# Nonmonotone Curvilinear Line Search Methods for Unconstrained Optimization

M.C. Ferris*   S. Lucidi†   M. Roma†

March 20, 1995

**Abstract**

We present a new algorithmic framework for solving unconstrained minimization problems that incorporates a curvilinear linesearch. The search direction used in our framework is a combination of an approximate Newton direction and a direction of negative curvature. Global convergence to a stationary point where the Hessian matrix is positive semidefinite is exhibited for this class of algorithms by means of a nonmonotone stabilization strategy. An implementation using the Bunch-Parlett decomposition is shown to outperform several other techniques on a large class of test problems.

## 1 Introduction

In this work we consider the unconstrained minimization problem

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

where $f$ is a real valued function on $\mathbb{R}^n$. 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 iterative methods for solving this problem have been proposed; they are usually descent methods that generate a sequence $\{x_k\}$ such that every limit point $x_*$ is stationary, i.e. $g(x_*) = 0$ or the weaker condition $\liminf_{k \to \infty} \|g(x_k)\| = 0$. The condition $g(x_*) = 0$ is only a necessary first order optimality condition so it does not guarantee that $x_*$ is a local minimum point. In order to increase the chances of obtaining a local minimum point, we try to detect only the stationary points that satisfy the second order necessary conditions. The key idea is to define algorithms that converge to a stationary point $x_*$ where the Hessian matrix $H(x_*)$ is positive semidefinite.

Algorithms with this property have been defined both in a trust region and a linesearch context. In fact, guaranteeing convergence only to stationary points that satisfy second order conditions depends heavily on the skill of the minimization method in escaping regions where the objective function is not convex. Trust region algorithms intrinsically have this capability due to the fact that they are based on the idea of minimizing the quadratic model of the objective function over a sphere (see, for example [23]). Linesearch algorithms do not enjoy this particular feature. In

*Computer Sciences Department, University of Wisconsin, Madison, Wisconsin 53706. The work of this author was based on research supported by the National Science Foundation grant CCR-9157632, the Air Force Office of Scientific Research grant F49620-94-1-0036 and the Department of Energy grant DE-FG03-94ER61915.

†Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Roma, Italy. These authors were partially supported by Agenzia Spaziale Italiana, Roma, Italy.

fact, if linesearch algorithms are to produce a sequence of points converging towards stationary points where the Hessian matrix is positive semidefinite, they must extract as much information as possible about the curvature of objective function from the second order derivatives.

In this context some interesting results have been proposed in [15] and [17]. In these papers, linesearch algorithms that generate sequences $\{x_k\}$ which converge to points $x_*$ where the Hessian $H(x_*)$ is positive semidefinite are defined. A key feature of these algorithms is the use of particular directions of negative curvature, i.e. directions $d$ such that $d^T H(x)d < 0$, along with a generalized monotonic Armijo step size rule.

Unfortunately, after these two papers, little work has followed this approach. Most globally convergent linesearch algorithms proposed in literature are based on the idea of perturbing the Hessian matrix without directly exploiting curvature information of the objective function. This could be due to the fact that the classical test problems (with standard starting points) do not allow a real evaluation of algorithms which use negative curvature directions. This is because these test problems present just a few points where the objective function has negative curvature, as was noted in [16, 17]. The advent of the CUTE collection of test problems [2] allows us to obtain a good evaluation of globally convergent algorithms, since it includes a variety of functions with negative curvature.

The main aim of our work is to encourage the development of new research in this area. In particular, we believe that further investigation is needed to define efficient new linesearch algorithms that exploit the curvature information of the objective function more deeply.

As a first step we show that the linesearch approach proposed in [17] can be embedded in a general nonmonotone globalization strategy based on the model of [14]. This allows us to define a general algorithm model that inherits the following properties:

- the capability to efficiently exploit potential local non convexity of the objective function (from [17])
- the capability to efficiently solve highly nonlinear and ill-conditioned minimization problems (from the nonmonotone approach of [14]).

As a second step we test the new algorithm against two Newton-type algorithms. The first of these follows the algorithm model proposed in [14] which uses a classical perturbation of the Newton direction (based on the modified Cholesky factorization [12]) as the search direction. Since the new algorithm uses essentially the same stabilization technique, the comparison between these two determines the effect of exploiting information on the curvature of the objective function via the curvilinear linesearch. The second algorithm we compare against is LANCELOT [4], which is a trust region based code. The numerical results which we report show that a significant improvement in terms of the number of iterations and function-gradient evaluations can be obtained by using an algorithm that exploits curvature effects.

Section 2 describes the general framework of our stabilization scheme and introduces conditions that are required on the search directions for convergence. Section 3 proves that such an algorithmic framework leads to a convergent algorithm. In Section 4 we describe a particular implementation and show that this example algorithm satisfies the assumptions of Section 2. The implementation is tested on a large set of standard test problems in Section 5 and is shown to have more efficient and reliable convergence than standard Newton-type approaches.

## 2 Nonmonotone stabilization algorithm

We consider an iterative scheme of the form

$$x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k,$$

where $s_k$ and $d_k$ are search directions and $\alpha_k$ is a step size. We assume throughout that, for a given $x_0 \in \mathbb{R}^n$, the level set

$$\Omega_0 = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\},$$

is compact. In practice, we think of $s_k$ as an approximation of the Newton step and $d_k$ as a direction of negative curvature. Both of these can be calculated using the Bunch–Parlett factorization [3]; this was first considered in [17]. In order that our convergence theorems remain applicable to more general algorithms than the ones we test numerically, we list below the general conditions that we require of the search directions. We assume that the directions $\{s_k\}$ and $\{d_k\}$ satisfy the following conditions:

**Condition 1.** *The direction $\{s_k\}$ and $\{d_k\}$ are bounded and satisfy*

$$g(x_k)^T s_k \leq 0 \qquad g(x_k)^T d_k \leq 0$$

$$g(x_k)^T s_k \to 0 \quad \textit{implies} \quad s_k \to 0.$$

$$g(x_k)^T (s_k + d_k) \to 0 \quad \textit{implies} \quad g(x_k) \to 0.$$

**Condition 2.** *The directions $\{s_k\}$ and $\{d_k\}$ have the property that*

$$\|s_k\| + \|d_k\| \to 0 \quad \textit{implies} \quad g(x_k) \to 0.$$

**Condition 3.** *The direction $\{d_k\}$ satisfies*

$$d_k^T H(x_k) d_k \leq 0,$$

$$d_k^T H(x_k) d_k \to 0 \quad \textit{implies} \quad \min[0, \lambda_m(H(x_k))] \to 0 \quad \textit{and} \quad d_k \to 0,$$

*where $\lambda_m(H)$ is the minimum eigenvalue of $H$.*

In Section 4, we show that a particular implementation satisfies these conditions.

The key difference between our approach and that of [17] is in the line search acceptance criterion. We allow possible increases to the sequence of function values by keeping track of a reference value (which we label $F$) in order to relax the normal Armijo acceptance criterion. Typically, $F$ would be set to the maximum function value over the last few iterations as opposed to $f(x_k)$ in the standard Armijo procedure. The **nonmonotone line search** procedure that we use can be stated precisely, as follows.

## Nonmonotone line search procedure

*Data:* $x_k$, $s_k$, $d_k$, $g(x_k)$, $H(x_k)$, $F$, $\gamma \in (0, \frac{1}{2})$, $0 < \sigma_1 < \sigma_2 < 1$.

*Step 1:* If necessary, modify $s_k$ and $d_k$ to satisfy Condition 1 and Condition 3 and set $\alpha = 1$.

*Step 2:* If $f(x_k + \alpha^2 s_k + \alpha d_k) \leq F + \gamma\alpha^2 \left[g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k\right]$ set $\alpha_k = \alpha$ and stop.

*Step 3:* Choose $\sigma \in [\sigma_1, \sigma_2]$, $\alpha = \sigma\alpha$ and go to Step 2.

This approach has been tested in [13] and proven successful on many ill-conditioned problems. However, it does enforce a monotonic decrease in the sequence $\{F_j\}$. We wish to allow even greater freedom for the function values, whereby we accept the "Newton–like" step without even checking the function value at the new point provided the length of the step is not too large. This is motivated by the fact that eventually the step length will decrease to zero when we converge to a solution and by the fact that always enforcing the point $x_k$ to be located in nested level sets may deteriorate the efficiency of the Newton–like method (see the discussion in Section 5.1 of [14]). Computational results [14] have shown this technique to be even better than the standard nonmonotone line search. This technique has also proven useful in other applications and extensions [6, 7, 8, 25, 26, 27]. Of course, sometimes it may lead to regions where the function is poorly behaved. In these cases, a backtracking scheme is incorporated into our algorithm. In effect, we backtrack to the last point where the function was evaluated (which we denote by $x_\ell$) and perform a nonmonotone line search from that point. The avoidance of the evaluation of $f$ is not the key point here; rather we believe that robustness is increased by considering other factors crucial to the convergence.

The full details of our nonmonotone stabilization scheme are detailed below. Note that $\ell$ denotes the index of the last accepted point where the objective function was evaluated.

## Nonmonotone Stabilization Algorithm (NMS)

*Data:* $x_0$, $\Delta_0 > 0$, $\beta \in (0, 1)$, $N \geq 1$ and $M \geq 0$.

*Step 1:* Set $k = \ell = j = 0$, $\Delta = \Delta_0$. Compute $f(x_0)$ and set $Z_0 = F_0 = f(x_0)$, $m(0) = 0$.

*Step 2:* Compute $g(x_k)$. If $\|g(x_k)\| = 0$ stop.

*Step 3:* If $k = \ell + N$ compute $f(x_k)$; then:

(a) if $f(x_k) \geq F_j$, replace $x_k$ by $x_\ell$, set $k = \ell$ and go to Step 5;

(b) if $f(x_k) < F_j$, set $\ell = k$, $j = j + 1$, $Z_j = f(x_k)$ and and update $F_j$ according to

$$F_j = \max_{0 \leq i \leq m(j)} Z_{j-i}, \qquad \text{where} \qquad m(j) \leq \min[m(j-1) + 1, M]; \tag{1}$$

compute directions $s_k$ and $d_k$ that satisfy Conditions 2 and 3;
if $\|s_k\| + \|d_k\| \leq \Delta$, set $x_{k+1} = x_k + s_k + d_k$, $k = k + 1$, $\Delta = \beta\Delta$ and go to Step 2;
otherwise go to Step 5.

*Step 4:* If $k \neq \ell + N$ compute directions $s_k$ and $d_k$ that satisfy Conditions 2 and 3; then:

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

(b) if $\|s_k\| + \|d_k\| > \Delta$, compute $f(x_k)$;
if $f(x_k) \geq F_j$, replace $x_k$ by $x_\ell$, set $k = \ell$;
otherwise set $\ell = k$, $j = j + 1$, $Z_j = f(x_k)$ and update $F_j$ according to (1).

*Step 5:* Compute $\alpha_k$ by means of a **nonmonotone line search**, set

$$x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k, \quad k = k+1, \quad \ell = k,$$

update $F_j$ according to (1) and go to Step 2.

Note that in practice the most frequently taken step is Step 4(a). In this case, we do not have to satisfy Condition 1, so that $s_k$ and $d_k$ may not even be descent directions for $f$. Furthermore, note that if a nonmonotone line search is performed at Step 5, then $s_k$ and $d_k$ may have to be modified to satisfy Condition 1, and hence may no longer satisfy Condition 2. However, at each iteration, $s_k$ and $d_k$ must satisfy at least one of Conditions 1 or 2.

For later reference let $\{x_{\ell(j)}\}$ be the sequence of points where $f$ was evaluated and let $\{F_j\}$ be the corresponding sequence of reference values. We initially set $j = 0$ and increment $j$ each time we define $\ell = k$.

As regards the reference value $F_j$ for the objective function, it is initially set to $f(x_0)$ and is updated whenever the function $f$ is evaluated and the corresponding point is accepted. More specifically, the updating takes into account a prefixed number $m(j) \leq M$ of previous function values, which is called the "memory".

**Remark** Notice that if we set $M = 0$ and $\Delta_0 = 0$, we obtained the same algorithm proposed in [17]. Moreover, it is important to note that we obtain the convergence results under weaker conditions than those ones required in [17]. This small difference allows us to use a particular pair of directions that do not satisfy the assumptions of [17], but do satisfy our conditions.

# 3 Convergence analysis

To establish the convergence properties of Algorithm NMS, we employ some technical lemmas. The first of these establishes three important facts regarding the reference values and the iterates of the algorithm. The proof of the first two lemmas easily follows, with minor modifications, from the proof of Lemma 1 and Lemma 2 in [14].

**Lemma 3.1** *Assume that Algorithm NMS produces an infinite sequence $\{x_k\}$; then:*

(a) *the sequence $\{F_j\}$ is non increasing and has a limit $F_*$;*

(b) *for any index $j$ we have*

$$F_i < F_j, \quad \text{for all} \quad i > j + M,$$

*that is, the reference value must decrease after at most $M+1$ function evaluations;*

(c) *$\{x_k\}$ remains in a compact set.*

However, note that the iterates need not remain in $\Omega_0$.

**Lemma 3.2** *Assume that Algorithm NMS produces an infinite sequence $\{x_k\}$; let $\{x_{\ell(j)}\}$ be the sequence of points where the objective function is evaluated.*

*Then, we can thin the sequence $\{x_{\ell(j)}\}$ so that it satisfies the following conditions:*

(a) *$F_j = f(x_{\ell(j)})$, for $j = 0, 1, \ldots$;*

(b) *for any integer $k$, there exists an index $j_k$ such that:*

$$0 < \ell(j_k) - k \le N(M+1), \qquad F_{j_k} = f(x_{\ell(j_k)}) < \hat{F}_k,$$

*where $\hat{F}_k$ is the value of $F_j$ at the $k$th iteration of Algorithm NMS.*

Note that $\lim_{k\to\infty} \hat{F}_k = \lim_{j\to\infty} F_j$ since $\{\hat{F}_k\}$ just "fills in" with values of $F_j$.

The following lemma is key to our development and shows that the function values on the whole sequence converge. In the standard Armijo case, this is easy to establish. In the nonmonotone case, the proof is somewhat more involved.

**Lemma 3.3** *Assume that Algorithm NMS produces an infinite sequence $\{x_k\}$.*

*Then, we have:*

(a) $$\lim_{k\to\infty} f(x_k) = \lim_{k\to\infty} \hat{F}_k = \lim_{j\to\infty} F_j = F_*,$$

(b) $$\lim_{k\to\infty} \alpha_k^2 \|s_k\| = 0, \lim_{k\to\infty} \alpha_k \|d_k\| = 0, \text{ implying that } \lim_{k\to\infty} \|x_{k+1} - x_k\| = 0.$$

**Proof** Note that in Step 3(b) and Step 4(a) of Algorithm NMS we may accept a step without performing a line search. Let $\{x_k\}_{\mathcal{L}}$ denote the set of points where a line search is performed. Then

$$\|s_k\| + \|d_k\| \le \Delta_0 \beta^t, \quad \text{for} \quad k \notin \mathcal{L}, \tag{2}$$

where the integer $t$ increases with $k \notin \mathcal{L}$; when $k \notin \mathcal{L}$ we set, for convenience, $\alpha_k = 1$. It follows from (2) that if we do not perform a line search an infinite number of times, then

$$\lim_{\substack{k\to\infty \\ k\notin\mathcal{L}}} \alpha_k^2 \|s_k\| = 0, \quad \text{and} \quad \lim_{\substack{k\to\infty \\ k\notin\mathcal{L}}} \alpha_k \|d_k\| = 0. \tag{3}$$

Let $\ell(j)$ be the indices defined in Lemma 3.2. We show by induction that, for any fixed integer $i \ge 1$, we have:

$$\lim_{j\to\infty} \alpha_{\ell(j)-i}^2 \left\| s_{\ell(j)-i} \right\| = 0, \qquad \lim_{j\to\infty} \alpha_{\ell(j)-i} \left\| d_{\ell(j)-i} \right\| = 0, \tag{4}$$

and

$$\lim_{j\to\infty} f(x_{\ell(j)-i}) = \lim_{j\to\infty} f(x_{\ell(j)}) = \lim_{j\to\infty} F_j = F_*. \tag{5}$$

(Here and in the sequel 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 $\mathcal{L}$ or not. If either of these subsequences are finite, then we can discard the corresponding elements. Otherwise, for $\ell(j) - 1 \notin \mathcal{L}$, then (4) holds with $i = 1$. Now consider the other subsequence, where $\ell(j) - 1 \in \mathcal{L}$. Recalling the acceptability criterion of the nonmonotone line search, we have

$$\hat{F}_{\ell(j)-1} - F_j \ge \gamma \alpha_{\ell(j)-1}^2 \left| g(x_{\ell(j)-1})^T s_{\ell(j)-1} + \frac{1}{2} d_{\ell(j)-1}^T H(x_{\ell(j)-1}) d_{\ell(j)-1} \right|. \tag{6}$$

Therefore, if $\ell(j) - 1 \in \mathcal{L}$ for an infinite subsequence, from Lemma 3.1(a) and (6) and the observations that $g(x_{\ell(j)-1})^T s_{\ell(j)-1} < 0$ and $d_{\ell(j)-1}^T H(x_{\ell(j)-1}) d_{\ell(j)-1} \le 0$, we get

$$\alpha_{\ell(j)-1}^2 g(x_{\ell(j)-1})^T s_{\ell(j)-1} \to 0, \quad \text{and} \quad \alpha_{\ell(j)-1}^2 d_{\ell(j)-1}^T H(x_{\ell(j)-1}) d_{\ell(j)-1} \to 0. \tag{7}$$

Now, (7) implies that either $\alpha_{\ell(j)-1} \to 0$ or $g(x_{\ell(j)-1})^T s_{\ell(j)-1} \to 0$, $d^T_{\ell(j)-1} H(x_{\ell(j)-1}) d_{\ell(j)-1} \to 0$. In the first case we have $\alpha^2_{\ell(j)-1} \|s_{\ell(j)-1}\| \to 0$ and $\alpha_{\ell(j)-1} \left\| d_{\ell(j)-1} \right\| \to 0$. In the second case, by Condition 1 we have $\alpha^2_{\ell(j)-1} \|s_{\ell(j)-1}\| \to 0$ (taking $\alpha_k \leq 1$) and by Condition 3, $\alpha_{\ell(j)-1} \left\| d_{\ell(j)-1} \right\| \to 0$. It can be concluded that (4) holds for $i = 1$. Moreover since

$$f(x_{\ell(j)}) = f(x_{\ell(j)-1} + \alpha^2_{\ell(j)-1} s_{\ell(j)-1} + \alpha_{\ell(j)-1} d_{\ell(j)-1}),$$

(4) and the uniform continuity of $f$ on the compact set containing $\{x_k\}$ imply that equation (5) also holds for $i = 1$.

Assume now that (4) and (5) hold for a given $i$ and consider the point $x_{\ell(j)-i-1}$. Reasoning as before, we can again distinguish the case $\ell(j) - i - 1 \notin \mathcal{L}$, when (2) holds with $k = \ell(j) - i - 1$, and the case $\ell(j) - i - 1 \in \mathcal{L}$, where we have:

$$\hat{F}_{\ell(j)-i-1} - f(x_{\ell(j)-i}) \geq \gamma \alpha^2_{\ell(j)-i-1} \left| g(x_{\ell(j)-i-1})^T s_{\ell(j)-i-1} + \frac{1}{2} d^T_{\ell(j)-i-1} H(x_{\ell(j)-i-1}) d_{\ell(j)-i-1} \right|. \quad (8)$$

Using (3), (5), (8) and recalling Conditions 1 and 3, we can assert that equations (4) hold with $i$ replaced by $i + 1$. By (4) and the uniform continuity of $f$, it follows that (5) 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 3.2 there is an index $j_k$ such that

$$0 < \ell(j_k) - k \leq (M + 1)N. \quad (9)$$

Then we can write:

$$x_k = x_{\ell(j_k)} - \sum_{i=1}^{\ell(j_k)-k} \left[ \alpha^2_{\ell(j_k)-i} s_{\ell(j_k)-i} + \alpha_{\ell(j_k)-i} d_{\ell(j_k)-i} \right],$$

and this implies, by (4) and (9), that:

$$\lim_{k\to\infty} \left\| x_k - x_{\ell(j_k)} \right\| = 0.$$

It follows from the uniform continuity of $f$ that

$$\lim_{k\to\infty} f(x_k) = \lim_{k\to\infty} f(x_{\ell(j_k)}) = \lim_{j\to\infty} F_j = \lim_{k\to\infty} \hat{F}_k, \quad (10)$$

and (a) is proved.

If $k \in \mathcal{L}$, we obtain $f(x_{k+1}) \leq \hat{F}_k + \gamma \alpha_k^2 g(x_k)^T s_k + \frac{\gamma}{2} \alpha_k^2 d_k^T H(x_k) d_k$ and hence we have that:

$$\hat{F}_k - f(x_{k+1}) \geq \gamma \alpha_k^2 \left| g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k \right|. \quad (11)$$

By (10) and (11)

$$\alpha_k^2 g(x_k)^T s_k + \frac{1}{2} \alpha_k^2 d_k^T H(x_k) d_k \to 0,$$

for $k \to \infty, k \in \mathcal{L}$. Since both terms in this expression are non positive, it follows that

$$\alpha_k^2 g(x_k)^T s_k \to 0, \qquad \alpha_k^2 d_k^T H(x_k) d_k \to 0.$$

Now by using Condition 1 we have $\alpha_k^2 \|s_k\| \to 0$ for $k \to \infty, k \in \mathcal{L}$. By using Condition 3 we obtain $\alpha_k \|d_k\| \to 0$ for $k \to \infty, k \in \mathcal{L}$. Therefore by (3) we can conclude that:

$$\lim_{k\to\infty} \alpha_k^2 \|s_k\| = 0, \qquad \lim_{k\to\infty} \alpha_k \|d_k\| = 0,$$

which establishes (b). $\square$

Now we can prove our main theorem.

**Theorem 3.4** *Let $f$ be twice continuously differentiable, $x_0$ be given and suppose that the level set $\Omega_0$ at $x_0$ is compact. Let $x_k$, $k = 0, 1, \dots$ be the points produced by Algorithm NMS. 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) *$\{x_k\}$ 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 there exists a limit point where $H$ is non-singular, the sequence $\{x_k\}$ converges to a local minimum point.*

**Proof** By Lemma 3.1, the points $x_k$, $k = 0, 1, \dots$ 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 3.3, we have

$$\lim_{k\to\infty} \alpha_k^2 \|s_k\| = 0 \text{ and } \lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{12}$$

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 continuity

$$0 = \lim_{k\to\infty} \min\left[0, \lambda_m\left(H(x_k)\right)\right] = \min\left[0, \lambda_m\left(H(x_*)\right)\right].$$

In the second case, the point $x_{k+1}$, $k \in K_1$, is produced by the nonmonotone line search procedure, and hence there exists an index $\hat{k}$ such that, for all $k \geq \hat{k}$, $k \in K_1$:

$$f\left(x_k + \left(\frac{\alpha_k}{\sigma_k}\right)^2 s_k + \frac{\alpha_k}{\sigma_k} d_k\right) > f(x_k) + \gamma \left(\frac{\alpha_k}{\sigma_k}\right)^2 \left[g(x_k)^T s_k + \frac{1}{2} d_k^T H(x_k) d_k\right], \tag{13}$$

for some $\sigma_k \in [\sigma_1, \sigma_2]$. By the Mean Value Theorem, we can find, for any $k \geq \hat{k}$, $k \in K_1$, a point $u_k = x_k + \omega_k \left((\alpha_k/\sigma_k)^2 s_k + (\alpha_k/\sigma_k) d_k\right)$ with $\omega_k \in (0,1)$ such that:

$$\begin{aligned} & f\left(x_k + \left(\frac{\alpha_k}{\sigma_k}\right)^2 s_k + \frac{\alpha_k}{\sigma_k} d_k\right) \\ \leq\ & f(x_k) + \left(\frac{\alpha_k}{\sigma_k}\right)^2 g(x_k)^T (s_k + d_k) + \frac{1}{2}\left[\left(\frac{\alpha_k}{\sigma_k}\right)^2 s_k + \frac{\alpha_k}{\sigma_k} d_k\right]^T H(u_k) \left[\left(\frac{\alpha_k}{\sigma_k}\right)^2 s_k + \frac{\alpha_k}{\sigma_k} d_k\right]. \end{aligned} \tag{14}$$

It follows from (13) and (14) that

$$\begin{aligned} & f(x_k) + \gamma \left(\frac{\alpha_k}{\sigma_k}\right)^2 \left[g(x_k)^T (s_k + d_k) + \frac{1}{2} d_k^T H(x_k) d_k\right] \\ \leq\ & f(x_k) + \left(\frac{\alpha_k}{\sigma_k}\right)^2 g(x_k)^T [s_k + d_k] + \frac{1}{2}\left[\left(\frac{\alpha_k}{\sigma_k}\right)^2 s_k + \frac{\alpha_k}{\sigma_k} d_k\right]^T H(u_k) \left[\left(\frac{\alpha_k}{\sigma_k}\right)^2 s_k + \frac{\alpha_k}{\sigma_k} d_k\right]. \end{aligned}$$

Dividing both sides by $\left(\frac{\alpha_k}{\sigma_k}\right)^2$ and by simple manipulation we obtain

$$
\begin{aligned}
(\gamma - 1)\left[g(x_k)^T(s_k + d_k) + \frac{1}{2}d_k^T H(x_k)d_k\right]& \\
\leq \; \frac{1}{2}d_k^T\left[H(u_k) - H(x_k)\right]d_k + \frac{1}{2}\left(\frac{\alpha_k}{\sigma_k}\right)^2 s_k^T H(u_k)s_k + \frac{\alpha_k}{\sigma_k}d_k^T H(u_k)s_k& \qquad (15)
\end{aligned}
$$

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

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

By (12) we have $u_k \to x_*$ as $k \to \infty, k \in K_2$. Since $\gamma - 1 < 0$, $g(x_k)^T s_k \leq 0$, $g(x_k)^T d_k \leq 0$ and $d_k^T H(x_k)d_k \leq 0$ for all $k \in K_2$, it follows from (15) that

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

and hence

$$
g(x_*)^T(s_* + d_*) = 0, \qquad {d_*}^T H(x_*)d_* = 0.
$$

Condition 1 now implies that $g(x_*) = 0$ and Condition 3 gives $H(x_*)$ is positive semidefinite. Moreover, by Lemma 3.1 and Lemma 3.3, we have that $x_* \in \Omega_0$. The proof of assertion (a) can be easily completed similar to the proof of Theorem 1 of [14].

Finally, assertion (b) follows from known results [19, p.478], by taking into account Lemma 3.3(b).
□

# 4 Computation of the search directions

In this section, we describe an example of how to determine search directions that satisfy our assumptions using the Bunch-Parlett decomposition. Of course, the use of this decomposition is expensive, but as we stated in the introduction, the scope of this work is to understand the effect of incorporating negative curvature directions in algorithms via curvilinear linesearch.

The Bunch-Parlett decomposition is very easy to implement and it gives information about the distribution of the eigenvalues of the Hessian matrix (for details, see [3, 17]). We recall that (at iteration $k$) the Bunch-Parlett decomposition gives

$$
H = WDW^T,
$$

where $W$ is a $n \times n$ non singular matrix and $D$ is a symmetric $n \times n$ block diagonal matrix with one by one and two by two diagonal blocks. If we diagonalize the two by two blocks of the matrix $D$, we obtain the following representation

$$
H = V\Lambda V^T,
$$

where $V$ is a $n \times n$ non singular matrix and $\Lambda$ is a diagonal matrix which has the same number of negative (positive) diagonal elements as the Hessian matrix $H$ has negative (positive) eigenvalues.

As a first step, we consider the following vector

$$
\tilde{d} = -V^{-T}\bar{\Lambda}^{-1}V^{-1}g.
$$

Here $\bar{\Lambda} = \text{diag}\{\bar{\lambda}_i\}$, $i = 1, \ldots, n$ and $\bar{\lambda}_i$ is given by

$$\bar{\lambda}_i = \begin{cases} \delta & \text{if} \quad |\lambda_i| < \delta \\ \lambda_i & \text{if} \quad |\lambda_i| > \delta \end{cases},$$

where $\lambda_i$, $i = 1, \ldots, n$ are the diagonal elements of $\Lambda$. The vector $\tilde{d}$ can be split as

$$\tilde{d} = d_+ - d_-.$$

Here

$$d_+ = -V^{-T} B_+ V^{-1} g \quad \text{and} \quad d_- = V^{-T} B_- V^{-1} g, \tag{16}$$

with $B_+ = \text{diag}\left\{b_i^+\right\}$, $B_- = \text{diag}\left\{b_i^-\right\}$, $i = 1, \ldots n$ and the elements $b_i^+$ and $b_i^-$ are defined in the following manner

$$b_i^+ = \begin{cases} 0 & \text{if} \quad \bar{\lambda}_i \leq 0 \\ 1/\bar{\lambda}_i & \text{if} \quad \bar{\lambda}_i > 0, \end{cases} \qquad b_i^- = \begin{cases} 0 & \text{if} \quad \bar{\lambda}_i \geq 0 \\ 1/\bar{\lambda}_i & \text{if} \quad \bar{\lambda}_i < 0. \end{cases}$$

**Proposition 4.1** *The vectors $d_+$ and $d_-$, defined in (16) satisfy Conditions 1 and 2 of Section 2 with $s = d_+$ and $d = d_-$.*

**Proof** The proof follows immediately from the definitions of $d_+$ and $d_-$. □

On the basis of this result, we set $s = d_+$ in our algorithm. In order to find a negative curvature direction $d$ satisfying Condition 3, we recall the following direction from [17]:

$$d_{MS} = -\text{sgn}\left(g^T v\right) |\lambda_m(D)|^{1/2} v, \tag{17}$$

where $v$ is the solution of the following system

$$W^T v = z.$$

Here, $z$ is the eigenvector of $D$ corresponding to the minimum eigenvalue $\lambda_m(D)$. It is shown in [17] that $d_{MS}$ satisfies Condition 3. However, our numerical experience indicates that a better choice is

$$d = d_- + \eta\left(-\text{sgn}\left(g^T u\right) u\right),$$

where $u$ is the solution of the following system

$$W^T u = \sum_{j=1}^{p} z_j.$$

Here, $\{z_j : j = 1, \ldots, p\}$ are the eigenvectors of $D$ corresponding to the negative eigenvalues $\lambda_j$, $j = 1, \ldots, p$ and

$$\eta = \min\left\{1, \frac{\beta}{\|g\|}\right\} \min\left\{1, |\lambda_m(D)|\right\}.$$

Of course, if $\lambda_m(D) \geq 0$ we set $d = 0$; moreover if $d^T H d > 0$ we set $\eta = 0$. Using Proposition 4.1 and Lemma 4.3 of [17], it can be easily proved that this direction satisfies Condition 3.

The motivating property of our direction $d$ is as follows. If the norm of the gradient is large then $d$ almost coincides with $d_-$ and hence approximately minimizes the quadratic model of the objective function. Otherwise, when the norm of the gradient is small (and hence $d_-$ is small too), $d$ almost coincides with $d_{MS}$ and hence is closely related to the minimum eigenvalue of the Hessian matrix.

# 5 Computational Results

In order to evaluate the behavior of our new algorithm, we have used the Bunch-Parlett decomposition as discussed in the previous section and we have tested the resulting algorithm on all the small unconstrained problems available from the CUTE collection [2]. This test set covers the classical test problems along with a large number of nonlinear optimization problems of various difficulty representing both "academic" and "real world" applications.

We have compared the results with those obtained by the algorithm proposed in [14]. For both the methods the stopping criterion is $\|g\| \leq 10^{-5}$ and also the parameters of the stabilization scheme are the same: $M = 20$, $N = 20$ and $\Delta_0 = 10^3$. The other parameters required by the algorithm have been set in the following way: $\beta = 10^{-3}$ and $\delta$ equal to the machine precision. All the runs were carried out on an IBM RISC System/6000 375 using Fortran in double precision with the default optimization compiler option.

For comparison, we consider only the test problems coherently solved by at least one of the two methods, namely all the problems where the algorithms converge to the same stationary point within 5000 iterations; the resulting test set consists in 177 functions. In this comparison, the results of two runs are considered equal if they differ by at most 5% .

In the Appendix we report the complete results of both the algorithms on all the test problems. In order to give a summary of this extensive numerical testing, in Table 1 we report the number of times each method performs the best in terms of number of iterations, function and gradient evaluations:

| | NEW ALGORITHM | ALGORITHM [14] | tie |
|---|---|---|---|
| iterations | 69 | 17 | 91 |
| function evaluations | 155 | 18 | 4 |
| gradient evaluations | 69 | 17 | 91 |

Table 1: number of times each method performs the best

Table 2 shows the cumulative results for all the problems solved by both algorithms. In this table iterations stands for the total number of iterations needed to solve all these problems; the same for function and gradient evaluations.

| | NEW ALGORITHM | ALGORITHM [14] |
|---|---|---|
| iterations | 3979 | 10330 |
| function evaluations | 4293 | 17678 |
| gradient evaluations | 4148 | 10499 |

Table 2: cumulative results

Moreover, there are 8 problems solved by our new algorithm while the algorithm proposed in [14] fails on these problems. The failures are caused by excessive number of iterations or functions evaluations.

On the basis of these results, the new method generates a considerable computational savings, along with an increase in robustness.

We have also compared our algorithm to LANCELOT [4], a trust region based code for large–scale nonlinear optimization, written in Fortran. We believe that LANCELOT is close to the state of the art for nonlinear optimization codes. We ran this code, with default parameters, on the same 177 problems as mentioned above and have included the complete set of results in the Appendix. Below, we give two summary tables that compare our new algorithm to LANCELOT. Of course, in this comparison it should be taken into account that LANCELOT does not solve the Newton linear system exactly.

In Table 3 we report the number of times each method performs the best in terms of number of function and gradient evaluations for the complete suite of test problems, where ties denote results that differ by less than 5%. We have not included the 3 problems where LANCELOT and the new algorithm converge to different points (in the tables of the Appendix brackets around numbers indicate that LANCELOT converges to different points).

| | NEW ALGORITHM | LANCELOT | tie |
|---|---|---|---|
| function evaluations | 122 | 42 | 10 |
| gradient evaluations | 85 | 50 | 39 |

Table 3: number of times each method performs the best

Table 4 shows the cumulative results for all the problems solved by both algorithms. We have not included the 3 problems that LANCELOT failed on, the 3 problems where LANCELOT converges to different points, and the largest MSQRTBLS problem (which could be considered a failure for the new algorithm). In this table function evaluations stands for the total number of function evaluations needed to solve all the remaining problems; the same for gradient evaluations.

| | NEW ALGORITHM | LANCELOT |
|---|---|---|
| function evaluations | 4741 | 7915 |
| gradient evaluations | 4136 | 6809 |

Table 4: cumulative results

While these results might indicate that the new algorithm performs better than LANCELOT in terms of function and gradient evaluations, it should be pointed out that the linear algebra carried out by the new algorithm is considerably more expensive than that performed by LANCELOT. The corresponding CPU times for LANCELOT are therefore generally better than those of the new algorithm. This indicates that more research is required to develop a computationally fast technique for extracting the information required by the new algorithm. The results above show that this should lead to more robust and faster solution of these minimization problems.

## 6 Conclusions

In this work we propose an algorithmic model based on a modified Newton method [17] and a non-monotone stabilization strategy [14]. This model exploits interesting features of both approaches, namely the global convergence towards points that satisfy second order conditions and more reliable and efficient solution of highly nonlinear and ill-conditioned problems.

We have implemented an algorithm that is based on the proposed algorithmic model and uses the Bunch-Parlett decomposition for computing the search directions. We have compared this algorithm on a large set of test problems to a similar one proposed in [14] that does not use any curvature information of the objective function. The results indicate that the new algorithm outperforms the old one. We believe that they should rekindle interest in defining new methods based on the curvilinear linesearch approach and using directions of negative curvature.

The scope of this work was to understand how to incorporate directions of negative curvature within linesearch algorithms and to determine whether such algorithms are effective at solving practical, unconstrained problems. Although our computational results show our method outperforms the default algorithm implemented in LANCELOT package on small problems, further investigation is needed to define a practical and efficient algorithm for large scale problems, where the Newton linear system can not be solved exactly and linear algebra costs become important. Such work should ascertain the best way to compute a negative curvature direction satisfying the conditions required for convergence. The most natural way to obtain such directions is to extract information on the curvature of the objective function during the iterative processes that computes the Newton-type direction. We believe that some promising approaches to compute both a Newton-type direction and a negative curvature direction are

- the use of the conjugate gradient method [1, 5, 24],
- the use of Lanczos decomposition [18, 20],
- and several new modifications of Cholesky factorization [9, 10, 11, 21, 22].

## Acknowledgement

We thank the referees for their constructive comments and remarks that helped improve the paper.

## References


[1] M. Arioli, T. F. Chan, I. S. Duff, N. I. M. Gould, and J. K. Reid. Computing a search direction for large-scale linearly-constrained nonlinear optimization calculations. Technical Report TR/PA/93/34, CERFACS, 1993.

[2] I. Bongartz, A. R. Conn, N. Gould, and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. Publications du Départment de Mathématique Report 93/10, Facultés Universitaires De Namur, 1993. To appear in *ACM Trans. Math. Soft.*

[3] J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 8:639–655, 1971.

[4] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *LANCELOT: A Fortran package for Large–Scale Nonlinear Optimization (Release A).* Number 17 in Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, 1992.

[5] R. S. Dembo and T. Steihaug. Truncated Newton method algorithms for large–scale unconstrained optimization. *Mathematical Programming*, 26:190–212, 1983.

[6] N.Y. Deng, Y. Xiao, and F.J. Zhou. Nonmonotonic trust region algorithm. *Journal of Optimization Theory and Applications*, 76:259–285, 1993.

[7] S. P. Dirkse and M. C. Ferris. The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods & Software*, 1995, forthcoming.

[8] M. C. Ferris and S. Lucidi. Nonmonotone stabilization methods for nonlinear equations. *Journal of Optimization Theory and Applications*, 81:53–74, 1994.

[9] A. Forsgren, P.E. Gill, and W. Murray. A modified Newton method for unconstrained optimization. Technical Report SOL 89-12, Department of Operations Research, Stanford University, Stanford, California, 1989.

[10] A. Forsgren, P. E. Gill, and W. Murray. Computing modified Newton directions using a partial Cholesky factorization. Technical Report TRITA-MAT-1993-9, Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden, 1993.

[11] P. E. Gill, W. Murray, D. B. Poncélon, and M. A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM Journal on Matrix Analysis and Applications*, 13:292–311, 1992.

[12] P. E. Gill and W. Murray. Newton–type methods for unconstrained and linearly constrained optimization. *Mathematical Programming*, 7:311–350, 1974.

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

[14] L. Grippo, F. Lampariello, and S. Lucidi. A class of nonmonotone stabilization methods in unconstrained optimization. *Numerische Mathematik*, 59:779–805, 1991.

[15] G. P. McCormick. A modification of Armijo's step–size rule for negative curvature. *Mathematical Programming*, 13:111–115, 1977.

[16] G. P. McCormick. *Nonlinear Programming: Theory, Algorithms and Applications.* John Wiley & Sons, New York, 1983.

[17] J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, 16:1–20, 1979.

[18] S. G. Nash. Newton–type minimization via Lanczos method. *SIAM Journal on Numerical Analysis*, 21:770–788, 1984.

[19] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, San Diego, California, 1970.

[20] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12:617–629, 1975.

[21] T. Schlick. Modified Cholesky factorization for sparse preconditioners. *SIAM Journal on Scientific Computing*, 14:424–445, 1993.

[22] R. B. Schnabel and E. Eskow. A new modified Cholesky factorization. *SIAM Journal on Scientific and Statistical Computing*, 11:1136–1158, 1990.

[23] G. A. Shultz, R. B. Schnabel, and R. H. Byrd. A family of trust–region–based algorithms for unconstrained minimization. *SIAM Journal on Numerical Analysis*, 22:44–67, 1985.

[24] Ph. L. Toint. Towards an efficient sparsity exploiting Newton method for Minimization. In I. S. Duff, editor, *Sparse matrices and their uses*, pages 57–88. Academic Press, London, 1981.

[25] Ph. L. Toint. An assesment of non–monotone linesearch techniques for unconstrained optimization. Technical Report 94/14, Department of Mathematics, FUNDP, Namur, Belgium, 1994.

[26] Ph. L. Toint. A non–monotone trust–region algorithm for nonlinear optimization subject to convex constraints. Technical Report 94/24, Department of Mathematics, FUNDP, Namur, Belgium, 1994.

[27] Y. Xiao, and F. J. Zhou. Nonmonotone trust region methods with curvilinear path in unconstrained minimization. *Computing*, 48:303–317, 1992.

## Appendix

| | | NEW ALGORITHM | | | ALGORITHM [14] | | | LANCELOT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PROBLEM | n | ITER | FUN | GRAD | ITER | FUN | GRAD | ITER | FUN | GRAD |
| ALLINITU | 4 | 7 | 10 | 8 | 7 | 8 | 8 | 9 | 10 | 9 |
| ARWHEAD | 100 | 5 | 4 | 6 | 5 | 8 | 6 | 5 | 6 | 6 |
| BARD | 3 | 8 | 6 | 9 | 8 | 11 | 9 | 7 | 8 | 8 |
| BDQRTIC | 100 | 9 | 3 | 10 | 9 | 8 | 10 | 11 | 12 | 12 |
| BEALE | 2 | 7 | 4 | 8 | 5 | 6 | 6 | 7 | 8 | 8 |
| BIGGS6 | 6 | 30 | 38 | 31 | 81 | 155 | 82 | 56 | 57 | 46 |
| BOX3 | 3 | 7 | 5 | 8 | 7 | 11 | 8 | 7 | 8 | 8 |
| BRKMCC | 2 | 2 | 3 | 3 | 2 | 9 | 3 | 3 | 4 | 4 |
| BROWNAL | 10 | 7 | 3 | 8 | 7 | 5 | 8 | 5 | 6 | 6 |
| BROWNBS | 2 | 8 | 6 | 9 | 12 | 35 | 13 | 34 | 35 | 34 |
| BROWNDEN | 4 | 8 | 5 | 9 | 8 | 10 | 9 | 11 | 12 | 12 |
| BRYBND | 10 | 12 | 8 | 13 | 12 | 8 | 13 | 11 | 12 | 11 |
| BRYBND | 100 | 12 | 14 | 13 | 9 | 7 | 10 | 12 | 13 | 12 |
| CHNROSNB | 10 | 20 | 10 | 21 | 20 | 15 | 21 | 29 | 30 | 27 |
| CLIFF | 2 | 27 | 12 | 28 | 27 | 21 | 28 | 27 | 28 | 28 |
| CRAGGLVY | 4 | 17 | 7 | 18 | 15 | 8 | 16 | 16 | 17 | 16 |
| CRAGGLVY | 100 | 14 | 3 | 15 | 14 | 6 | 15 | 14 | 15 | 15 |
| CUBE | 2 | 23 | 20 | 24 | 5 | 11 | 6 | 36 | 37 | 33 |
| DENSCHNA | 2 | 5 | 4 | 6 | 5 | 9 | 6 | 5 | 6 | 6 |
| DENSCHNB | 2 | 6 | 12 | 7 | 6 | 7 | 7 | 3 | 4 | 4 |
| DENSCHNC | 2 | 10 | 6 | 11 | 10 | 11 | 11 | 9 | 10 | 10 |
| DENSCHND | 3 | 30 | 20 | 31 | 39 | 59 | 40 | 33 | 34 | 33 |
| DENSCHNE | 3 | 11 | 16 | 12 | 12 | 15 | 13 | 11 | 12 | 11 |
| DENSCHNF | 2 | 6 | 4 | 7 | 6 | 9 | 7 | 6 | 7 | 7 |
| DIXMAANA | 15 | 5 | 4 | 6 | 6 | 13 | 7 | 5 | 6 | 6 |
| DIXMAANA | 90 | 5 | 4 | 6 | 6 | 13 | 7 | 5 | 6 | 6 |
| DIXMAANB | 15 | 11 | 13 | 12 | 16 | 46 | 17 | 6 | 7 | 7 |
| DIXMAANB | 90 | 16 | 30 | 17 | 36 | 137 | 37 | 8 | 9 | 9 |
| DIXMAANC | 15 | 9 | 8 | 10 | 17 | 46 | 18 | 9 | 10 | 9 |
| DIXMAANC | 90 | 18 | 36 | 19 | 68 | 275 | 69 | 9 | 10 | 9 |
| DIXMAAND | 15 | 10 | 11 | 11 | 22 | 74 | 23 | 9 | 10 | 9 |
| DIXMAAND | 90 | 25 | 71 | 26 | 47 | 182 | 48 | 10 | 11 | 10 |
| DIXMAANE | 15 | 10 | 9 | 11 | 9 | 30 | 10 | 6 | 7 | 7 |
| DIXMAANE | 90 | 16 | 45 | 17 | 35 | 188 | 36 | 6 | 7 | 7 |
| DIXMAANF | 15 | 8 | 8 | 9 | 22 | 69 | 23 | 10 | 11 | 10 |
| DIXMAANF | 90 | 20 | 31 | 21 | 62 | 236 | 63 | 10 | 11 | 10 |
| DIXMAANG | 15 | 7 | 7 | 8 | 19 | 57 | 20 | 11 | 12 | 10 |
| DIXMAANG | 90 | 25 | 68 | 26 | 73 | 281 | 74 | 16 | 17 | 15 |
| DIXMAANH | 15 | 11 | 17 | 12 | 24 | 63 | 25 | 13 | 14 | 12 |
| DIXMAANH | 90 | 25 | 44 | 26 | 75 | 293 | 76 | 19 | 20 | 18 |
| DIXMAANI | 15 | 7 | 6 | 8 | 9 | 29 | 10 | 5 | 6 | 6 |
| DIXMAANI | 90 | 15 | 25 | 16 | 40 | 211 | 41 | 6 | 7 | 7 |

| | | NEW ALGORITHM | | | ALGORITHM [14] | | | LANCELOT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PROBLEM | n | ITER | FUN | GRAD | ITER | FUN | GRAD | ITER | FUN | GRAD |
| DIXMAANJ | 15 | 12 | 15 | 13 | 21 | 62 | 22 | 12 | 13 | 11 |
| DIXMAANJ | 90 | 30 | 64 | 31 | 88 | 291 | 89 | 20 | 21 | 18 |
| DIXMAANK | 15 | 27 | 32 | 28 | 38 | 84 | 39 | 16 | 17 | 14 |
| DIXMAANK | 90 | 35 | 67 | 36 | 86 | 304 | 87 | 18 | 19 | 17 |
| DIXMAANL | 15 | 13 | 17 | 14 | 40 | 78 | 41 | 15 | 16 | 13 |
| DIXMAANL | 90 | 36 | 71 | 37 | 74 | 292 | 75 | 42 | 43 | 35 |
| DIXON3DQ | 10 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| DJTL | 2 | 72 | 201 | 73 | 1978 | 2013 | 1979 | 135 | 134 | 113 |
| DQDRTIC | 10 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| DQDRTIC | 100 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| DQRTIC | 10 | 16 | 17 | 17 | 16 | 9 | 17 | 17 | 18 | 18 |
| DQRTIC | 100 | 24 | 25 | 25 | 24 | 9 | 25 | 26 | 27 | 27 |
| EDENSCH | 36 | 12 | 5 | 13 | 12 | 10 | 13 | 12 | 13 | 13 |
| EIGENALS | 110 | 22 | 43 | 23 | 28 | 59 | 29 | 24 | 25 | 22 |
| EIGENBLS | 110 | 67 | 143 | 68 | 120 | 571 | 121 | 171 | 172 | 140 |
| EIGENCLS | 30 | 31 | 59 | 32 | 98 | 430 | 99 | 58 | 59 | 50 |
| ENGVAL1 | 2 | 7 | 4 | 8 | 7 | 9 | 8 | 7 | 8 | 8 |
| ENGVAL1 | 100 | 7 | 3 | 8 | 7 | 8 | 8 | 7 | 8 | 8 |
| ERRINROS | 10 | 18 | 10 | 19 | 21 | 28 | 22 | (53) | (54) | (47) |
| ERRINROS | 50 | 21 | 21 | 22 | 53 | 137 | 54 | 74 | 75 | 64 |
| EXPFIT | 2 | 8 | 7 | 9 | 11 | 46 | 12 | 13 | 14 | 12 |
| EXTROSNB | 10 | 74 | 38 | 75 | 74 | 49 | 75 | 521 | 522 | 441 |
| EXTROSNB | 50 | 97 | 49 | 98 | 82 | 49 | 83 | 867 | 868 | 711 |
| EXTROSNB | 100 | 85 | 42 | 86 | 82 | 49 | 83 | 934 | 935 | 764 |
| FLETCBV2 | 100 | 1 | 2 | 2 | 1 | 8 | 2 | 1 | 2 | 2 |
| FLETCHBV | 10 | 373 | 394 | 374 | 449 | 447 | 450 | 336 | 337 | 297 |
| FLETCHCR | 10 | 20 | 11 | 21 | 21 | 21 | 22 | 34 | 35 | 30 |
| FLETCHCR | 100 | 138 | 105 | 139 | 139 | 140 | 140 | 227 | 228 | 191 |
| FMINSURF | 16 | 24 | 63 | 25 | * | * | * | 17 | 18 | 15 |
| FMINSURF | 121 | 19 | 63 | 20 | * | * | * | 83 | 84 | 70 |
| FREUROTH | 2 | 6 | 5 | 7 | 6 | 10 | 7 | 9 | 10 | 10 |
| FREUROTH | 50 | 31 | 19 | 32 | 12 | 26 | 13 | 10 | 11 | 10 |
| FREUROTH | 100 | 27 | 9 | 28 | 16 | 44 | 17 | 10 | 11 | 10 |
| GENROSE | 10 | 27 | 18 | 28 | 23 | 46 | 24 | 41 | 42 | 36 |
| GENROSE | 100 | 86 | 130 | 87 | 128 | 471 | 129 | 120 | 121 | 97 |
| GROWTHLS | 3 | 83 | 87 | 84 | 32 | 48 | 33 | 173 | 174 | 143 |
| GULF | 3 | 21 | 28 | 22 | 33 | 106 | 34 | 35 | 36 | 32 |
| HAIRY | 2 | 45 | 56 | 46 | 411 | 1806 | 412 | 104 | 105 | 86 |
| HATFLDD | 3 | 14 | 10 | 15 | 16 | 12 | 17 | 19 | 20 | 20 |
| HATFLDE | 3 | 17 | 14 | 18 | 20 | 24 | 21 | 22 | 23 | 21 |
| HEART6LS | 6 | 320 | 466 | 321 | 347 | 1009 | 348 | * | * | * |
| HELIX | 3 | 18 | 17 | 19 | 19 | 13 | 20 | 12 | 13 | 12 |
| HILBERTA | 2 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| HILBERTB | 5 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| HIMMELBB | 2 | 12 | 14 | 13 | 19 | 25 | 20 | 12 | 13 | 12 |
| HIMMELBF | 4 | 98 | 101 | 99 | 3156 | 3133 | 3157 | 209 | 210 | 190 |

| | | NEW ALGORITHM | | | ALGORITHM [14] | | | LANCELOT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| PROBLEM | n | ITER | FUN | GRAD | ITER | FUN | GRAD | ITER | FUN | GRAD |
| HIMMELBG | 2 | 4 | 4 | 5 | 6 | 7 | 7 | 8 | 9 | 8 |
| HIMMELBH | 2 | 5 | 12 | 6 | 4 | 7 | 5 | 1 | 2 | 2 |
| JENSMP | 2 | 9 | 6 | 10 | 9 | 16 | 10 | 9 | 10 | 10 |
| KOWOSB | 4 | 10 | 16 | 11 | 8 | 11 | 9 | 10 | 11 | 8 |
| LIARWHD | 36 | 10 | 3 | 11 | 10 | 8 | 11 | 10 | 12 | 11 |
| LIARWHD | 100 | 10 | 3 | 11 | 10 | 8 | 11 | 13 | 14 | 14 |
| MANCINO | 100 | 5 | 3 | 6 | 5 | 9 | 6 | 15 | 16 | 16 |
| MOREBV | 10 | 2 | 3 | 3 | 2 | 8 | 3 | 2 | 3 | 3 |
| MOREBV | 100 | 1 | 2 | 2 | 1 | 8 | 2 | 1 | 2 | 2 |
| MSQRTALS | 4 | 12 | 7 | 13 | 8 | 14 | 9 | 15 | 16 | 15 |
| MSQRTALS | 49 | 2482 | 4980 | 2483 | * | * | * | (19) | (20) | (16) |
| MSQRTALS | 100 | 112 | 341 | 113 | * | * | * | 17 | 18 | 16 |
| MSQRTBLS | 9 | 14 | 23 | 15 | 21 | 57 | 22 | 13 | 14 | 12 |
| MSQRTBLS | 49 | 62 | 151 | 63 | * | * | * | 17 | 18 | 16 |
| MSQRTBLS | 100 | 415 | 1516 | 416 | * | * | * | 25 | 26 | 22 |
| NCB20B | 50 | 12 | 21 | 13 | 15 | 84 | 16 | 27 | 28 | 23 |
| NCB20B | 100 | 14 | 20 | 15 | 28 | 79 | 29 | 22 | 23 | 18 |
| NONDIA | 50 | 6 | 9 | 7 | 10 | 8 | 11 | 19 | 20 | 17 |
| NONDIA | 100 | 6 | 8 | 7 | 9 | 8 | 10 | 19 | 20 | 18 |
| NONDQUAR | 100 | 15 | 3 | 16 | 15 | 8 | 16 | 15 | 16 | 16 |
| OSBORNEA | 5 | 30 | 35 | 31 | 22 | 32 | 23 | 37 | 38 | 35 |
| OSBORNEB | 11 | 30 | 39 | 31 | * | * | * | (19) | (20) | (18) |
| PALMER1C | 8 | 1 | 2 | 2 | 1 | 8 | 2 | 17 | 18 | 18 |
| PALMER1D | 7 | 1 | 2 | 2 | 1 | 8 | 2 | 14 | 15 | 15 |
| PALMER2C | 8 | 1 | 2 | 2 | 1 | 8 | 2 | 12 | 13 | 13 |
| PALMER3C | 8 | 1 | 2 | 2 | 1 | 8 | 2 | 13 | 14 | 14 |
| PALMER4C | 8 | 1 | 2 | 2 | 1 | 8 | 2 | 46 | 47 | 47 |
| PENALTY1 | 4 | 16 | 7 | 17 | 16 | 12 | 17 | 36 | 37 | 32 |
| PENALTY1 | 10 | 23 | 9 | 24 | 23 | 14 | 24 | 40 | 41 | 35 |
| PENALTY1 | 50 | 28 | 8 | 29 | 28 | 13 | 29 | 69 | 70 | 59 |
| PENALTY1 | 100 | 32 | 9 | 33 | 32 | 14 | 33 | 48 | 49 | 44 |
| PENALTY2 | 4 | 8 | 5 | 9 | 8 | 10 | 9 | 8 | 9 | 9 |
| PENALTY2 | 10 | 29 | 14 | 30 | 29 | 19 | 30 | 98 | 99 | 82 |
| PENALTY2 | 50 | 21 | 9 | 22 | 21 | 14 | 22 | 39 | 40 | 36 |
| PENALTY2 | 100 | 19 | 7 | 20 | 19 | 12 | 20 | 19 | 20 | 20 |
| PENALTY3 | 50 | 17 | 27 | 18 | 18 | 16 | 19 | 21 | 22 | 21 |
| PENALTY3 | 100 | 16 | 9 | 17 | 17 | 38 | 18 | * | * | * |
| PFIT1LS | 3 | 168 | 238 | 169 | 48 | 36 | 49 | 433 | 434 | 351 |
| PFIT2LS | 3 | 36 | 34 | 37 | 42 | 35 | 43 | 249 | 250 | 220 |
| PFIT3LS | 3 | 38 | 38 | 39 | 79 | 76 | 80 | 155 | 156 | 130 |
| PFIT4LS | 3 | 57 | 58 | 58 | 229 | 630 | 230 | 386 | 387 | 317 |
| POWELLSG | 4 | 15 | 5 | 16 | 15 | 10 | 16 | 15 | 16 | 16 |
| POWELLSG | 20 | 16 | 3 | 17 | 16 | 8 | 17 | 15 | 16 | 16 |
| POWELLSG | 100 | 16 | 3 | 17 | 16 | 8 | 17 | 15 | 16 | 16 |

| PROBLEM | n | NEW ALGORITHM | | | ALGORITHM [14] | | | LANCELOT | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ITER | FUN | GRAD | ITER | FUN | GRAD | ITER | FUN | GRAD |
| POWER | 10 | 17 | 4 | 18 | 17 | 9 | 18 | 16 | 17 | 17 |
| POWER | 50 | 21 | 3 | 22 | 21 | 8 | 22 | 19 | 20 | 20 |
| POWER | 100 | 23 | 4 | 24 | 23 | 9 | 24 | 22 | 23 | 23 |
| QUARTC | 25 | 19 | 20 | 20 | 19 | 8 | 20 | 21 | 22 | 22 |
| QUARTC | 100 | 24 | 25 | 25 | 24 | 9 | 25 | 26 | 27 | 27 |
| ROSENBR | 2 | 5 | 5 | 6 | 5 | 11 | 6 | 30 | 31 | 26 |
| S308 | 2 | 9 | 6 | 10 | 9 | 11 | 10 | 10 | 11 | 10 |
| SCHMVETT | 3 | 3 | 3 | 4 | 3 | 13 | 4 | 3 | 4 | 4 |
| SCHMVETT | 10 | 3 | 3 | 4 | 3 | 13 | 4 | 3 | 4 | 4 |
| SCHMVETT | 100 | 3 | 3 | 4 | 3 | 13 | 4 | 3 | 4 | 4 |
| SINQUAD | 5 | 11 | 4 | 12 | 11 | 9 | 12 | 11 | 12 | 12 |
| SINQUAD | 50 | 11 | 3 | 12 | 11 | 8 | 12 | 38 | 39 | 35 |
| SINQUAD | 100 | 11 | 3 | 12 | 11 | 8 | 12 | 56 | 57 | 50 |
| SISSER | 2 | 12 | 5 | 13 | 12 | 10 | 13 | 12 | 13 | 13 |
| SNAIL | 2 | 118 | 129 | 119 | 129 | 311 | 130 | 90 | 91 | 77 |
| SPMSRTLS | 28 | 21 | 28 | 22 | 28 | 87 | 29 | 12 | 13 | 11 |
| SPMSRTLS | 100 | 27 | 65 | 28 | 111 | 444 | 112 | 14 | 15 | 13 |
| SROSENBR | 10 | 5 | 4 | 6 | 5 | 10 | 6 | 6 | 7 | 7 |
| SROSENBR | 50 | 5 | 4 | 6 | 5 | 10 | 6 | 6 | 7 | 7 |
| SROSENBR | 100 | 5 | 4 | 6 | 5 | 10 | 6 | 6 | 7 | 7 |
| TOINTGOR | 50 | 6 | 3 | 7 | 6 | 9 | 7 | 9 | 10 | 10 |
| TOINTGSS | 10 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| TOINTGSS | 50 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| TOINTGSS | 100 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| TOINTPSP | 50 | 112 | 78 | 113 | 71 | 246 | 72 | 23 | 24 | 21 |
| TOINTQOR | 50 | 1 | 2 | 2 | 1 | 8 | 2 | 6 | 7 | 7 |
| TQUARTIC | 5 | 1 | 2 | 2 | 1 | 8 | 2 | 1 | 2 | 2 |
| TQUARTIC | 50 | 1 | 2 | 2 | 1 | 8 | 2 | 10 | 11 | 9 |
| TQUARTIC | 100 | 1 | 2 | 2 | 1 | 8 | 2 | 12 | 13 | 12 |
| TRIDIA | 10 | 1 | 2 | 2 | 1 | 8 | 2 | 1 | 2 | 2 |
| TRIDIA | 50 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| TRIDIA | 100 | 1 | 2 | 2 | 1 | 8 | 2 | 2 | 3 | 3 |
| VARDIM | 10 | 14 | 4 | 15 | 14 | 9 | 15 | 14 | 15 | 15 |
| VARDIM | 50 | 22 | 4 | 23 | 22 | 9 | 23 | 22 | 23 | 23 |
| VARDIM | 100 | 25 | 4 | 26 | 25 | 9 | 26 | 25 | 26 | 26 |
| VAREIGVL | 10 | 12 | 7 | 13 | 50 | 48 | 51 | 18 | 19 | 14 |
| VAREIGVL | 50 | 13 | 4 | 14 | 13 | 9 | 14 | 13 | 14 | 14 |
| VAREIGVL | 100 | 12 | 3 | 13 | 12 | 8 | 13 | 12 | 13 | 13 |
| VIBRBEAM | 8 | 23 | 34 | 24 | 38 | 84 | 39 | * | * | * |
| WATSON | 12 | 8 | 5 | 9 | 9 | 12 | 10 | 8 | 9 | 9 |
| WATSON | 31 | 147 | 349 | 148 | * | * | * | 11 | 12 | 12 |
| WOODS | 4 | 16 | 10 | 17 | 16 | 15 | 17 | 15 | 16 | 16 |
| WOODS | 100 | 17 | 8 | 18 | 17 | 13 | 18 | 15 | 16 | 16 |
| YFITU | 3 | 20 | 22 | 21 | 17 | 20 | 18 | 94 | 95 | 80 |
| ZANGWIL2 | 2 | 1 | 2 | 2 | 1 | 8 | 2 | 1 | 2 | 2 |