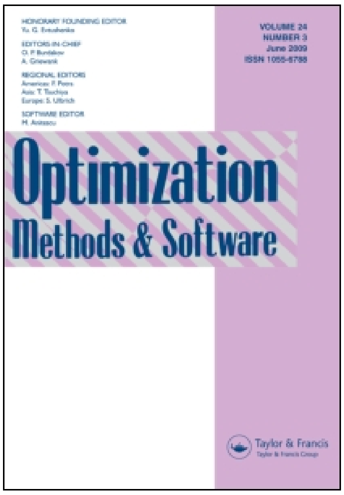

Optimization Methods and Software
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713645924

Exploiting negative curvature directions in linesearch methods for unconstrained optimization

N. I. M. Gould[a]; S. Lucidi[b]; M. Roma[b]; PH. L. Toint[c]
[a] Department for Computation and Information, Ruther ord Appleton & Laboratory, Chilton, Oxfordshire, England [b] Dipartimento di Informatica e Sistemistica, Universitá di Roma "La Sapienza", Roma, Italy [c] Department of Mathematics, Facultés Universitaires ND de la Paix, Namur, Belgium

To cite this Article Gould, N. I. M. , Lucidi, S. , Roma, M. and Toint, PH. L.(2000) 'Exploiting negative curvature directions in linesearch methods for unconstrained optimization', Optimization Methods and Software, 14: 1, 75 — 98
To link to this Article: DOI: 10.1080/10556780008805794
URL: http://dx.doi.org/10.1080/10556780008805794

Optimization Meth. & Soft., 2000, Vol. 14, pp. 75-98
Reprints available directly from the publisher
Photocopying permitted by license only

EXPLOITING NEGATIVE CURVATURE DIRECTIONS IN LINESEARCH METHODS FOR UNCONSTRAINED OPTIMIZATION

N. I. M. GOULD[a,*], S. LUCIDI[b,†], M. ROMA[b,‡] and PH. L. TOINT[c,¶]

[a]*Department for Computation and Information, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England;* [b]*Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Buonarroti 12-00185 Roma, Italy;* [c]*Department of Mathematics, Facultés Universitaires ND de la Paix, 61 rue de Bruxelles, B-5000 Namur, Belgium*

(Received 31 March 1999; In final form 10 January 2000)

In this paper we propose efficient new linesearch algorithms for solving large scale unconstrained optimization problems which exploit any local nonconvexity of the objective function. Current algorithms in this class typically compute a pair of search directions at every iteration: a Newton-type direction, which ensures both global and fast asymptotic convergence, and a negative curvature direction, which enables the iterates to escape from the region of local non-convexity. A new point is generated by performing a search along a line or a curve obtained by combining these two directions. However, in almost all of these algorithms, the relative scaling of the directions is not taken into account.

We propose a new algorithm which accounts for the relative scaling of the two directions. To do this, only the most promising of the two directions is selected at any given iteration, and a linesearch is performed along the chosen direction. The appropriate direction is selected by estimating the rate of decrease of the quadratic model of the objective function in both candidate directions. We prove global convergence to second-order critical points for the new algorithm, and report some preliminary numerical results.

Keywords: Linesearch algorithms; Newton-type direction; Convergence analysis; Conjugate gradient; Lanczos methods

*e-mail: n.gould@rl.ac.uk
†e-mail: lucidi@dis.uniroma1.it
‡e-mail: roma@dis.uniroma1.it
¶Corresponding author. e-mail: philippe.toint@fundp.ac.be

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 exist 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 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. Along these directions, the quadratic model of the objective function is unbounded from below, and this indicates the potential for a large reduction of the objective function. Algorithms which use such negative curvature directions can be made to converge globally to a second-order critical point using either a linesearch (see, *e.g.* [4, 7, 8, 13, 15–17]) or a trust region (see, *e.g.* [5, 18]) approach.

In what follows we concentrate on linesearch algorithms. At each iteration, such algorithms determine 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. These two directions are combined to define trajectories of the form

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

[4, 13–16],

$$x(\alpha) = x_k + \alpha s_k + \alpha^2 d_k \tag{1.3}$$

[9], or

$$x(\alpha) = x_k + \alpha s_k + \alpha d_k \tag{1.4}$$

[7, 8]. A new point is determined by taking a "suitable" step along the relevant trajectory. Unfortunately, the relative scaling of s_k and d_k is not taken into account when defining these trajectories. This may be a serious drawback as, for example, too little weight may be given to the direction of negative curvature, despite this direction being the more significant for the minimization process. Indeed, ideally the two directions s_k and d_k should be exploited in different ways. A unit step along the Newton-type s_k is normally sought, 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 – or at least to reduce – the above-mentioned drawbacks while at the same time still ensuring global convergence towards second order critical points. Our framework is based on the simple idea of using only *one* of the two directions s_k or d_k at any given iteration. This enables us to separate the contributions from the two directions, and to determine the steplength using a linesearch procedure appropriate for the particular direction selected. A standard backtracking Armjio-type linesearch might be used along the Newton-type direction, while one which steps forward as well as backward along the the negative curvature direction may be useful in escaping rapidly from regions of nonconvexity – the latter strategy helps to limit problems which arise because, unlike for the Newton-type direction s_k, we do not know of any natural scaling for the negative curvature 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 deduce whether the positive curvature information is 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 and it is able to guarantee the global convergence of the algorithm towards second order critical points. 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 rate of decrease that we would obtain by performing a unit steplength along the Newton direction.

The idea of selecting either a Newton-type direction or a direction which contains negative curvature information of the objective function, at each iteration, is not new. In particular the algorithms proposed in [6] and [17] are similar in aim to our approach. Both these algorithms use as criterion for selecting a Newton-type direction or an alternative the simple fact that a negative curvature direction exists (or that the Newton-type direction can not be computed).

More specifically, Fletcher and Freeman [6] use a simple negative curvature direction in preference to the Newton direction whenever the former can be found, but give no convergence details. Mukai and Polak [17] propose that a combination of the steepest descent and an eigenvector corresponding to the minimum eigenvalue of the Hessian matrix is used, whenever the Hessian matrix is indefinite. The use of this combination allows [17] to ensure global convergence to second order stationary points, but of course suffers from the arbitrary scaling of these two directions. As far as we are aware, the algorithm described in this paper is the first linesearch-type algorithm which is globally convergent to second order stationary points and which is free to choose between a gradient related Newton-type direction and a pure negative curvature direction (not necessarily gradient related).

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.

The proposed algorithm has been tested on a set of test functions from CUTE collection [1]. Its numerical behaviour has been compared with the one of an algorithm based on the curve (1.2) which has been shown to be very efficient in [13] and [14] when solving large scale unconstrained problems. The preliminary numerical testing we report here 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*

$$s_k^T g_k \leq -c_1 \|g_k\|^2,$$
$$\|s_k\| \leq c_2 \|g_k\|,$$

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 satisfied.

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 $\eta : \mathbb{R}^+ \to \mathbb{R}^+$ is a function such that $\eta(t) \to 0$ as $t \to 0$ 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 [13] and is an extension of the assumption usually required to obtain second order convergence (see [16]). 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 $m(x_k + w) = (1/2)w^T H_k w + g_k^T w$.

2.1. 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 $\mu \in (0, (1/2))$ 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

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

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

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

Step 4 Linesearch in a negative curvature direction Set $p_k = d_k$ and choose $\sigma_k > 0$. If

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

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

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

and

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

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.

Following [4, 6–8, 13, 15–17], 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 producing the new trial point along a combination of these directions, 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

$$\frac{m(x_k + s_k)}{\|s_k\|} \leq \frac{m(x_k + d_k)}{\|d_k\|}. \tag{2.6}$$

If we assume that s_k is exactly the Newton direction then we know that

$$m(x_k + s_k) = \frac{1}{2} g_k^T s_k. \tag{2.7}$$

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 (2.7) in (2.6), we obtain our test (2.1) with $\tau = 2$.

The reason behind our choice of this particular test derives from the possibility of ensuring good global convergence properties for the algorithm even if a nongradient-related negative curvature direction is used. Classical approaches to unconstrained optimization indicate that global convergence towards a first order stationary point can be guaranteed by taking "suitable" steps along good descent directions, namely directions which ensure, at least locally, a sufficient decrease of the objective function. This property is ensured by all the directions which guarantee a significant decrease of a local model of the objective function at any nonstationary point. In fact, the widely used gradient-related directions s_k are defined in a way that a unit steplength along their normalization $s_k/\|s_k\|$ produces a sufficient decrease of the linear model of the objective function. The role of a negative curvature direction is not related to the linear model of the function, but instead aims to exploit the local nonconvexity of the objective function. Its usefulness must be evaluated by considering the quadratic model of the objective function. Therefore, in our algorithm, we select the normalized negative curvature direction d_k if the decrease of the quadratic model obtained by performing a unit steplength along this direction is at least a fraction of the decrease of the linear model obtained by performing a unit steplength along $s_k/\|s_k\|$. This test, recalling that s_k is a gradient-related direction, implies that the negative curvature direction produces a significant reduction of the quadratic model of the objective function and guarantees that it can be used as search direction without preventing the global convergence towards first order stationary points. Furthermore the structure of this test and the assumptions on the direction d_k are able also to ensure the convergence towards second-order critical points

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

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \mu\left(\alpha_k g_k^T p_k + \frac{1}{2}\alpha_k^2 \min\left[0, p_k^T H_k p_k\right]\right) \qquad (2.8)$$

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 $\Omega_0 = \{x \in \mathbb{R}^n \,|\, f(x) \le f(x_0)\}$. Then there exists an $\alpha_k > 0$ such that* (2.8) *is satisfied.*

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

(i) $s_k^T H_k s_k \ge 0$;
(ii) $s_k^T H_k s_k < 0$.

In Case (i), (2.8) becomes

$$f(x_k + \alpha_k s_k) \le f(x_k) + \mu \alpha_k g_k^T s_k \tag{2.9}$$

which is the standard Armijo rule, while in Case (ii) (2.8) becomes

$$f(x_k + \alpha_k s_k) \le f(x_k) + \mu \left[\alpha_k g_k^T s_k + \frac{1}{2}\alpha_k^2 s_k^T H_k s_k\right]. \tag{2.10}$$

In order to show that there exists an $\alpha_k > 0$ satisfying (2.10) we proceed by contradiction; if this inequality (2. 10) were never satisfied, then there exists a sequence α_j converging to 0 as $j \to \infty$ such that

$$f(x_k + \alpha_j s_k) - f(x_k) > \mu \left[\alpha_j g_k^T s_k + \frac{1}{2}\alpha_j^2 s_k^T H_k s_k\right]. \tag{2.11}$$

Using the mean-value theorem, and dividing both sides by α_j (2.11) can be rewritten as

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

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

$$g^T(x_k + \delta_j \alpha_j s_k) s_k - g_k^T s_k > (\mu - 1) g_k^T s_k + \frac{1}{2}\mu \alpha_j s_k^T H_k s_k$$

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

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

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

Whenever $p_k = d_k$, (2.8) becomes

$$f(x_k + \alpha_k d_k) \le f(x_k) + \mu\left[\alpha_k g_k^T d_k + \frac{1}{2}\alpha_k^2 d_k^T H_k d_k\right]. \tag{2.12}$$

If test (2.3) is satisfied, the existence of a finite ℓ is implied by (2.12) 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 (2.12), we again proceed by contradiction. If the inequality (2.12) is never satisfied, then there exists a sequence α_j converging to 0 as $j \to \infty$ such that

$$f(x_k + \alpha_j d_k) - f(x_k) > \mu\left[\alpha_j g_k^T d_k + \frac{1}{2}\alpha_j^2 d_k^T H_k d_k\right]. \tag{2.13}$$

By the mean-value theorem, (2.13) can be rewritten as

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

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

$$\begin{aligned} g_k^T d_k &+ \frac{1}{2}\alpha_j d_k^T H(x_k + \delta_j \alpha_j d_k) d_k - \frac{1}{2}\alpha_j d_k^T H_k d_k \\ &> \mu\left[g_k^T d_k + \frac{1}{2}\alpha_j d_k^T H_k d_k\right] - \frac{1}{2}\alpha_j d_k^T H_k d_k. \end{aligned}$$

Therefore we have that

$$\begin{aligned} \frac{1}{2}(\mu - 1)\alpha_j d_k^T H_k d_k &\le \frac{1}{2}(\mu - 1)\alpha_j d_k^T H_k d_k \\ &\quad + (\mu - 1) g_k^T d_k < \frac{1}{2}\alpha_j d_k^T [H(x_k + \delta_j \alpha_j d_k) - H_k] d_k. \end{aligned}$$

Hence it follows that

$$(\mu - 1) d_k^T H_k d_k < d_k^T [H(x_k + \delta_j \alpha_j d_k) - H_k] d_k \tag{2.14}$$

where $\alpha_j \to 0$ as $j \to \infty$. This contradicts the fact that the left hand side of (2.14) is positive, since $\mu \in (0, (1/2))$ and $d_k H_k d_k < 0$. ■

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

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

and

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

$$f(x_k + \alpha_k d_k) \le f(x_k) + \mu\left(\alpha_k g_k^T d_k + \frac{1}{2}\alpha_k^2 d_k^T H_k d_k\right). \qquad (3.2)$$

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

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

Suppose first that K_s is infinite. Then we have

$$\begin{aligned} f(x_k + \alpha_k s_k) &\le f(x_k) + \mu\left(\alpha_k g_k^T s_k + \frac{1}{2}\alpha_k^2 \min\left[0, s_k^T H_k s_k\right]\right) \\ &\le f(x_k) + \mu\alpha_k g_k^T s_k \end{aligned}$$

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

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

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

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

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

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

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

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

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

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

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

Since $1 - \mu > 0$ and $g_k^T s_k < 0$ for all $k \in K_s'$ we have that $g^T(x_*) s_* = 0$ which implies, by using Condition 1, $g(x_*) = 0$ and this contradicts the fact that $\|g_k\| > \varepsilon$. 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 $|g_k^T s_k| \to 0$ 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 $\|g_k\| > \varepsilon$. Hence this latter assumption is itself impossible and we conclude that $g(x_*) = 0$ whenever K_s is infinite.

Now, suppose that K_d is infinite. In this case, it follows from (3.2) that

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

for $k \in K_d$, and hence that

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

Therefore, either

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

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

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

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

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

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

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

and

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

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 $g_k^T s_k / \|s_k\| \ge \tau m(x_k + d_k)$, we have, from Condition 1, that

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

which contradicts (3.4). Thus our assumption that $\| g_k \| > \varepsilon$ is again impossible and we conclude that $g(x_*) = 0$ whenever K_d is infinite.

Hence, 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 $\{x_k\}$ converging to x_*, we have, by Condition 2, that $d_k^T H_k d_k < -\varepsilon$ for sufficiently large $k \in K_*$. As g_k converges to zero, we have that $g_k^T s_k / \|s_k\|$ and $g_k^T d_k$ tend to zero when $k \to \infty$, $k \in K_*$, and hence that

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

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 is infinite, we obtain (3.4) again, which together with the fact that $g_k \to 0$ and Condition 2 yields

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

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. ∎

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, 20]). 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 [19]. Recently [13, 14] 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.

As regards the Newton-type direction, if we denote by p_i the conjugate directions generated by CG method, a truncated Newton direction is given by

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

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 $\min((1/2)\|g_k\|, \|g_k\|^2)$ if $k \leq 5$ and below $\min((1/10)\ \|g_k\|, \|g_k\|^2)$ 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, 12], 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

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

we pick the direction

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

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 below.

We also considered the choice

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

where

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

(see [12]), but this alternative did not prove to be globally as effective in practice.

Turning now to the required negative curvature direction, we again use the CG/Lanczos algorithm. In fact, in our algorithm, the negative curvature direction is computed *via* the strict connections between the CG and Lanczos methods (see Section 2 of [19] and [11]). 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

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

the fundamental Lanczos relationship (see [2, 10]) can be written as

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

where $e_m = (0, 0, \ldots, 0, 1)^T \in \mathbb{R}^m$. 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 [11, 19]), 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, in computing the approximate eigenvector $Q_m v_m$, the storage of the matrix Q_m is avoided by discarding the vectors q_k and by rerunning the recurrences to regenerate them; more in detail, during this second pass, at each iteration a vector q_k is computed and the eigenvector estimate is updated until the required accuracy is obtained, similarly to the truncated Lanczos approach described in Section 5 of [11]. 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

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

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.

5. NUMERICAL EXPERIENCE

In order to evaluate the behaviour of our new algorithm, we tested it on a set of 34 large-scale unconstrained test problems selected from the CUTE collection [1] where negative curvature has been reported. All the tests 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

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

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 :urvilinear path (1.2) 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 [13–15]). In [13, 14] it has been shown that this strategy can produce very efficient algorithms for solving large scale unconstrained problems. Note that taking $\alpha_k > 1$ is unnatural by using (1.2) 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 complete results are reported in the Appendix. Here we summarize the results obtained by the two algorithms on only 20 test problems where negative curvature directions have been encountered. In particular, in Table I we report time results obtained by the two algorithms on those test problems where they converge to the same local minima. These results are reported in terms of the numbers of gradient and function evaluations, the number of CG iterations, the CPU time (in seconds); in boldface we indicate the better of the two algorithms in terms of CPU time (we consider two results a tie when they differ by at most 5%); in the last row the totals (excluding problem MSQRTBLS) are reported. These results, although far from exhaustive, indicate that the new algorithm is normally more efficient than the curvilinear variant. In particular, in terms of CPU time, our new algorithm wins eight times and only on problem the curvilinear search algorithm performs better. Moreover, the curvilinear

TABLE I Comparison between the two algorithms

		New algorithm				*Curvilinear search algorithm*			
Problem	*n*	*NG*	*NF*	*CG-it*	*Time*	*NG*	*NF*	*CG-it*	*Time*
COSINE	1000	9	19	44	0.44	7	8	40	0.32
CURLY10	1000	15	23	8298	**39.53**	15	30	9013	42.51
CURLY20	1000	16	28	8332	**58.37**	17	46	9080	66.26
CURLY30	1000	17	22	8104	**73.22**	18	55	8438	76.52
EIGENALS	930	45	60	1037	**83.33**	56	147	1249	95.91
FLETCHCR	1000	1482	1744	16774	123.46	1481	1752	16764	118.68
GENHUMPS	1000	1128	3096	25927	296.68	1263	5182	28468	303.18
GENROSE	1000	592	1234	13340	114.79	555	2910	13351	117.41
MSQRTALS	1024	46	83	20051	**2829.96**	116	881	42636	6134.60
MSQRTBLS	1024	35	56	10240	**1449.91**	*	*	*	> 18000
NCB20B	1000	20	35	2430	**216.77**	20	125	2766	263.58
SINQUAD	1000	79	147	203	4.94	85	195	212	4.71
SPARSINE	1000	19	34	5751	78.05	14	19	3236	**43.52**
VAREIGVL	1000	17	22	1618	**16.74**	33	129	9489	102.02
	Total	3485	6547	117660	3936.28	3680	11479	144742	7369.22

search algorithm is not able to locate a local minimizer of problem MSQRTBLS.

On the remaining 6 test problems where negative curvature directions have been encountered the performance of the two algorithms are not directly comparable as the two methods converge to different local minima. For these problems, in Table II we report the complete results obtained by the two algorithms where we evidenced in boldface the best optimal value obtained. As it can be observed form this table, in most cases the new algorithm is able to converge towards "better points", *i.e.*, points where the objective function value is lower.

In conclusion, on the basis of these results, the new algorithm proposed in this paper presents an overall better behaviour with respect to the one based on the curvilinear search. The main reason for this improvement appears to lie 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 MSQRTALS, MSQRTBLS and VAREIGVL). We also note that the new algorithm use most of the negative curvature directions found, which confirms our intuition that

TABLE II Comparison of the optimal objective function values ($n = 1000$)

	New algorithm					*Curvilinear search algorithm*				
Problem	*NG*	*NF*	*CG-it*	*Time*	*F*	*NG*	*NF*	*CG-it*	*Time*	*F*
BROYDN7D	55	107	2260	18.67	**3.8411*E* + 02**	45	246	1503	13.33	5.4307*E* + 02
CHAINWOO	445	803	19589	183.42	**1.6596*E* + 01**	327	1435	14916	141.81	2.1171*E* + 02
FREUROTH	13	26	50	0.96	1.2147*E* + 05	16	46	96	1.45	**1.2136*E* + 05**
NONCVXUN	230	498	15500	146.88	2.3346*E* + 03	124	852	11477	104.62	**2.3280*E* + 03**
NONCVXU2	250	546	9446	99.03	**2.3186*E* + 03**	145	1059	6268	64.18	2.3193*E* + 03
SPMSRTLS	14	22	325	4.27	**4.4189*E* − 16**	277	1116	19442	255.21	3.2814*E* + 00

these directions should be exploited when present (see Tabs. III and IV reported in the Appendix). We finally note that other tests using values of τ other than 2 did not prove to be numerically as effective.

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] Bongartz, I., Conn, A. R., Gould, N. I. M. and Toint, Ph. L. (1995). CUTE: Constrained and unconstrained testing environment. *ACM Transaction on Mathematical Software*, **21**, 123–160.
[2] Cullum, J. K. and Willoughby, R. A., *Lanczos algorithms for large symmetric eigenvalue computations*, Birkhauser, Boston, 1985.
[3] Dembo, R. S. and Steihaug, T. (1983). Truncated-Newton algorithms for large-scale unconstrained optimization. *Mathematical Programming*, **26**, 190–212.
[4] Ferris, M. C., Lucidi, S. and Roma, M. (1996). Nonmonotone curvilinear line-search methods for unconstrained optimization. *Computational Optimization and Applications*, **6**, 117–136.
[5] Fletcher, R., *Practical Methods of Optimization*, John Wiley and Sons, New York, 1987.
[6] Fletcher, R. and Freeman, T. L. (1977). A modified Newton method for minimization. *Journal of Optimization Theory and Applications*, **23**(3), 357–372.
[7] Forsgren, A. and Murray, W. (1993). Newton methods for large-scale linear equality-constrained minimization. *SIAM Journal on Matrix Analysis and Applications*, **14**, 560–587.

[8] Forsgren, A. and Murray, W. (1997). Newton methods for large-scale linear inequality-constrained minimization. *SIAM Journal on Optimization*, **7**, 162–176.
[9] Goldfarb, D. (1980). Curvilinear path steplength algorithms for minimization which use directions of negative curvature. *Mathematical Programming*, **18**, 31–40.
[10] Golub, G. H. and Van Loan, C. F., *Matrix Computations*, The John Hopkins Press, Baltimore, 1989. Second edition.
[11] Gould, N. I. M., Lucidi, S., Roma, M. and Toint, Ph. L. (1999). Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, **9**, 504–525.
[12] Grippo, L., Lampariello, F. and Lucidi, S. (1989). A truncated Newton method with nonmonotone linesearch for unconstrained optimization. *Journal of Optimization Theory and Applications*, **60**, 401–419.
[13] Lucidi, S., Rochetich, F. and Roma, M. (1998). Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization. *SIAM Journal on Optimization*, **8**, 916–939.
[14] Lucidi, S. and Roma, M. (1997). Numerical experiences with new truncated Newton methods in large scale unconstrained optimization. *Computational Optimization and Applications*, **7**, 71–87.
[15] McCormick, G. P. (1977). A modification of Armijo's step-size rule for negative curvature. *Mathematical Programming*, **13**, 111–115.
[16] Moré, J. J. and Sorensen, D. C. (1979). On the use of directions of negative curvature in a modified Newton method. *Mathematical Programming*, **16**, 1–20.
[17] Mukai, H. and Polak, E. (1978). A second-order method for unconstrained optimization. *Journal of Optimization Theory and Applications*, **26**, 501–513.
[18] Shultz, G. A., Schnabel, R. B. and Byrd, R. H. (1985). A family of trust-region-based algorithms for unconstrained minimization. *SIAM Journal on Numerical Analysis*, **22**, 47–67.
[19] Stoer, J., Solution of large linear systems of equations by conjugate gradient type methods. In: Bachem, A., Grötschel, M. and Korte, B. Editors, *Mathematical Programming. The State of the Art*, pp. 540–565, Berlin Heidelberg, 1983. Springer-Verlag.
[20] Toint, Ph. L., Towards an efficient sparsity exploiting Newton method for minimization. In: Duff, I. S. Editor, *Sparse Matrices and Their Uses*, pp. 57–88, London, 1981. Academic Press.

APPENDIX

In this appendix we report the complete results of the numerical experience. The results are reported in Tables III and IV 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.

TABLE III Results for the new algorithm

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	1.52	$4.8174E-19$	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
FLETCHCR	1000	1482	1744	16774	123.46	$3.4122E-15$	2	2
FMINSURF	1024	37	318	15937	142.77	$1.0000E+00$	0	0
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
MSQRTBLS	1024	35	56	10240	1449.91	$2.5317E-17$	13	13
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
NONDIA	1000	7	7	8	0.26	$5.3285E-12$	0	0
NONDQUAR	1000	108	427	127227	363.23	$6.9441E-09$	0	0
POWER	1000	32	32	776	2.88	$1.3384E-09$	0	0
SCHMVETT	1000	8	8	47	0.98	$-2.9940E+03$	0	0
SINQUAD	1000	79	147	203	4.94	$3.4971E-08$	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
SROSENBR	1000	9	9	11	0.16	$2.9456E-18$	0	0
TESTQUAD	1000	14	14	1022	3.05	$6.3707E-15$	0	0
TRIDIA	1000	12	12	586	1.67	$4.9780E-15$	0	0
VAREIGVL	1000	17	22	1618	16.74	$7.8694E-10$	2	2

TABLE IV Results for the curvilinear search 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

TABLE IV (Continued)

Problem	*n*	*NG*	*NF*	*CG-it*	*Time*	*F*	*d used*	*d found*
DQRTIC	1000	31	31	30	0.34	$2.7446E-07$	0	0
EIGENALS	930	56	147	1249	95.91	$4.1574E-14$	16	16
FLETCHCR	1000	1481	1752	16764	118.68	$9.4380E-15$	2	2
FMINSURF	1024	37	318	15937	142.05	$1.0000E+00$	0	0
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
MSQRTBLS	1024	*	*	*	> 18000	*	*	*
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
NONDIA	1000	7	7	8	0.24	$5.3285E-12$	0	0
NONDQUAR	1000	108	430	127227	363.11	$1.7062E-08$	0	0
POWER	1000	32	32	776	2.78	$1.3384E-09$	0	0
SCHMVETT	1000	8	8	47	0.94	$-2.9940E+03$	0	0
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
SROSENBR	1000	9	9	11	0.14	$2.9456E-18$	0	0
TESTQUAD	1000	14	14	1022	3.03	$6.3707E-15$	0	0
TRIDIA	1000	12	12	586	1.66	$4.9780E-15$	0	0
VAREIGVL	1000	33	129	9489	102.02	$2.1388E-10$	8	8