Exploiting Negative Curvature Directions in Linesearch Methods for Unconstrained Optimization

N. I. M. Gould*　S. Lucidi†　M. Roma‡　Ph. L. Toint§

November 25, 1997

Abstract

In this paper we consider the definition of new efficient linesearch algorithms for solving large scale unconstrained optimization problems which exploit the local nonconvexity of the objective function. Existing algorithms of this class compute, at each iteration, two search directions: a Newton-type direction which ensures a global and fast convergence, and a negative curvature direction which enables the iterates to escape from the region of local nonconvexity. A new point is then generated by performing a movement along a curve obtained by combining these two directions. However, the respective scaling of the directions is typically ignored. We propose a new algorithm which aims to avoid the scaling problem by selecting the more promising of the two directions, and then performs a step along this direction. The selection is based on a test on the rate of decrease of the quadratic model of the objective function. We prove global convergence to second-order critical points for the new algorithm, and report some preliminary numerical results.

1 Introduction

We consider the unconstrained minimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.1}$$

where f is a real valued function on $\mathbb{R}^n$. We assume throughout that both the gradient $g(x) = \nabla_x f(x)$ and the Hessian matrix $H(x) = \nabla_{xx} f(x)$ of f exis t and are continuous. Our aim is to define a robust and efficient algorithm able to handle large scale problems.

Many algorithms have been proposed for solving this class of problems. In this paper we intend to concentrate on a particular aspect which we believe to play an important role in designing efficient algorithms, namely the effective use of the second order information contained in the Hessian matrix. It is now accepted that computing second derivatives for a large class of optimization problems is not only feasible but relatively inexpensive. As a result, more information about the problem is

*Department for Computation and Information, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England. E-mail: n.gould@rl.ac.uk

†Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”, via Buonarroti 12 - 00185 Roma, Italy. E-mail: lucidi@dis.uniroma1.it

‡Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”, via Buonarroti 12 - 00185 Roma, Italy. E-mail: roma@dis.uniroma1.it

§Department of Mathematics, Facultés Universitaires ND de la Paix, 61 rue de Bruxelles, B-5000 Namur, Belgium. E-mail: pht@math.fundp.ac.be

available than simply from the gradient, and one would like to exploit it. To these ends, we intend to exploit negative curvature directions, (i.e. directions d such that $d^T H(x) d < 0$), when they exist. In these directions, the quadratic model of the objective function is unbounded below, which shows potential for a large reduction of the objective function. Moreover, it is well-known that algorithms using such directions may be proved to converge to second-order critical points [10, 11, 12].

In what follows, we focus our attention on Newton-type algorithms because of their good asymptotic convergence properties. Such algorithms can be made globally convergent using one of two basic approaches, the linesearch and the trust region approach. We shall concentrate on linesearch algorithms in the spirit of [10, 11]. The key idea is to determine, at each iteration, a pair of descent directions, (s_k, d_k) where, loosely speaking, s_k represents a direction calculated from positive curvature information given by the Hessian matrix, and d_k is a negative curvature direction. This pair of directions is then used in a search along the trajectory

$$x(\alpha) = x_k + \alpha^2 s_k + \alpha d_k \tag{1.2}$$

and the new point is obtained by a *curvilinear linesearch* along this path. In [4] the approach proposed in [10, 11] was embedded in a nonmonotone framework and the importance of using negative curvature directions in solving small and medium problems was emphasized. In subsequent papers [8, 9], a new algorithmic framework, based on the Lanczos method for determining both search directions, was proposed in order to extend this approach to the solution of large scale problems. Numerical experience has shown the effectiveness of the combined use of the nonmonotone strategy and the negative curvature directions.

All these methods are characterized by a common feature, namely that the new point x_{k+1} is found by a curvilinear search along the path (1.2). Note that the relative scaling of s_k and d_k is not taken into account in the definition of the path. This may be a serious drawback because, for instance, too little weight may be given to the direction of negative curvature, although this direction might be the most significant for the minimization process. Indeed, the two directions s_k and d_k should ideally be exploited in different ways. A unit step along the Newton-type s_k is usually to be preferred, while the step along d_k typically requires a more sophisticated linesearch.

The aim of this paper is to propose a new algorithmic framework which tries to overcome the previous drawbacks while still ensuring global convergence towards second order critical points. It is based on the simple idea of separating the contribution of the two directions by performing, at each iteration, a step along *one* of s_k or d_k. The crucial issue is then which direction to use at each iteration. It is evident that an efficient strategy should be based on the attempt to determine which is the most promising direction or, equivalently, to guess whether the positive curvature information are more significative than the negative curvature information or vice versa. The rule we adopt is based on the rate of decrease of the quadratic model of the objective function. In particular we compare the decrease of the quadratic model along the negative curvature direction by performing a unit steplength along a normalized d_k, with the decrease that we would obtain by performing a unit steplength along the normalized truncated Newton direction.

Since we are interested in solving large scale problems and thus cannot rely on matrix factorizations, we concentrate on iterative methods to compute the search directions. In particular, we consider the preconditioned conjugate gradient and Lanczos methods, and exploit the fact that they are closely related. Preliminary numerical testing shows that the approach proposed in this paper is promising.

The paper is organized as follows. In Section 2 we describe the details of the algorithm we propose, and we prove the convergence of the iterates to second order points in Section 3. In Section 4 we describe how we compute the search directions used in our algorithm and finally report in Section 5 the results of our numerical experiments.

2 The adaptive linesearch algorithm

In this section we describe our new algorithmic framework, and state the conditions required on the search directions in order to ensure the global convergence of the algorithm to second-order critical points, that is points where the gradient of the objective function is zero and where its Hessian matrix is positive semidefinite. We first state the required conditions on the search directions used in our algorithm.

Let s_k be a *gradient-related* descent direction, that is a direction for which the following conditions are satisfied.

Condition 1. *There exist positive numbers c_1 and c_2 such that*

$$\begin{aligned} s_k^T g_k &\leq -c_1 \|g_k\|^2 , \\ \|s_k\| &\leq c_2 \|g_k\| , \end{aligned}$$

where $g_k = g(x_k)$ and $\|\cdot\|$ is the Euclidean norm. Furthermore, let d_k be a direction of *sufficient negative curvature*, that is a direction for which the following conditions are satisfie d.

Condition 2. *The directions $\{d_k\}$ are such that, for some $\theta \in (0, 1)$,*

$$d_k^T g_k \leq 0, \quad d_k^T H_k d_k \leq 0, \quad d_k^T H_k d_k \leq (\theta \lambda_{\min}(H_k) + \eta(g_k))\|d_k\|^2,$$

where $\eta \to 0$ as $g_k \to 0$ and $\lambda_{\min}(H_k)$ is the leftmost eigenvalue of the Hessian matrix $H_k = H(x_k)$.

Condition 1 is standard condition on the Newton-type directions. The last inequality of Condition 2 is needed to ensure the second-order global convergence of the algorithm and, roughly speaking, it requires that the direction d_k has some resemblance to an eigenvector of the Hessian matrix corresponding to its leftmost eigenvalue. This requirement was introduced in [8] and is an extension of the assumption usually required to obtain second order convergence (see [11]). It indicates that the contribution of a direction d_k which has a strict connection with an eigenvector of the Hessian matrix corresponding to the most negative eigenvalue is essential only when the gradient is small.

We now describe the details of our algorithm. We denote the quadratic model of the function $f(x) - f(x_k)$ by $m(x_k + w) = \frac{1}{2} w^T H_k w + g_k^T w$.

Adaptive linesearch algorithm

Step 0. **Initialisation**

The initial point $x_0 \in \mathbb{R}^n$ *and the constants* $\beta \in (0, 1)$, $\tau > 0$ *and* $\mu \in (0, \frac{1}{2})$ *are given. Set* $k = 0$.

Step 1. **Test for convergence**

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

Step 2. **Computation and choice of the search direction**

Compute the search directions s_k *and* d_k.
If $d_k = 0$, *execute Step 3. Otherwise, rescale* d_k *such that* $\|d_k\| = 1$. *If*

$$\frac{g_k^T s_k}{\|s_k\|} \le \tau m(x_k + d_k) \tag{2.1}$$

then execute Step 3, otherwise execute Step 4.

Step 3. **Linesearch in a gradient-related direction**

Set $p_k = s_k$ *and compute* $\alpha_k = \beta^\ell$ *where* l *is the smallest nonnegative integer such that*

$$f(x_k + \alpha_k p_k) \le f(x_k) + \mu \left(\alpha_k g_k^T p_k + \tfrac{1}{2} \alpha_k^2 \min \left[0, p_k^T H_k p_k \right] \right) \tag{2.2}$$

Step 4. **Linesearch in a negative curvature direction**

Set $p_k = d_k$ *and choose* $\sigma_k > 0$. *If*

$$f(x_k + \sigma_k p_k) \le f(x_k) + \mu \left(\sigma_k g_k^T p_k + \tfrac{1}{2} \sigma_k^2 p_k^T H_k p_k \right), \tag{2.3}$$

compute $\alpha_k = \beta^\ell \sigma_k$, *where* ℓ *is the largest non-positive integer such that*

$$f(x_k + \alpha_k p_k) \le f(x_k) + \mu \left(\alpha_k g_k^T p_k + \tfrac{1}{2} \alpha_k^2 p_k^T H_k p_k \right) \tag{2.4}$$

and

$$f(x_k + \frac{\alpha_k}{\beta} p_k) > f(x_k) + \mu \left(\frac{\alpha_k}{\beta} g_k^T p_k + \tfrac{1}{2} \left(\frac{\alpha_k}{\beta} \right)^2 p_k^T H_k p_k \right) \tag{2.5}$$

Otherwise compute $\alpha_k = \beta^\ell \sigma_k$, *where* ℓ *is the smallest positive integer such that (2.4) holds.*

Step 5. **New iterate**

Set $x_{k+1} = x_k + \alpha_k p_k$, $k = k + 1$ *and go to Step 1.*

As in [4, 8, 10, 11], at each iteration we compute a pair of descent directions (s_k, d_k). The distinguishing feature of our new approach is that, instead of using a curvilinear search to produce a new trial point of the form

$$x_{k+1} = x_k + \alpha_k^2 s_k + \alpha_k d_k, \tag{2.6}$$

we select only one of the two directions (see Step 2), and the new point is chosen as

$$x_{k+1} = x_k + \alpha_k p_k,$$

where p_k is the direction s_k or $d_k/\|d_k\|$. We aim to select the best of these two directions by considering the *rate of decrease of the model* along both directions. In other words, we intend to choose s_k whenever

$$\frac{m(x_k + s_k)}{\|s_k\|} \geq \frac{m(x_k + d_k)}{\|d_k\|}. \tag{2.7}$$

If we assume that s_k minimizes the model in the direction $s_k/\|s_k\|$, then we know that

$$m(x_k + s_k) = \tfrac{1}{2} g_k^T s_k. \tag{2.8}$$

On the other hand, the scaling of the problem along d_k is unknown, and we may as well choose to normalize d_k, as in Step 2. Using this normalization, and substituting (2.8) in (2.7), we obtain our test (2.1) with $\tau = 2$.

If there is no negative curvature direction or if the gradient-related direction looks more profitable, we perform a backtracking linesearch (Step 3). This linesearch is of the Armijo variety, but includes a second-order term to encourage convergence to second-order critical points. On the other hand, if the negative curvature direction appears more attractive, then we perform a specialized linesearch (Step 4) that allows forward ($\ell \leq 0$) or backward ($\ell > 0$) stepping, starting from a guess σ_k. We allow forward steps because our guess σ_k may not reflect the local scaling of the problem, and because of the potential for a large decrease of the objective function along negative curvature directions. For future reference, we note from (2.4), (2.5) and (2.2) that, in all cases, we obtain a steplength α_k for which

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \mu \left(\alpha_k g_k^T p_k + \tfrac{1}{2} \alpha_k^2 \min \left[0, p_k^T H_k p_k \right] \right) \tag{2.9}$$

The flexibility in choosing σ_k may be exploited for improving numerical performance. For instance, we may choose σ_k as the steplength α_j that was computed at the previous linesearch along a negative curvature direction, in the hope that, in the mean time, the problem's scaling has not significantly changed.

We also emphasize that the test (2.1) is scale invariant, that is it does not depend on the actual length of s_k (nor d_k, since this latter direction is normalized before the test).

We now prove that the linesearch procedures are well defined.

Lemma 2.1 *Assume that s_k is a descent direction and that d_k is a normalized descent negative curvature direction. Suppose furthermore that f is bounded below on the level set $\Omega_0 = \{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}$. Then there exists an $\alpha_k > 0$ such that (2.9) is satisfied.*

Proof Whenever $p_k = s_k$ we distinguish two cases:

(i) $s_k H_k s_k \geq 0$;

(ii) $s_k H_k s_k < 0$.

In case (i), (2.9) becomes

$$f(x_k + \alpha_k s_k) \leq f(x_k) + \mu \alpha_k g_k^T s_k \tag{2.10}$$

which is the standard Armijo rule, while in case (ii) (2.9) becomes

$$f(x_k + \alpha_k s_k) \leq f(x_k) + \mu \left[\alpha_k g_k^T s_k + \tfrac{1}{2} \alpha_k^2 s_k^T H_k s_k \right]. \tag{2.11}$$

In order to show that there exists an $\alpha_k > 0$ satisfying (2.11) we proceed by contradiction; if this inequality (2.11) were never satisfied, then there exists a sequence α_j converging to 0 as $j \to \infty$ such that

$$f(x_k + \alpha_j s_k) - f(x_k) > \mu \left[\alpha_j g_k^T s_k + \tfrac{1}{2} \alpha_j^2 s_k^T H_k s_k \right]. \tag{2.12}$$

Using the mean-value theorem, and dividing both sides by α_j (2.12) can be rewritten as

$$g_k^T (x_k + \delta \alpha_j s_k) s_k > \mu \left[g_k^T s_k + \tfrac{1}{2} \alpha_j s_k^T H_k s_k \right],$$

where $\delta \in (0, 1)$. Therefore we have

$$g_k^T (x_k + \delta \alpha_j s_k) s_k - g_k^T s_k > (\mu - 1) g_k^T s_k + \tfrac{1}{2} \mu \alpha_j s_k^T H_k s_k$$

which, for $j \to \infty$, yields

$$(\mu - 1) g_k^T s_k < 0$$

contradicting the fact that $\mu \in (0, \frac{1}{2})$ and $g_k^T s_k < 0$.

Whenever $p_k = d_k$, (2.9) becomes

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \mu \left[\alpha_k g_k^T d_k + \tfrac{1}{2} \alpha_k^2 d_k^T H_k d_k \right]. \tag{2.13}$$

If test (2.3) is satisfied, the existence of a finite ℓ is implied by (2.13) and the assumption that f is bounded below. Assume now that (2.3) fails. In order to show that there exists an $\alpha_k > 0$ satisfying (2.13), we again proceed by contradiction. If the inequality (2.13) is never satisfied, then there exists a sequence α_j converging to 0 as $j \to \infty$ such that

$$f(x_k + \alpha_j d_k) - f(x_k) > \mu \left[\alpha_j g_k^T d_k + \tfrac{1}{2} \alpha_j^2 d_k^T H_k d_k \right]. \tag{2.14}$$

By the mean-value theorem, (2.14) can be rewritten as

$$\alpha_j g_k^T d_k + \tfrac{1}{2} \alpha_j^2 d_k^T H(x_k + \delta \alpha_j d_k) d_k > \mu \left[\alpha_j g_k^T d_k + \tfrac{1}{2} \alpha_j^2 d_k^T H_k d_k \right]$$

where $\delta \in (0, 1)$. Dividing both sides by α_j we obtain

$$g_k^T d_k + \tfrac{1}{2} \alpha_j d_k^T H(x_k + \delta \alpha_j d_k) d_k - \tfrac{1}{2} \alpha_j d_k^T H_k d_k > \mu \left[g_k^T d_k + \tfrac{1}{2} \alpha_j d_k^T H_k d_k \right] - \tfrac{1}{2} \alpha_j d_k^T H_k d_k.$$

Therefore we have that

$$\tfrac{1}{2}(\mu - 1) \alpha_j d_k^T H_k d_k \leq \tfrac{1}{2}(\mu - 1) \alpha_j d_k^T H_k d_k + (\mu - 1) g_k^T d_k < \tfrac{1}{2} \alpha_j d_k^T \left[H(x_k + \delta \alpha_j d_k) - H_k \right] d_k.$$

Hence it follows that

$$(\mu - 1) d_k^T H_k d_k < d_k^T \left[H(x_k + \delta \alpha_j d_k) - H_k \right] d_k \tag{2.15}$$

where $\alpha_j \to 0$ as $j \to \infty$. This contradicts the fact that the left hand side of (2.15) is positive, since $\mu \in (0, \frac{1}{2})$ and $d_k H_k d_k < 0$. $\square$

3 Convergence analysis

In this section we study the convergence properties of our algorithm. In particular we prove that, under Conditions 1 and 2 the iterates converge to second-order critical points.

Theorem 3.1 *Let f be twice continuously differentiable, let x_0 be given and suppose that the level set $\Omega_0 = \{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}$ is compact. Assume that the directions s_k and d_k satisfy Conditions 1 and 2. Let $\{x_k\}$ be the points produced by the Algorithm. Then, every limit point x_* of $\{x_k\}$ belongs to Ω_0 and satisfy $g(x_*) = 0$. Moreover $H(x_*)$ is positive semidefinite.*

Proof Because of the compactness of Ω_0, we know that the sequence of iterates $\{x_k\}$ admits at least one limit point, and that all limit points belong to Ω_0. Suppose now that x_* is a limit point. Let K_s and K_d be index sets of two subsequences of iterates converging to x_* such that

(i) for all $k \in K_s$, (2.1) holds and hence

$$f(x_k + \alpha_k s_k) \leq f(x_k) + \mu \left(\alpha_k g_k^T s_k + \tfrac{1}{2}\alpha_k^2 \min\left[0, s_k^T H_k s_k\right]\right), \tag{3.1}$$

and

(ii) for all $k \in K_d$, (2.1) fails and hence

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \mu \left(\alpha_k g_k^T d_k + \tfrac{1}{2}\alpha_k^2 d_k^T H_k d_k\right). \tag{3.2}$$

Note that one of these index sets may be empty, but not both.

In order to prove that $g(x_*) = 0$ we proceed by contradiction. Suppose that $\|g_k\| > \epsilon$ for all $k \in K_s \cup K_d$.

Suppose first that K_s is not empty. Then we have

$$f(x_k + \alpha_k s_k) \leq f(x_k) + \mu \left(\alpha_k g_k^T s_k + \tfrac{1}{2}\alpha_k^2 \min\left[0, s_k^T H_k s_k\right]\right) \leq f(x_k) + \mu\alpha_k g_k^T s_k$$

for each $k \in K_s$, and hence that

$$|f(x_{k+1}) - f(x_k)| \geq \mu\alpha_k \left|g_k^T s_k\right|.$$

It follows that $\alpha_k \left|g_k^T s_k\right| \to 0$, as $k \to \infty, k \in K_s$. Therefore either $\alpha_k \to 0$ or $\left|g_k^T s_k\right| \to 0$ as $k \to \infty, k \in K_s$.

Suppose first that $\alpha_k \to 0$ as $k \to \infty, k \in K_s$. Since

$$f\left(x_k + \frac{\alpha_k}{\beta} s_k\right) - f(x_k) > \mu \left(\frac{\alpha_k}{\beta} g_k^T s_k + \tfrac{1}{2}\left(\frac{\alpha_k}{\beta}\right)^2 \min\left[0, s_k^T H_k s_k\right]\right),$$

then by the mean-value theorem we have, for $k \in K_s$,

$$\frac{\alpha_k}{\beta} g^T(x_k + \delta\frac{\alpha_k}{\beta} s_k) s_k > \mu \left(\frac{\alpha_k}{\beta} g_k^T s_k + \tfrac{1}{2}\left(\frac{\alpha_k}{\beta}\right)^2 \min\left[0, s_k^T H_k s_k\right]\right),$$

for some $\delta \in (0,1)$. Dividing by α_k/β and by $\|s_k\|$, we obtain

$$\frac{g^T(x_k + \delta\frac{\alpha_k}{\beta} s_k) s_k}{\|s_k\|} > \mu \left(\frac{g_k^T s_k}{\|s_k\|} + \tfrac{1}{2}\frac{\alpha_k}{\beta}\frac{\min\left[0, s_k^T H_k s_k\right]}{\|s_k\|}\right) \tag{3.3}$$

for $k \in K_s$. Now, we can extract a subsequence whose indices lie in the set $K_s' \subseteq K_s$ such that

$$x_k \to x_* \quad \text{and} \quad \frac{s_k}{\|s_k\|} \to s_*$$

for $k \in K_s'$. From (3.3), taking the limit as $k \to \infty, k \in K_s'$ we obtain that

$$(1-\mu) g^T(x_*) s_* \geq 0,$$

and, using Condition 1, it follows that $g(x_*) = 0$, which contradicts the fact that $\|g_k\| > \epsilon$. Hence α_k cannot tend to zero for $k \in K_s$. This implies that there exists a subsequence $K_s'' \subseteq K_s$ such that $\left|g_k^T s_k\right| \to 0$ as $k \to \infty, k \in K_s''$. Condition 1 and the continuity of the gradient imply that $g(x_*) = 0$, which again contradicts the assumption that $\|g_k\| > \epsilon$. Hence this latter assumption is itself impossible and we conclude that $g(x_*) = 0$ whenever K_s is not empty.

Now, suppose that K_d is not empty. In this case, it follows from (3.2) that

$$|f(x_{k+1}) - f(x_k)| \geq \mu \left| \alpha_k g_k^T d_k + \tfrac{1}{2} \alpha_k^2 d_k^T H_k d_k \right|.$$

for $k \in K_d$, and hence that

$$\left| \alpha_k g_k^T d_k + \tfrac{1}{2} \alpha_k^2 d_k^T H_k d_k \right| \to 0 \quad \text{as} \quad k \to \infty.$$

Therefore, either

$$g_k^T d_k \to 0 \quad \text{and} \quad d_k^T H_k d_k \to 0 \quad (k \to \infty, k \in K_d) \tag{3.4}$$

or $\alpha_k \to 0$ when $k \to \infty, k \in K_d$. If $\alpha_k \to 0$, $k \to \infty, k \in K_d$, we have

$$f\left(x_k + \frac{\alpha_k}{\beta} d_k\right) - f(x_k) > \mu \left(\frac{\alpha_k}{\beta} g_k^T d_k + \tfrac{1}{2} \left(\frac{\alpha_k}{\beta}\right)^2 d_k^T H_k d_k \right),$$

which, by the mean-value theorem, can be rewritten as

$$\frac{\alpha_k}{\beta} g_k^T d_k + \tfrac{1}{2} \frac{\alpha_k^2}{\beta^2} d_k^T \bar{H}_k d_k > \mu \left(\frac{\alpha_k}{\beta} g_k^T d_k + \tfrac{1}{2} \left(\frac{\alpha_k}{\beta}\right)^2 d_k^T H_k d_k \right) \tag{3.5}$$

for some $\delta \in (0,1)$, $k \in K_d$, and where $\bar{H}_k = H\left(x_k + \delta \frac{\alpha_k}{\beta} d_k\right)$. From (3.5) and Condition 2 we obtain

$$0 \leq (\mu - 1) \left[g_k^T d_k + \tfrac{1}{2} \frac{\alpha_k}{\beta} d_k^T H_k d_k \right] < \tfrac{1}{2} \frac{\alpha_k}{\beta} d_k^T \left[\bar{H}_k - H_k\right] d_k \tag{3.6}$$

and

$$0 \leq (\mu - 1) d_k^T H_k d_k \leq d_k^T \left[\bar{H}_k - H_k\right] d_k. \tag{3.7}$$

for $k \in K_d$. By (3.6) we have that, for $k \in K_d$, $g_k^T d_k \to 0$ and by (3.7) we have $d_k^T H_k d_k \to 0$, as $k \to \infty, k \in K_d$. Therefore, we can conclude that (3.4) holds even when $\alpha_k\ to 0$. But, as $k \in K_d$, and therefore that $\frac{1}{2} g_k^T s_k / \|s_k\| \geq \tau m(x_k + d_k)$, we have, from Condition 1, that

$$\tau \left| g_k^T d_k + \tfrac{1}{2} d_k^T H_k d_k \right| \geq \frac{\left|g_k^T s_k\right|}{\|s_k\|} \geq \frac{c_1}{c_2} \|g_k\| > \frac{c_1}{c_2} \epsilon,$$

which contradicts (3.4). Thus our assumption that $\|g_k\| > \epsilon$ is again impossible and we conclude that $g(x_*) = 0$ whenever K_d is not empty.

Hence, since $K_s \cup K_d \neq \emptyset$, we have proved that any limit point of the sequence is a stationary point. In order to complete the proof we proceed again by contradiction and assume that there exists x_* limit point of $\{x_k\}$ such that $g(x_*) = 0$ and $H(x_*)$ is not positive semidefinite. If we define K_* to be the set of indices of a subsequence of iterates $\{x_k\}$ a subsequence converging to x_*, we have, by Condition 2, that $d_k^T H_k d_k < -\epsilon$ for sufficiently large $k \in K_*$. As g_k converges to zero, we have that both $g_k^T s_k/\|s_k\|$ and $g_k^T d_k$ tend to zero when $k \to \infty, k \in K_*$, and hence that

$$\frac{g_k^T s_k}{\|s_k\|} > \tau \left(g_k^T d_k + \tfrac{1}{2} d_k^T H_k d_k \right). \tag{3.8}$$

for $k \in K_*$ sufficiently large. Therefore for $k \in K_*$ sufficiently large, condition (2.1) fails and the points x_k are generated by the algorithm by using the direction d_k. By repeating the same argument used before for the case where $K_d \neq \emptyset$, we obtain (3.4) again, which together with the fact that $g_k \to 0$ and Condition 2 yields

$$0 = \lim_{\substack{k\to\infty \\ k \in K_1}} \lambda_{\min}(H(x_k)) = \lambda_{\min}(H(x_*)).$$

This contradicts the fact that $\lim_{k\to\infty} H(x_k) = H(x_*)$ and $H(x_*)$ not positive semidefinite. Hence this latter assumption is false, which concludes the proof. $\square$

4 Computation of the search directions

We are interested in solving large scale problems. Therefore we focus our attention on iterative methods, and in particular on the preconditioned conjugate gradient (CG) and Lanczos methods. The CG algorithm is the most popular method for computing Newton-type directions. It is most effective when truncated, that is the iteration is terminated short of optimality (see [3, 14]). If the Hessian is indefinite, the CG procedure may fail or may prove to be unstable, and the equivalent Lanczos process is to be preferred [13]. Recently [8, 9] have used the Lanczos method in conjunction with a curvilinear linesearch. In practice, this produces both a good Newton-type direction and an efficient negative curvature direction after few iterations. We prefer the CG method here since, despite the drawbacks mentioned above, it is slightly less expensive.

In our algorithm, the negative curvature direction is computed via the strict connections between the CG and Lanczos methods (see Section 2 of [13] and [6]). To be more precise, we recall that after m iterations, the Lanczos algorithm generates m vectors $q_1, \ldots, q_m$, the *Lanczos vectors*, and the scalars $\gamma_1, \ldots, \gamma_m$ and $\delta_1, \ldots, \delta_{m-1}$. If we define the $m \times n$ matrix Q_m whose columns are the Lanczos vectors that is

$$Q_m = [q_1 \; q_2 \cdots \; q_m] \tag{4.1}$$

and the $m \times m$ tridiagonal symmetric matrix T_m given by

$$T_m = \begin{bmatrix} \gamma_1 & \delta_1 & & & \\ \delta_1 & \gamma_2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \gamma_{m-2} & \delta_{m-1} \\ & & & \delta_{m-1} & \gamma_m \end{bmatrix} \tag{4.2}$$

the fundamental Lanczos relationship (see [2, 5]) can be written as

$$H_k Q_m - Q_m T_m = \delta_m q_{m+1} e_m^T. \tag{4.3}$$

Therefore, if (λ_m, v_m) is an eigenvalue–eigenvector (Ritz) pair of T_m, we have that

$$H_k Q_m v_m - \lambda_m Q_m v_m = \delta_m e_m^T v_m q_{m+1}. \tag{4.4}$$

As a consequence $(\lambda_m, Q_m v_m)$ can be used as approximate eigenvalue–eigenvector pair of H_k whenever the right-hand side of (4.4) is small. As the tridiagonal matrix T_m and the Lanczos vectors can be easily recovered from the CG method (see [6, 13]), so long as this iteration does not break down, the eigenvalue–eigenvector pair of the Hessian matrix H_k may be estimated directly from the CG iteration. However, as the vectors q_k are normally discarded after use, a second pass may be needed to regenerate them when computing the approximate eigenvector, $Q_m v_m$. Note that if CG iteration breaks down, it is easy to continue the process by using the Lanczos method itself, as all the vectors required to continue the Lanczos iteration are available.

To compute the required negative curvature direction d_k, we therefore use the leftmost eigenvalue $\lambda_{\min}$ of the tridiagonal matrix T_m as an approximation of the leftmost eigenvalue of H_k and $Q_m v_{\min}$ (i.e. the eigenvector of the matrix T_m corresponding to $\lambda_{\min}$ pre-multiplied by Q_m) as an approximation of the corresponding eigenvector. If $\lambda_{\min}$ is negative, we select d_k as

$$d_k = -\text{sgn}\left[g_k^T \tilde{d}_k\right] \tilde{d}_k$$

where $\tilde{d}_k = Q_m v_{\min}$, and choose $d_k = 0$ otherwise. One drawback of the Lanczos process is that it is impossible to guarantee the last part of Condition 2, simply because the Krylov space investigated may not contain any eigenvector corresponding to the leftmost eigenvalue. However this happens with probability zero in exact arithmetic, and we don't expect it to happen in presence of rounding. The leftmost Ritz value found is determined to within 10%, and thus θ in Condition 2 is effectively 0.1.

Turning now to the required gradient-related direction, we may again use the CG/Lanczos algorithm. If c_i are the conjugate directions generated by this algorithm, a truncated Newton direction is given by

$$s_k = -\sum_{i=1}^{m} \frac{g_k^T c_i}{c_i^T H_k c_i} c_i$$

whenever H_k is positive definite. The stopping (truncation) rule is to stop at iteration m which is the first iteration for which the gradient of the model falls below $\min(\frac{1}{2}\|g_k\|, \|g_k\|^2)$ if $k \leq 5$ and below $\min(\frac{1}{10}\|g_k\|, \|g_k\|^2)$ otherwise. This choice allows the iterates to "settle down" for a few iterations before one really starts to require more accuracy. It is a compromise between a conceptually preferable strategy totally independent of k, and the observably efficient technique used in [3, 7], where the required accuracy increases linearly with k.

Here we allow for the possibility that H_k is indefinite by including only those terms corresponding to directions of *positive* curvature. That is if

$$I_1 = \left\{ i \in \{1, \ldots, m\} : c_i^T H_k c_i > 0 \right\},$$

we pick the direction

$$s_k = -\sum_{i \in I_1} \frac{g_k^T c_i}{c_i^T H_k c_i} c_i.$$

If $I_1 = \emptyset$ or if this direction is not gradient-related, we simply take the negative gradient. When negative curvature is encountered, the stopping test is uniquely determined by the quality of the approximation of the Ritz values, as explained above.

We also considered the choice

$$s_k = -\sum_{i \in I_1} \frac{g_k^T c_i}{c_i^T H_k c_i} c_i + \sum_{i \in I_2} \frac{g_k^T c_i}{c_i^T H_k c_i} c_i$$

where

$$I_2 = \left\{ i \in \{1, \ldots, m\} : c_i^T H_k c_i < 0 \right\}$$

(see [7]), but this alternative did not prove to be globally as effective in practice.

5 Numerical Experience

In order to evaluate the behaviour of our new algorithm, we tested it on the set of all large-scale unconstrained test problems from the CUTE collection [1] where negative curvature has been reported. All the test were performed on an IBM RISC System/6000 375. The codes are double precision Fortran 90 compiled under xlf90 with the optimization compiling option. We used the settings

$$c_1 = n\epsilon_{\text{mach}}, \quad c_2 = 10^{20}, \quad \beta = \tfrac{1}{2}, \quad \tau = 2 \quad \text{and} \quad \mu = 10^{-3}.$$

As indicated above, we chose the initial step for the linesearch along negative curvature directions as $\sigma_k = \alpha_j$, where $j < k$ is the index of the last iteration at which the test (2.1) fails. The function $\eta(g_k)$ in Condition 2 is chosen to be identically zero. All tests reported below are performed without preconditioning the CG/Lanczos algorithm, but of course preconditioning is possible (and may well be essential for more difficult examples).

We compare the new algorithm with an algorithm which uses the curvilinear path (2.6) in which s_k and d_k are computed as in our algorithm, and the stepsize α_k is determined by a simple backtracking strategy along the arc (1.2), starting from an initial step of one (see [8, 9, 10]). Note that taking $\alpha_k > 1$ is unnatural in this context since the step d_k would then likely be dominated by its gradient-related component, for which a stepsize larger than one is not expected to provide a good reduction in the objective function. Also note that the two algorithms are identical when no negative curvature is found.

The results are reported in Tables 1 and 2 in terms of the numbers of gradient and function evaluations, the number of CG iterations, the CPU time (in seconds), the final objective function value, the number of directions of negative curvature used (that is along which a linesearch is performed), and the number of negative curvature directions found. These two last numbers are identical for the curvilinear variant because the curvilinear arc is used whenever negative curvature is detected.

Although our tests are far from exhaustive, the results obtained indicate that the new algorithm is normally more efficient than the curvilinear variant when negative curvature is found—the performance on BROYDN7D, CHAINWOO, FREUROTH, NONCVXUN, NONCVXU2 and SPMSRTLS are not directly comparable as the two methods converge to different local minima. The main reason for this improvement appears to be that forward stepping in such directions is very effective. Remarkably, the difference in performance does not appear to be linked to the number of negative curvature directions found or used, but substantial differences in numerical efficiency and reliability may result from the use of a few, presumably highly significant, negative curvature directions (see GENHUMPS, MSQRTALS and SPMRTLS). We also note that the new algorithm use most of the negative curvature directions found, which confirms our intuition that these directions should be exploited when present. We finally note that other tests using values of τ other than 2 did not prove to be numerically as effective.

Problem	n	NG	NF	CG-it	TIME	F	d used	d found
BROYDN7D	1000	55	107	2260	18.67	3.8411E+02	42	43
BRYBND	1000	11	11	89	1.72	4.8174E-19	0	0
CHAINWOO	1000	445	803	19589	183.42	1.6596E+01	137	139
COSINE	1000	9	19	44	0.44	-9.9900E+02	1	1
CRAGGLVY	1000	15	15	107	1.30	3.3642E+02	0	0
CURLY10	1000	15	23	8298	39.53	-1.0032E+05	3	3
CURLY20	1000	16	28	8332	58.37	-1.0032E+05	4	4
CURLY30	1000	17	22	8104	73.22	-1.0032E+05	2	2
DIXMAANA	1500	8	8	8	0.45	1.0000E+00	0	0
DIXMAANE	1500	10	10	239	3.29	1.0000E+00	0	0
DQRTIC	1000	31	31	30	0.43	2.7446E-07	0	0
EIGENALS	930	45	60	1037	83.33	1.6649E-14	2	2
FREUROTH	1000	13	26	50	0.96	1.2147E+05	2	2
GENHUMPS	1000	1128	3096	25927	296.68	2.7970E-11	1065	1066
GENROSE	1000	592	1234	13340	114.79	1.0000E+00	411	411
MANCINO	100	11	11	11	11.07	6.0590E-22	0	0
MSQRTALS	1024	46	83	20051	2829.96	8.1053E-13	20	20
NCB20B	1000	20	35	2430	216.77	1.6760E+03	9	9
NONCVXUN	1000	230	498	15500	146.88	2.3346E+03	212	215
NONCVXU2	1000	250	546	9446	99.03	2.3186E+03	232	237
SINQUAD	1000	79	147	203	5.12	3.4971E-08	1	1
SPARSINE	1000	19	34	5751	78.05	1.2668E-16	6	7
SPMSRTLS	1000	14	22	325	4.27	4.4189E-16	3	3

Table 1: Results for the new algorithm.

Problem	n	NG	NF	CG-it	TIME	F	d used	d found
BROYDN7D	1000	45	246	1503	13.33	5.4307E+02	33	33
BRYBND	1000	11	11	89	1.49	4.8174E-19	0	0
CHAINWOO	1000	327	1435	14916	141.81	2.1171E+02	114	114
COSINE	1000	7	8	40	0.32	-9.9900E+02	1	1
CRAGGLVY	1000	15	15	107	1.18	3.3642E+02	0	0
CURLY10	1000	15	30	9013	42.51	-1.0032E+05	3	3
CURLY20	1000	17	46	9080	66.26	-1.0032E+05	4	4
CURLY30	1000	18	55	8438	76.52	-1.0032E+05	5	5
DIXMAANA	1500	8	8	8	0.32	1.0000E+00	0	0
DIXMAANE	1500	10	10	239	3.10	1.0000E+00	0	0
DQRTIC	1000	31	31	30	0.34	2.7446E-07	0	0
EIGENALS	930	56	147	1249	95.91	4.1574E-14	16	16
FREUROTH	1000	16	46	96	1.45	1.2136E+05	4	4
GENHUMPS	1000	1263	5182	28468	303.18	1.3985E-12	1215	1215
GENROSE	1000	555	2910	13351	117.41	1.0000E+00	412	412
MANCINO	100	11	11	11	10.96	6.0590E-22	0	0
MSQRTALS	1024	116	881	42636	6134.60	4.6202E-14	79	79
NCB20B	1000	20	125	2766	263.58	1.6760E+03	9	9
NONCVXUN	1000	124	852	11477	104.62	2.3280E+03	109	109
NONCVXU2	1000	145	1059	6268	64.18	2.3193E+03	135	135
SINQUAD	1000	85	195	212	4.71	3.2131E-08	9	9
SPARSINE	1000	14	19	3236	43.52	4.4309E-13	2	2
SPMSRTLS	1000	277	1116	19442	255.21	3.2814E+00	82	82

Table 2: Results for the curvilinear search algorithm.

6 Conclusions

We have proposed a linesearch method that exploits negative curvature directions without explicitly combining them with Newton-type directions to define a curvilinear path. This has the advantage that the relative scaling of these directions no longer matters. We have proved that all limit points of the sequence of iterates produced by the new algorithm are second-order critical. Preliminary numerical experiments indicate that the new algorithm is an improvement over the curvilinear search variant, particularly on harder problems.

Acknowledgements

We are grateful to Jorge Nocedal for his useful comments on the work in progress. Nick Gould and Philippe Toint wish to thank the members of the Department of Informatics and Systems (Rome) for their hospitality. Finally, the first three authors appreciate the support provided by the British Council/MURST travel grant ROM/889/95/53.

References

[1] I. Bongartz, A.R. Conn, N.I.M. Gould, and Ph.L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transaction on Mathematical Software*, 21:123–160, 1995.

[2] J.K. Cullum and R.A. Willoughby. *Lanczos algorithms for large symmetric eigenvalue computations*. Birkhauser, Boston, 1985.

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

[4] M.C. Ferris, S. Lucidi, and M. Roma. Nonmonotone curvilinear linesearch methods for unconstrained optimization. *Computational Optimization and Applications*, 6:117–136, 1996.

[5] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The John Hopkins Press, Baltimore, 1989. Second edition.

[6] N.I.M. Gould, S. Lucidi, M. Roma, and Ph.L. Toint. Solving the trust-region subproblem using the Lanczos method. Technical Report RAL-TR-97-028, Department for Computation and Information, Rutheford Appleton Laboratory, Oxfordshire, 1997.

[7] L. Grippo, F. Lampariello, and S. Lucidi. A truncated Newton method with nonmonotone linesearch for unconstrained optimization. *Journal of Optimization Theory and Applications*, 60:401–419, 1989.

[8] S. Lucidi, F. Rochetich, and M. Roma. Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization. To appear on *SIAM Journal on Optimization*.

[9] S. Lucidi and M. Roma. Numerical experiences with new truncated Newton methods in large scale unconstrained optimization. *Computational Optimization and Applications*, 7:71–87, 1997.

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

[11] 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.

[12] 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*, 22:47–67, 1985.

[13] J. Stoer. Solution of large linear systems of equations by conjugate gradient type methods. In A. Bachem, M.Grötschel, and B. Korte, editors, *Mathematical Programming. The State of the Art*, pages 540–565, Berlin Heidelberg, 1983. Springer-Verlag.

[14] 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, London, 1981. Academic Press.