to approximately solve the Newton equation. In fact, the cumulative CPU time needed for solving the Newton equation is 3649.19 seconds, while for computing $\mu_i$ and $w_i$ it is 17.13 seconds. We remark that in our framework the computation of $\mu_i$ and $w_i$ could be performed by some truncated scheme; however, on the basis of these results, this less expensive truncated computation seems not to be necessary at least for the problems considered.

Now, we report some summaries of the results obtained by the four algorithms Mon, MonNC, NMon, and NMonNC. In the comparisons of the performances of these algorithms reported in Tables 4.2 and 4.3 we have not considered the test problems BROYDEN7D and EIGENALS since the four algorithms converge to different points. In particular, in Table 4.2 we report how many times each of the four algorithms is the best, second, third, and worst, and in Table 4.3 we give the cumulative

### Table 4.1

Cumulative results for NMonNC with different values of $L$.

<table>
<thead>
<tr>
<th>$L$</th>
<th>Iterat.</th>
<th>Funct.</th>
<th>Grad.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6206</td>
<td>6378</td>
<td>6304</td>
<td>4934.05</td>
</tr>
<tr>
<td>50</td>
<td>5054</td>
<td>5369</td>
<td>5152</td>
<td>3846.99</td>
</tr>
<tr>
<td>100</td>
<td>4884</td>
<td>5243</td>
<td>4982</td>
<td>4067.2</td>
</tr>
<tr>
<td>200</td>
<td>4897</td>
<td>5488</td>
<td>4995</td>
<td>6131.46</td>
</tr>
<tr>
<td>500</td>
<td>4786</td>
<td>5303</td>
<td>4884</td>
<td>8319.56</td>
</tr>
</tbody>
</table>
Table 4.2
Comparative ranking for the algorithms Mon, MonNC, NMon, and NMonNC.

<table>
<thead>
<tr>
<th></th>
<th>Algorithm</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>Mon</td>
<td>78</td>
<td>0</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>Functions</td>
<td>MonNC</td>
<td>80</td>
<td>10</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Evaluations</td>
<td>NMon</td>
<td>77</td>
<td>2</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>Gradients</td>
<td>NMonNC</td>
<td>79</td>
<td>4</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>CPU time</td>
<td>Mon</td>
<td>50</td>
<td>13</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>MonNC</td>
<td>35</td>
<td>13</td>
<td>40</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4.3
Cumulative results for the algorithms Mon, MonNC, NMon, and NMonNC.

<table>
<thead>
<tr>
<th></th>
<th>Mon Monotone without neg. curv.</th>
<th>MonNC Monotone with neg. curv.</th>
<th>NMon Nonmonotone without neg. curv.</th>
<th>NMonNC Nonmonotone with neg. curv.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>5693</td>
<td>5077</td>
<td>5466</td>
<td>4793</td>
</tr>
<tr>
<td>Funct. eval.</td>
<td>8690</td>
<td>8163</td>
<td>5324</td>
<td>5091</td>
</tr>
<tr>
<td>Grad. eval.</td>
<td>5789</td>
<td>5173</td>
<td>5562</td>
<td>4889</td>
</tr>
<tr>
<td>CPU time</td>
<td>6444.09</td>
<td>6376.21</td>
<td>4540.92</td>
<td>3674.77</td>
</tr>
</tbody>
</table>

Table 4.4
The obtained objective function values of test problems BROYDN7D and EIGENALS.

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>MonNC</th>
<th>NMon</th>
<th>NMonNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROYDN7D</td>
<td>369.225</td>
<td>368.240</td>
<td>378.249</td>
<td>368.223</td>
</tr>
<tr>
<td>EIGENALS</td>
<td>485.00</td>
<td>.008738D-9</td>
<td>1327.00</td>
<td>.162078D-9</td>
</tr>
</tbody>
</table>

As can be easily seen from these tables, algorithms MonNC and NMonNC are the best in terms of number of iterations and gradient evaluations, while NMon and NMonNC are very clearly the best in terms of number of function evaluations and CPU time. This indicates that the use of negative curvature directions has a beneficial effect in terms of number of iterations and gradient evaluations. On the other hand, the effect of the nonmonotone strategy is especially evidenced as regards the number of function evaluations and CPU time.

Another indication of the importance of the negative curvature directions is suggested by the results obtained on the test problems EIGENALS and BROYDN7D, which we did not consider in the previous comparison. In fact, Table 4.4, where we report the best objective function value obtained by the four algorithms, shows that on these problems the use of negative curvature directions enables both algorithms MonNC and NMonNC to converge toward “better points,” i.e., points where the objective function value is lower.

In conclusion, the results of Tables 4.2, 4.3, and 4.4 indicate that the best algorithm is NMonNC since the simultaneous use of both the negative curvature
directions and the nonmonotone strategy allows a substantial improvement in terms of number of iterations, function and gradient evaluations, CPU time, and a tendency to converge towards “better” stationary points.

In the sequel we report the comparisons between the results of our best algorithm **NMonNC** and those obtained by the LANCELOT software package [3]; in particular, we have considered the default LANCELOT band preconditioner and the version without preconditioner. We excluded from these comparisons the test problem BROYDN7D, where the two LANCELOT algorithms converge to different points with respect to **NMonNC** algorithm.

In Table 4.5 we report the number of wins in terms of number of iterations, function and gradient evaluations, CPU time of the new algorithm **NMonNC** and the default LANCELOT band preconditioner on the test set, while in Table 4.6 we show the distribution of these wins (namely, how many times the percent gain of each algorithm is between 5% and 25%, 25% and 50%, 50% and 75%, 75% and 100%). Finally, in Table 4.7 we summarize the cumulative results obtained by these
algorithms. The reported results clearly show the good behavior of the new algorithm NMonNC and the relevant computational savings. It is important to notice, as the total CPU time required by our new algorithm NMonNC to solve all the problems is reduced more than 50%.

However, it can be pointed out that the great difference between the cumulative time required by the two algorithms is mainly due to the particular efficiency of our method on the two test problems FMINSURF 5625 and EIGENBLS, as is shown in Table 4.8. However, Algorithm NMonNC still outperforms LANCELOT even if we do not consider these problems in the computation of the cumulative results (see Table 4.9).

Now, in the tables that will follow, we report the results of the comparison between our algorithm NMonNC and LANCELOT without preconditioner. In particular, Table 4.10 summarizes the number of wins of these two algorithms on the test set, and Table 4.11 reports the distribution of their wins.

From Table 4.10 we can note that the version of LANCELOT without preconditioner appears efficient with regard to the number of gradient evaluations, but it is greatly beaten with regard to the wins in terms of CPU times and number of function evaluations. Table 4.11, besides confirming the superiority of NMonNC with regard to wins in terms of number of iterations, number of function evaluations, and CPU times, shows that most of the wins of LANCELOT without preconditioner in terms of number of gradient evaluations produce moderate gains. The cumulative results of our NMonNC and LANCELOT without preconditioner (see Table 4.12) show that NMonNC outperforms LANCELOT without preconditioner also in terms of gradient evaluation.

However, by observing the detailed results, it can be seen that the large number of gradient evaluations required by the version without preconditioner of LANCELOT is essentially due to the excessive number of gradient evaluations (1458 for NMonNC...
Table 4.11
Statistic of rankings of the wins for NMonNC and LANCELOT without preconditioner.

<table>
<thead>
<tr>
<th></th>
<th>5%-25%</th>
<th>25%-50%</th>
<th>50%-75%</th>
<th>75%-100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMonNC</td>
<td>20</td>
<td>13</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>LANCELOT without prec.</td>
<td>15</td>
<td>18</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4.12
Cumulative results for NMonNC and LANCELOT without preconditioner.

<table>
<thead>
<tr>
<th></th>
<th>NMonNC</th>
<th>LANCELOT without prec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>4964</td>
<td>8026</td>
</tr>
<tr>
<td>Function evaluations</td>
<td>5209</td>
<td>8123</td>
</tr>
<tr>
<td>Gradient evaluations</td>
<td>5061</td>
<td>31663</td>
</tr>
<tr>
<td>CPU time</td>
<td>3840.17</td>
<td>4652.86</td>
</tr>
</tbody>
</table>

...and 27791 for LANCELOT) needed for solving the test problem FLETCHCR 1000.

However, even if we do not consider this function, a significant computational savings is still noticeable (see Table 4.13).

On the basis of these results, it can be easily argued that our method outperforms the behavior of both the default band preconditioner algorithm and the version without preconditioner implemented in LANCELOT since a considerable reduction of the computational burden and an additional robustness are shown in both cases. This computational saving is more evident in terms of iterations, number of function evaluations, and CPU time.

5. Conclusions. In this paper we have considered a nonmonotone stabilization algorithmic model based on a curvilinear linesearch, which uses a combination of a Newton-type direction and a negative curvature direction. In particular we have shown that, under weak assumptions on the negative curvature direction, this algorithmic model produces a sequence of points converging toward stationary points which satisfy the second order necessary optimality condition. This new theoretical result gives useful hints on how to define new efficient large scale algorithms. In particular, we propose a class of algorithms which use the Lanczos method for computing, at each iteration, both a truncated Newton direction and a negative curvature direction.

The results obtained by a preliminary numerical experimentation indicate that the use of the nonmonotone strategy is always very effective both if it is used in an algorithm which does not use negative curvature directions and if it is used in an algorithm which uses negative curvature directions. With regard to the use of
negative curvature directions, our numerical experiences show that

- the use of negative curvature directions in a monotone algorithm leads to an improvement in terms of number of iterations and function and gradient evaluations but no significant difference in terms of CPU time;
- the benefits of the use of negative curvature directions are very evident in a nonmonotone context—in fact, in this case, good improvements in terms of CPU time can also be obtained.

The conclusion could be that the potentiality of negative curvature directions in a linesearch approach can be fully exploited by using nonmonotone strategies.

These computational results show also that our method outperforms the default band preconditioner version and the version without preconditioner of the LANCELOT package both in terms of computational burden and in terms of robustness. Even if no final conclusion can be drawn, this seems to indicate that the proposed approach is reliable and promising for defining new efficient algorithms for large scale minimization problems.

Another point which must be taken into account in comparing our approach with the one of the LANCELOT algorithms is the price that our algorithm must pay for using more sophisticated negative curvature directions. In fact, the computation of this direction requires the storage of an $n \times L$ matrix whose columns are the first $L$ Lanczos basis vectors produced by SYMMLQ routine. Even if relatively small values of the parameter $L$ are sufficient to ensure good behavior of our algorithm, the fact of requiring no matrix storage is an unquestionable advantage of the LANCELOT algorithms. Therefore, in conclusion, we think that new very efficient algorithms for large scale minimization problems could be defined by trying to join the advantages of our approach with those of the LANCELOT approach. In particular, a challenging research argument is the computation of good negative curvature directions which include enough second order information contained in the Hessian matrix without using matrix storage. As future work, we believe that promising approaches could be a smarter utilization of the information obtained during the conjugate gradient iterates and the use of a new efficient version of the Lanczos algorithm (see, for example, [25]).

Appendix. In this appendix we report some technical lemmas and the proof of Theorem 2.1. First of all we introduce some notations used in the following: we denote by $\{x_{k(j)}\}$ the sequence of points produced by Algorithm NMC where the objective function is evaluated, and we indicate with $\hat{f}_R^j$ the value of the reference value $f^*_R$ at the $k$th iteration of the algorithm.

**Lemma A.1 (see [8]).** Assume that Algorithm NMC produces an infinite sequence $\{x_k\}$; then

(a) the sequence $\{f^*_R\}$ is nonincreasing and has a limit $f^*_R$;
(b) for any index \( j \) we have
\[
f_i^R < f_j^R \quad \text{for all } \quad i > j + M;
\]
that is, the reference value \( f_i^R \) must decrease after at most \( M + 1 \) function evaluations;
(c) the sequence \( \{x_k\} \) remains in a compact set.

{
\begin{align*}
\text{Lemma A.2 (see [8]). Assume that Algorithm NMC produces an infinite sequence } & \{x_k\}. \text{ Then we can thin the sequence } \{x_{\ell(j)}\} \text{ so that it satisfies the following conditions:} \\
(\text{a}) & \quad f_j^R = f(x_{\ell(j)}) \text{ for } j = 0, 1, \ldots; \\
(\text{b}) & \quad \text{for any integer } k, \text{ there exists an index } j_k \text{ such that} \\
& \quad 0 < \ell(j_k) - k \leq N(M + 1), \quad f_{j_k}^R = f(x_{\ell(j_k)}) < f_k^R.
\end{align*}
\}

\text{Lemma A.3. Assume that Algorithm NMC produces an infinite sequence } \{x_k\} \text{ and that the assumptions of Theorem 2.1 hold. Then we have }
\begin{align*}
(\text{a}) & \quad \lim_{k \to \infty} f(x_k) = \lim_{k \to \infty} f_k^R = \lim_{j \to \infty} f_j^R = f_*^R, \\
(\text{b}) & \quad \lim_{k \to \infty} \alpha_k^2 \|s_k\| = 0, \lim_{k \to \infty} \alpha_k \|d_k\| = 0, \text{ implying that} \\
& \quad \lim_{k \to \infty} \|x_{k+1} - x_k\| = 0.
\end{align*}

\text{Proof. Note that in Step 3 (a) and Step 4 (b) of Algorithm NMC we may accept a step without performing a linesearch. Let } \{x_k\}_L \text{ denote the set of points where a linesearch is performed. Then}
\begin{equation}
\|s_k\| + \|d_k\| \leq \Delta_0 \delta^t \quad \text{for} \quad k \notin L,
\end{equation}
where the integer \( t \) increases with \( k \notin L \); when \( k \notin L \), we set \( \alpha_k = 1 \) for convenience.

It follows from (A.1) that if we do not perform a linesearch an infinite number of times, then
\begin{equation}
\lim_{k \to \infty} \alpha_k^2 \|s_k\| = 0 \quad \text{and} \quad \lim_{k \to \infty} \alpha_k \|d_k\| = 0.
\end{equation}

We show by induction that, for any fixed integer \( i \geq 1 \), we have
\begin{equation}
\lim_{j \to \infty} \alpha_{\ell(j)-i}^2 \|s_{\ell(j)-i}\| = 0 \quad \lim_{j \to \infty} \alpha_{\ell(j)-i} \|d_{\ell(j)-i}\| = 0,
\end{equation}
and
\begin{equation}
\lim_{j \to \infty} f(x_{\ell(j)-i}) = \lim_{j \to \infty} f(x_{\ell(j)}) = \lim_{j \to \infty} f_j^R = f_*^R.
\end{equation}

(Here and in what follows we assume that the index \( j \) is large enough to avoid the occurrence of negative subscripts.) Assume first that \( i = 1 \), and consider two subsequences of \( \{\ell(j) - 1\} \), corresponding to whether \( \ell(j) - 1 \) is in \( L \) or not. If either of these subsequences is finite, then we can discard the corresponding elements. Otherwise, for \( \ell(j) - 1 \notin L \), then (A.3) holds with \( i = 1 \). Now consider the other subsequence, where \( \ell(j) - 1 \in L \). Recalling the acceptability criterion of the nonmonotone linesearch, and by Lemma A.2 (a) we have
\[
f_j^R = f(x_{\ell(j)}) = f \left( x_{\ell(j)-1} + \alpha_{\ell(j)-1}^2 s_{\ell(j)-1} + \alpha_{\ell(j)-1} d_{\ell(j)-1} \right)
\]
\[
\leq f_{\ell(j)-1}^R + \gamma \alpha_{\ell(j)-1}^2 \left[ g(x_{\ell(j)-1})^T s_{\ell(j)-1} + \frac{1}{2} d_{\ell(j)-1}^T H(x_{\ell(j)-1}) d_{\ell(j)-1} \right].
\]
Therefore, if
\[ (A.5) \quad \tilde{f}^R_{\ell(j)-1} - f^R_{\ell(j)} \geq \gamma \alpha_{\ell(j)-1}^2 \left| g(x_{\ell(j)-1})^T s_{\ell(j)-1} + \frac{1}{2} d^T_{\ell(j)-1} H(x_{\ell(j)-1}) d_{\ell(j)-1} \right|. \]

It follows that
\[ (A.6) \quad \alpha_{\ell(j)-1}^2 g(x_{\ell(j)-1})^T s_{\ell(j)-1} \rightarrow 0 \quad \text{and} \quad \alpha_{\ell(j)-1}^2 d^T_{\ell(j)-1} H(x_{\ell(j)-1}) d_{\ell(j)-1} \rightarrow 0. \]

Now, (A.6) implies that either \( \alpha_{\ell(j)-1} \rightarrow 0 \) or
\[ \left\{ \begin{array}{l}
g(x_{\ell(j)-1})^T s_{\ell(j)-1} \rightarrow 0, \\
d^T_{\ell(j)-1} H(x_{\ell(j)-1}) d_{\ell(j)-1} \rightarrow 0. \end{array} \right. \]

In the first case we have \( \alpha_{\ell(j)-1}^2 \| s_{\ell(j)-1} \| \rightarrow 0 \) and \( \alpha_{\ell(j)-1} \| d_{\ell(j)-1} \| \rightarrow 0 \). In the second case, by Condition 1 we have \( \alpha_{\ell(j)-1}^2 \| s_{\ell(j)-1} \| \rightarrow 0 \) (taking \( \alpha_k \leq 1 \)). Condition 1 also implies \( g(x_{\ell(j)-1}) \rightarrow 0 \), and hence, by Condition 3, we obtain \( \alpha_{\ell(j)-1} \| d_{\ell(j)-1} \| \rightarrow 0 \).

It can be concluded that (A.3) holds for \( i = 1 \). Moreover, since
\[ f(x_{\ell(j)}) = f(x_{\ell(j)-1} + \alpha_{\ell(j)-1} s_{\ell(j)-1} + \alpha_{\ell(j)-1} d_{\ell(j)-1}), \]
(A.3) and the uniform continuity of \( f \) on the compact set containing \( \{ x_k \} \) imply that (A.4) also holds for \( i = 1 \).

Assume now that (A.3) and (A.4) hold for a given \( i \geq 1 \), and consider the point \( x_{\ell(j)-i-1} \). Reasoning as before, we can again distinguish between the case \( \ell(j) - i - 1 \notin \mathcal{L} \), when (A.1) holds with \( k = \ell(j) - i - 1 \), and the case \( \ell(j) - i - 1 \in \mathcal{L} \), in which we have
\[ \tilde{f}^R_{\ell(j)-i-1} - f(x_{\ell(j)-i}) \]
\[ (A.7) \geq \gamma \alpha_{\ell(j)-i-1}^2 \left| g(x_{\ell(j)-i-1})^T s_{\ell(j)-i-1} + \frac{1}{2} d^T_{\ell(j)-i-1} H(x_{\ell(j)-i-1}) d_{\ell(j)-i-1} \right|. \]

Using (A.2), (A.4), and (A.7) and recalling Conditions 1 and 3 we can assert that equations (A.3) hold with \( i \) replaced by \( i + 1 \). By (A.3) and the uniform continuity of \( f \), it follows that (A.4) is also satisfied with \( i \) replaced by \( i + 1 \), which completes the induction.

Now, let \( x_k \) be any given point produced by the algorithm. Then by Lemma A.2 there is an index \( j_k \) such that
\[ (A.8) \quad 0 < \ell(j_k) - k \leq N(M + 1). \]

Then we can write
\[ x_k = x_{\ell(j_k)} - \sum_{i=1}^{\ell(j_k) - k} \left[ \alpha_{\ell(j_k) - i}^2 s_{\ell(j_k) - i} + \alpha_{\ell(j_k) - i} d_{\ell(j_k) - i} \right], \]
and this implies, by (A.3) and (A.8), that
\[ \lim_{k \to \infty} \| x_k - x_{\ell(j_k)} \| = 0. \]
It follows from the uniform continuity of \( f \) that
\[
\lim_{k \to \infty} f(x_k) = \lim_{k \to \infty} f(x_{\ell(j_k)}) = \lim_{j \to \infty} f^R = \lim_{k \to \infty} f^R,
\]
and (a) is proved.

If \( k \in \mathcal{L} \), we obtain \( f(x_{k+1}) \leq f^R + \gamma \alpha_k^2 g(x_k)^T s_k + \frac{\gamma}{2} \alpha_k^2 d_k^T H(x_k) d_k \), and hence we have that
\[
\alpha_k^2 g(x_k)^T s_k + \frac{1}{2} \alpha_k^2 d_k^T H(x_k) d_k \to 0
\]
for \( k \to \infty, k \in \mathcal{L} \). Since both terms in this expression are nonpositive, it follows that
\[
\alpha_k^2 g(x_k)^T s_k \to 0 \text{ and } \alpha_k^2 d_k^T H(x_k) d_k \to 0.
\]
Then either \( \alpha_k \to 0 \) or
\[
\begin{cases}
g(x_k)^T s_k \to 0, \\
d_k^T H(x_k) d_k \to 0
\end{cases}
\]
for \( k \to \infty, k \in \mathcal{L} \). In the first case we have \( \alpha_k^2 \|s_k\| \to 0 \) and \( \alpha_k \|d_k\| \to 0 \). In the second case, by Condition 1 we have \( \alpha_k^2 \|s_k\| \to 0 \). Moreover, Condition 1 also implies that \( g(x_k) \to 0 \) for \( k \to \infty, k \in \mathcal{L} \), and hence, by using Condition 3, we obtain \( \alpha_k \|d_k\| \to 0 \) for \( k \to \infty, k \in \mathcal{L} \). Therefore, by (A.2) we can conclude that
\[
\lim_{k \to \infty} \alpha_k^2 \|s_k\| = 0 \text{ and } \lim_{k \to \infty} \alpha_k \|d_k\| = 0,
\]
which establishes (b).

**Proof of Theorem 2.1.** By Lemma A.1, the points \( x_k, k = 0, 1, \ldots, \) remain in a compact set. If the algorithm terminates, the assertion is obvious. Therefore, let \( x_* \) be any limit point of \( \{x_k\} \), and relabel \( \{x_k\} \) a subsequence converging to \( x_* \). By Lemma A.3, we have
\[
\lim_{k \to \infty} \alpha_k^2 \|s_k\| = 0 \text{ and } \lim_{k \to \infty} \alpha_k \|d_k\| = 0.
\]
Thus, either
\[
\lim_{k \to \infty} \|s_k\| = 0 \text{ and } \lim_{k \to \infty} \|d_k\| = 0
\]
or there exists a subsequence \( \{x_k\}_{K_1} \) of \( \{x_k\} \) such that \( \alpha_k \to 0 \) for \( k \to \infty, k \in K_1 \).

In the first case, taking into account the fact that \( s_k \) satisfies either Condition 1 or 2, we have \( \lim_{k \to \infty} \|g(x_k)\| = 0 \) and, by continuity, \( g(x_*) = 0 \); moreover, since \( \lim_{k \to \infty} d_k^T H(x_k) d_k = 0 \) by Condition 3 and by continuity
\[
0 = \lim_{k \to \infty} \min \{0, \lambda_{\min}(H(x_k))\} = \min \{0, \lambda_{\min}(H(x_*))\}.
\]
In the second case, the point $x_{k+1}$, $k \in K_1$, is produced by the nonmonotone linesearch procedure, and hence there exists an index $\hat{k}$ such that, for all $k \geq \hat{k}$, $k \in K_1$,

$$f\left(x_k + \left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right) > f_k^R + \gamma \left(\frac{\alpha_k}{\sigma}\right)^2 \left[g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k\right]$$

$$\geq f(x_k) + \gamma \left(\frac{\alpha_k}{\sigma}\right)^2 \left[g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k\right]$$

for some $\sigma \in (0, 1)$. By the mean value theorem, we can find, for any $k \geq \hat{k}$, $k \in K_1$, a point $u_k = x_k + \omega_k \left((\alpha_k/\sigma)^2 s_k + (\alpha_k/\sigma) d_k\right)$ with $\omega_k \in (0, 1)$ such that, recalling that $g(x_k)^T d_k \leq 0$ we have

$$f\left(x_k + \left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right) = f(x_k) + g(x_k)^T \left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k$$

$$+ \frac{1}{2} \left[\left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right]^T H(u_k) \left[\left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right]$$

$$\leq f(x_k) + \left(\frac{\alpha_k}{\sigma}\right)^2 g(x_k)^T s_k + \frac{1}{2} \left[\left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right]^T H(u_k) \left[\left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right].$$

It follows that

$$f(x_k) + \gamma \left(\frac{\alpha_k}{\sigma}\right)^2 \left[g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k\right]$$

$$\leq f(x_k) + \left(\frac{\alpha_k}{\sigma}\right)^2 g(x_k)^T s_k + \frac{1}{2} \left[\left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right]^T H(u_k) \left[\left(\frac{\alpha_k}{\sigma}\right)^2 s_k + \frac{\alpha_k}{\sigma} d_k\right].$$

Dividing both sides by $(\alpha_k/\sigma)^2$ and by simple manipulations we obtain

$$(\gamma - 1) \left[g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k\right]$$

$$\leq \frac{1}{2} d_k^T \left[H(u_k) - H(x_k)\right] d_k + \frac{1}{2} \left(\frac{\alpha_k}{\sigma}\right)^2 s_k^T H(u_k) s_k + \frac{\alpha_k}{\sigma} d_k^T H(u_k) d_k,$$

where $\gamma < 1/2$. Now let $\{x_k\}_{K_2} \subset \{x_k\}_{K_1}$ be a subsequence such that

$$\lim_{k \to \infty, k \in K_2} x_k = x_*, \quad \lim_{k \to \infty, k \in K_2} s_k = s_*, \quad \lim_{k \to \infty, k \in K_2} d_k = d_*.$$

From $\alpha_k \to 0$ for $k \to \infty$, $k \in K_1$, we have $u_k \to x_*$ as $k \to \infty$, $k \in K_2$. Since $\gamma - 1 < 0$, $g(x_k)^T s_k \leq 0$ and $d_k^T H(x_k) d_k \leq 0$ for all $k \in K_2$, then by (A.11) and (A.12) we have

$$\lim_{k \to \infty, k \in K_2} (\gamma - 1) \left[g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k\right] = 0$$

and hence

$$g(x_*)^T s_* = 0 \quad \text{and} \quad d_*^T H(x_*) d_* = 0.$$

Condition 1 now implies that $g(x_*) = 0$, and Condition 3 gives that $H(x_*)$ is positive semidefinite. Moreover, by Lemma A.1 and Lemma A.3, we have that $x_* \in \Omega_0$. The
proof of assertion (a) can be easily completed similarly to the proof of Theorem 1 of [13].

Assertion (b) follows from Theorem 14.1.5 of [20] and by using Lemma A.3(b). Finally, assertion (c) follows from known results [20, p. 478], taking into account Lemma A.3(b).

Acknowledgments. We thank M. A. Saunders for providing us with the code of the SYMMLQ routine and J. K. Cullum for the complete code of Lanczos algorithms. Moreover, we are indebted to N. I. M. Gould and Ph. L. Toint for helpful discussions and for their constructive comments. Finally, we would like to thank very much the anonymous referee and A. Sartenaer (the second referee, who agreed to reveal her identity for this acknowledgment) for their useful suggestions and remarks which led to significantly improving the paper.

REFERENCES


