International Journal of Pure and Applied Mathematics
Volume 83 No. 2 2013, 349-364
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
doi: http://dx.doi.org/10.12732/ijpam.v83i2.13

NONMONOTONE CONVERGENCE AND RELAXING FUNCTIONS

Melisa Hendrata[1] §, P.K. Subramanian[2]

[1,2]Department of Mathematics
California State University
Los Angeles, 5151, State University Drive, Los Angeles, CA 90032, USA

Abstract: In the minimization of real valued functions, Newton's algorithm is often combined with a line search method. Grippo et al [SIAM J. Numer. Anal., Vol. 23, No. 4] first suggested a nonmonotone stepsize selection rule based on the maximum of a fixed set of previous function values. In this paper we introduce the notion of relaxing functions and suggest several other nonmonotone procedures using a modified Newton direction. Computational performance on several standard test problems is presented, which shows that the proposed models are viable alternatives.

AMS Subject Classification: 90, 90-08
Key Words: Newton's method, Armijo line search, nonmonotone line search, global optimization

1. Introduction

We are concerned in this paper with the study of the minimization problem

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

where $f : \mathbb{R}^n \to \mathbb{R}$. Given $x_0 \in D \subset \mathbb{R}^n$, most algorithms generate a sequence of points,

$$x_{k+1} = x_k + \lambda_k p_k,\ k \geq 0 \tag{1.1}$$

where λ_k is the stepsize chosen along the direction p_k [1]. If $f(x)$ is differentiable

Received: November 24, 2012

§Correspondence author

and $p_k = p(x_k)$ is a *descent direction* ($\nabla f(x_k)^T p_k < 0$) then $f(x)$ decreases in a neighborhood of x_k along p_k. The sequence $\{f(x_k)\}$ so generated may not necessarily be decreasing, but if $f(x)$ has a minimizer x^*, then usually $x_k \to x^*$ under appropriate conditions.

In case $f(x)$ is twice continuously differentiable, the celebrated Newton's algorithm [1] is defined by

$$\begin{aligned} x_{k+1} &= x_k + p_k, \ k \geq 0 \\ p_k &= -\{\nabla^2 f(x_k)\}^{-1} \nabla f(x_k) \end{aligned} \tag{1.2}$$

assuming $\nabla f(x_k) \neq 0$, the algorithm terminating otherwise. In particular $\lambda_k = 1$ in (1.1). Of course (1.2) is well defined only if the Hessian $\nabla^2 f(x_k)$ is invertible, and for $p(x_k)$ to be a descent direction, it must be positive definite. It is important to note that the sequence $\{f(x_k)\}$ of function values need not be decreasing. If $f(x)$ has a minimizer x^* in D, $\nabla f(x^*) = 0$ and if $\nabla^2 f(x^*)$ is positive definite, then as is well known, Algorithm (1.2) guarantees (with additional strong conditions) convergence of the sequence $\{x_k\}$ to x^* quadratically.

However when the Hessian is not invertible, Algorithm (1.2) fails. In most applications, however, $f(x)$ is convex and the Hessian is at least positive semidefinite. In such cases the Hessian can be modified as follows [2, 3]: Given $x \in D$, let $E(x)$ be an $n \times n$ continuous matrix such that

$$E(x) \text{ is } \begin{cases} \text{positive definite,} & \text{if } \nabla^2 f(x) \text{ is not positive definite;} \\ 0, & \text{otherwise.} \end{cases} \tag{1.3}$$

We now define $A(x_k)$ as

$$A(x_k) = \nabla^2 f(x_k) + E(x_k). \tag{1.4}$$

If $x^* \in D$ is a minimizer of $f(x)$, and $\nabla^2 f(x^*)$ is positive definite, then $A(x_k)$ coincides with the Hessian in a neighborhood of x^*.

For a large class of functions the *Modified Newton Algorithm*

$$x_{k+1} = x_k - \{A(x_k)\}^{-1} \nabla f(x_k), \tag{1.5}$$

is well defined at all points. With these modifications, one can show that the usual quadratic convergence associated with Newton's method is still preserved [2] under appropriate conditions.

The bane of Newton's algorithm, including the modification suggested in equation (1.5) above, is that convergence is essentially local. To insure one has global convergence, Algorithm (1.5) is modified as

$$x_{k+1} = x_k - \lambda_k \{A(x_k)\}^{-1} \nabla f(x_k), \tag{1.6}$$

where the stepsize λ_k is not always one, but chosen carefully by using one of several line search methods. In the well known *Armijo Line Search Algorithm* [2, 3, 4], given any descent direction $\{p_k\}$, we insure that

$$\begin{aligned} \text{AI}: \quad & f(x_{k+1}) < f(x_k) + \delta \lambda_k \nabla f(x_k)^T p_k, \\ & \delta \in (0,1),\ \bar{\lambda} > 0,\ \lambda_k = \max_{j \geq 0} \left\{ 2^{-j} \bar{\lambda} \right\}, \end{aligned} \tag{1.7}$$

is satisfied. AI is referred to as the *Armijo inequality* and it insures that $\{f(x_k)\}$ is monotonically decreasing. Under appropriate conditions, every limit point of $\{x_k\}$, if it exists, is a stationary point of f. However, the monotonicity has problems of its own in that in some difficult problems where the trajectory lies in a valley, the values of the stepsize λ_k can be very small leading to a zigzag trajectory and even end in failure.

In their seminal paper [5], Grippo et al. showed that the monotonicity of $\{f(x_k)\}$ is not essential (Newton's algorithm need not produce monotone iterates) and may be relaxed. Their work on *nonmonotone convergence* has been the inspiration for considerable literature on the subject. Given the descent direction p_k, and $\{x_k\}$ defined by (1.1), $\delta, \bar{\lambda}$, and λ_k as in (1.7), they suggest replacing the Armijo inequality AI in (1.7) by

$$\begin{aligned} & f(x_{k+1}) < R(k) + \delta \lambda_k \nabla f(x_k)^T p_k, \\ & R(k) = f(x_{l(k)}) = \max\{f(x_{k-j}) : 0 \leq j \leq m(k)\}, \end{aligned} \tag{1.8}$$

where $M \in Z^+, m(0) = 0,\ 0 \leq m(k) \leq \min[m(k-1)+1, M],\ k \geq 1$.

Notice that if $M = 0$, then (1.8) reverts to the usual Armijo algorithm (1.7). The authors show that under appropriate conditions, one can arrive at the same conclusions as in Armijo algorithm.

The aim of this paper is to present other choices for $R(k)$ and prove convergence of the algorithm that corresponds to (1.8) in each case. This is done in Section 2. In Section 3, we describe our computational experience with the choices for $R(k)$ suggested here. Our results show that each of the procedures suggested in this paper may be used with advantage. In Section 4 we comment on our experience with the algorithms and also point out several other possibilities for $R(k)$.

We follow the conventions established in [2, 3]. In particular, we use the Euclidean norm on $\mathbb{R}^n$. Real valued functions are denoted by lower case letters. We write f_k for $f(x_k), f_*$ for $f(x^*),\ p_k$ for $p(x_k)$,
A_k for $A(x_k)$, etc. If $\nabla f(x)$ is Lipschitz in an open convex set D with constant B (written $\nabla f(x) \in \mathrm{Lip}_B(D)$), an important consequence of Taylor's theorem is the inequality

$$\forall\, x,\, y \in D,\ |f(y) - f(x) - \nabla f(x)(y-x)| \le \tfrac{1}{2}\, B\, \|y-x\|^2, \tag{1.9}$$

sometimes referred to as the *Quadratic Bounded Lemma.* Finally, we indicate the end of a proof by □.

2. Generalized Armijo Line Search Algorithm

The function $R(k)$ in equation (1.8) is an example of what we shall call in the sequel as a *relaxing function*, that is, a function that relaxes the monotonicity condition on $\{f_k\}$ in the (monotone) Armijo Algorithm. In this paper, we shall be mainly concerned with the following three relaxing functions. Other choices are suggested in Section 4.

$$1.R(k) = C(k) = \sum_{j=0}^{k} \beta_j\, f_j,\ \beta_j \ge 0,\ \sum_{0}^{k} \beta_j = 1 \tag{2.1}$$

$$2.R(k) = G(k) = \prod_{j=0}^{k} {f_j}^{\beta_j},\ \beta_j \ge 0,\ \sum_{0}^{k} \beta_j = 1, \tag{2.2}$$

$$\text{and } \exists\, \gamma > 0, f_j > \gamma, \forall\, j$$

$$3.R(k) = f(x_{m(k)}) = \text{median}\ \{f(x_{k-j}) : 0 \le j \le M-1\}, \tag{2.3}$$

$$M \text{ odd},\ k \ge M-1.$$

2.1. Remarks

1. It is assumed that in the case of the first two relaxing functions (2.1)-(2.2), $f(x_1)$ is computed using the monotone Armijo algorithm while in the last case $f(x_1), \ldots, f(x_{M-1})$ are computed similarly.

2. A version of $C(k)$ occurs in [6] while other variations of this idea have been suggested in [7]. The reader would recognize that $C(k)$ is the *generalized arithmetic mean* or *convex linear combination* of the iterates $f_0, \ldots, f_k$.

In the special case when $\beta_j = \frac{1}{k+1}$ for all j, one gets the usual arithmetic mean.

3. In analogy with $C(k)$, we call $G(k)$ the *generalized geometric mean.* Here $\forall\, j,\ \beta_j = \frac{1}{k+1}$ gives rise to the well known geometric mean

$$G = (\prod_{j=0}^{k} f_j)^{1/k+1}.$$

However, unlike $C(k)$, $G(k)$ is defined and meaningful only if $f_j > 0$ for all j. In fact if $f_j = 0$ for some j the algorithm terminates since $G(k) = 0$ for all $k \geq j$. To avoid this situation, one can add a large enough constant K to $f(x)$ to begin with so that $f_j > \gamma$ for some $\gamma > 0$.

4. We note that in (2.3), $R(k) = f(x_{m(k)})$, $M = 1$ gives the usual Armijo algorithm. It would also appear that $f(x_{l(k)}) = f_{l(k)}$ [5] suffers from the fact that it disregards better (smaller) values that may occur during the iteration. Both $C(k)$ and $G(k)$ require that the entire sequence of function values be used. Unlike the case of $f(x_{l(k)})$, the advantage of using $f(x_{m(k)}) = f_{m(k)}$ is that all function values greater than the median value are not considered. However, both $f_{l(k)}$ and $f_{m(k)}$ depend on the value of M which may influence the performance of the algorithms in such cases.

We shall call any Armijo line search algorithm that uses a relaxing function as a *Generalized Armijo Algorithm (GAI).* As we shall see later, the crucial properties of a relaxing function $R(k)$, usually needed in convergence proofs are:

$$f_{k+1} < R_k \Rightarrow (i)\ f_{k+1} \leq R_{k+1},\quad (ii)\ R_{k+1} \leq R_k. \tag{2.4}$$

Theorem 2.1. *All the relaxing functions defined in* (2.1)-(2.3) *satisfy* (2.4).

Proof. We assume $f_{k+1} < R_k$ in all three cases.

Case 1. $R(k) = C(k)$

We shall find it convenient to define $C(k) = C_k$ recursively by the formula

$$C_{k+1} = \frac{1}{1+\alpha_k}\{\alpha_k C_k + f_{k+1}\},\ \alpha_k \geq 0. \tag{2.5}$$

Here α_k represents the proportional weight given to C_k and can be varied through the iterations. Notice that the choice $\alpha_k = 0$ converts GAI to the standard Armijo algorithm. Since $f_{k+1} < C_k$,

$$C_{k+1} = \frac{\alpha_k C_k + f_{k+1}}{1+\alpha_k} < \frac{1}{1+\alpha_k}\{\alpha_k C_k + C_k\} = C_k,$$
$$C_{k+1} = \frac{\alpha_k C_k + f_{k+1}}{1+\alpha_k} > \frac{1}{1+\alpha_k}\{\alpha_k f_{k+1} + f_{k+1}\} = f_{k+1},$$

showing that $\{C_k\} \downarrow$ and that $C_{k+1} > f_{k+1}$.

Case 2. $R(k) = G(k)$

In analogy with the last case, we shall define $G(k) = G_k$ recursively by

$$G_{k+1} = \{{G_k}^{\alpha_k} f_{k+1}\}^{1/(1+\alpha_k)},\ \alpha_k \geq 0. \tag{2.6}$$

Here too $\alpha_k = 0$ converts GAI to the standard Armijo algorithm AI. By taking logs in (2.6), we get

$$\ln G_{k+1} = \frac{1}{1+\alpha_k}\{\alpha_k \ln G_k + \ln f_{k+1}\}.$$

Analogous to the last case, the use of $f_{k+1} < G_k$, now gives that

$$\ln G_{k+1} < \ln G_k, \text{ and } \ln G_{k+1} > \ln f_{k+1},$$

from which we get that $\{G_k\} \downarrow$ and that $G_{k+1} > f_{k+1}$.

Case 3. $R(k) = f(x_{m(k)})$

At the end of k iterations we have $M = 2s+1$ elements $x(k), x(k-1), \ldots, x(k-M+1)$. We relabel these as $x(0)', x(1)', \ldots, x(s)', x(s+1)', x(s+2)', \ldots, x(2s)'$ and list the corresponding function values into the groups A and B so that

$$\underbrace{f_{(2s)'} \leq \ldots \leq f_{(s+1)'}}_{A} \leq f_{(s)'} \leq \underbrace{f_{(s-1)'} \leq \ldots \leq f_{(0)'}}_{B}.$$

In particular, $f_{m(k)} = f_{(s)'}$. At the end of $k+1$ iterations, f_{k-M+1} is removed and f_{k+1} added to A since $f_{k+1} < f_{m(k)} = f_{(s)'}$.

If $f_{k-M+1} \in A$ then its removal and the addition of f_{k+1} to A causes no change to B and the new median is the same as the old median, that is $f_{m(k+1)} = f_{m(k)}$.

If $f_{k-M+1} \in B$, after its removal there are only $(s-1)$ elements in B. In this case $f_{(s)'}$ moves to B and the new median is $f_{m(k+1)} = \max\{f_{k+1}, f_{(s+1)'}\}$. It is clear that $f_{k+1} \leq f_{m(k+1)} \leq f_{m(k)}$.

If $f_{k-M+1} = f_{m(k)} = f_{(s)'}$ then after its removal, the new median is defined just as in the last case, that is, $f_{m(k+1)} = \max\{f_{k+1}, f_{(s+1)'}\}$, so that $f_{k+1} \leq f_{m(k+1)} \leq f_{m(k)}$. □

Algorithm 2.1. (Generalized Armijo Algorithm) Let

(i) $f : \mathbb{R}^n \to \mathbb{R}$ and D an open convex subset of $\mathbb{R}^n$. Let f be continuously differentiable on D. Given $x_0 \in D$, let the level set $S = \{x | f(x) \leq f(x_0)\}$ be compact with $S \subset D$ with $\nabla f \in \mathrm{Lip}_K(D)$.

(ii) $\delta \in (0,1)$, $\bar{\lambda} \in (0,1]$.

(iii) $\{x_k\}_{k\geq 0}$ be defined as follows:

(a) If $\nabla f_k = 0$, stop; else $x_{k+1} := x_k + \lambda_k p_k$,
where p_k is a descent direction and $\lambda_k = \max_{j\geq 0}\left\{2^{-j}\bar{\lambda}\right\}$ satisfies the *Generalized Armijo inequality*:

$$\text{GAI} \ : f_{k+1} < R(k) + \delta\, \lambda_k\, \nabla f_k^T p_k, \tag{2.7}$$

and where $R(k)$ is any of the relaxing functions defined in (2.1)-(2.3).

(iv) Let $c_1, c_2 > 0$, so that $\nabla f_k^T p_k \leq -c_1 \|\nabla f_k\|^2$, $\|p_k\| \leq c_2 \|\nabla f_k\|$.

The following theorem shows convergence of Algorithm 2.1 for any of the three choices for R_k given in equations (2.1)-(2.3).

Theorem 2.2. (GAI Convergence Theorem) *Algorithm 2.1 is well defined, $\{x_k\} \subset S$, and if x^* is a limit point of $\{x_k\}$, then $\nabla f(x^*) = 0$.*

Proof. We first prove that Algorithm 2.1 is well defined. By the Quadratic Bounded Lemma,

$$f(x_k + \lambda_k p_k) \leq f_k + \lambda_k \nabla f_k^T p_k + \tfrac{1}{2} K \lambda_k^2 \|p_k\|^2.$$

Hence, there exists $\lambda > 0$ such that

$$\lambda \leq \frac{2(1-\delta)[-\nabla f_k^T p_k]}{K\|p_k\|^2}, \tag{2.8}$$

satisfying (1.7). Since $f_k \leq R_k$ (Theorem 2.1), for all the relaxing functions in (2.1)-(2.3), we can find $\lambda_k = \max_{j\geq 0} 2^{-j}\bar{\lambda}$ so that

$$f(x_k + \lambda_k p_k) < R(k) + \lambda_k \delta[\nabla f_k^T p_k], \tag{2.9}$$

showing that (2.7) is satisfied and the algorithm is well defined.

Next we show $\{x_k\} \subset S$. If $R(k) = C_k$ or G_k, we have $\{R(k)\} \downarrow$ and $R(0) = f_0$. Hence, by Theorem 2.1, $f_k \leq R(k) \leq f_0$ so that $x_k \in S$. In case 3, $R(k) = f(x_{m(k)})$, $k \geq M-1$, and by assumption $\{x_k\}_{k=1}^{M-1}$ are computed by the standard Armijo algorithm. In particular, $f_0 > f_1 > \ldots > f_{M-1}$ and the median $f(x_{m(M-1)}) = f(x_s)$ where $M = 2s+1$. Since $\{f_{m(k)}\}_{(k\geq M-1)}$ is non-increasing, it follows that $f_{k+1} < R(k) = f_{m(k)} \leq f_{m(M-1)} < f_0$ and hence $x_{k+1} \in S$ for $k \geq M-1$. Since $f_i < f_0,\ 1 \leq i \leq M-1$, clearly $x_k \in S$ for all k.

We claim that the theorem is proved if we show

$$\lambda_k \nabla f_k^T p_k \to 0. \tag{2.10}$$

in all the three cases.

To see this, note that either $\lambda_k = \bar{\lambda}$, or $2\lambda_k$ violates (2.8). Hence,

$$\lambda_k = \bar{\lambda} \quad \text{or} \quad \lambda_k > \frac{(1-\delta)[-\nabla f_k^T p_k]}{K\|p_k\|^2}. \tag{2.11}$$

In the first case, $|\lambda_k \nabla f_k^T p_k| = \bar{\lambda}|\nabla f_k^T p_k| \geq c_1 \bar{\lambda}\|\nabla f_k\|^2$. Hence, (2.10) implies $\|\nabla f_k\| \to 0$.

In the second case, we have

$$\lambda_k \nabla f_k^T p_k > \frac{-(1-\delta)(\nabla f_k^T p_k)^2}{K\|p_k\|^2} \to 0. \tag{2.12}$$

By (iv) in Algorithm 2.1, $[-\nabla f_k^T p_k]/\|p_k\| \geq (c_1/c_2)\|\nabla f_k\|$. Hence, $\{p_k\}$ is gradient related, that is

$$\frac{[-\nabla f_k^T p_k]}{\|p_k\|} \geq \sigma(\|\nabla f_k\|)$$

for some forcing function $\sigma(x)$ [1], and from (2.12), $\sigma(\|\nabla f_k\|) \to 0$. Hence, $\|\nabla f_k\| \to 0$.

In both cases, if x^* is a limit point of $\{x_k\}$ then $\nabla f(x^*) = 0$.

We proceed now to prove that (2.10) is satisfied for all choices of $R(k)$ in (2.1)-(2.3).

Case 1. $R(k) = C(k)$. From (2.5) and GAI,

$$\begin{aligned} C_{k+1} &< \frac{1}{(1+\alpha_k)}\{\alpha_k C_k + C_k + \delta\lambda_k[\nabla f_k^T p_k]\} \\ &= C_k + \frac{1}{(1+\alpha_k)}[\delta\lambda_k \nabla f_k^T p_k] \end{aligned}$$

so that

$$C_k - C_{k+1} \geq \frac{1}{(1+\alpha_k)}[-\delta\lambda_k \nabla f_k^T p_k]$$

Since $\{C_k\}$ converges it follows $\lambda_k \nabla f_k^T p_k \to 0$, that is, equation (2.10) is satisfied.

Case 2. $R(k) = G(k)$. Since $f_k > \gamma$ it follows that $G_k > \gamma$. Since $\{G_k\}$ is decreasing it follows that $G_k \to G^* > \gamma$. From (2.6),

$$\begin{aligned} {G_{k+1}}^{1+\alpha_k} &= {G_k}^{\alpha_k} f_{k+1} \\ &< {G_k}^{\alpha_k}\{G(k) + \delta\lambda_k[\nabla f_k^T p_k]\} \\ &= {G_k}^{1+\alpha_k} + \delta {G_k}^{\alpha_k}\lambda_k[\nabla f_k^T p_k], \end{aligned}$$

and taking limits on both sides,

$$(G^*)^{1+\alpha_k} \leq (G^*)^{1+\alpha_k} + \delta(G^*)^{\alpha_k} \lim_{k\to\infty} \lambda_k[\nabla f_k^T p_k],$$

that is,

$$0 \leq \lim_{k\to\infty} \lambda_k[\nabla f_k^T p_k] \leq 0$$

showing that (2.10) holds and we are done.

Case 3. $R(k) = f_{m(k)},\ k \geq M-1$.

$$\|x_{L(k)} - x_{k+1}\| \to 0.$$

Set $L(k) = m(k+M+2)$. Since $f_{m(k)} = \text{median}\{f_k, f_{k-1}, \ldots, f_{k-(M-1)}\}$, we have $k - M + 1 \leq m(k) \leq k$. Also $(L(k) - k - 1) = m(k+M+2) - k - 1 \leq (k+M+2) - k - 1 = M+1$. For any k,

$$\begin{aligned} x_{L(k)} &= x_{L(k)-1} + \lambda_{L(k)-1}\, p_{L(k)-1} \\ &= x_{L(k)-2} + \lambda_{L(k)-2}\, p_{L(k)-2} + \lambda_{L(k)-1}\, p_{L(k)-1} \\ &= \ldots = x_{k+1} + \sum_{j=1}^{L(k)-k-1} \lambda_{L(k)-j}\, p_{L(k)-j}, \end{aligned}$$

which follows that

$$\|x_{L(k)} - x_{k+1}\| \leq \sum_{j=1}^{M+1} \lambda_{L(k)-j} \|p_{L(k)-j}\|. \tag{2.13}$$

Thus, if we prove that

$$\lambda_{L(k)-j} \|p_{L(k)-j}\| \to 0, \quad 1 \leq j \leq M+1, \tag{2.14}$$

we are done. We use induction. From GAI,

$$f(x_{m(k)}) \; < \; f(x_{m(m(k)-1)}) + \delta \lambda_{m(k)-1} \nabla f_{m(k)-1}^T p_{m(k)-1}.$$

Since $\{f(x_{m(m(k)-1)})\} \subseteq \{f(x_{m(k)})\} \; \searrow$, it follows that

$$\lambda_{m(k)-1} \nabla f_{m(k)-1}^T p_{m(k)-1} \to 0$$

and from (iv) in Algorithm 2.1, we see that

$$0 = \lim_{k\to\infty} \lambda_{m(k)-1} \nabla f_{m(k)-1}^T p_{m(k)-1} \leq -\frac{c_1}{c_2^2} \lim_{k\to\infty} \lambda_{m(k)-1} \|p_{m(k)-1}\|^2 \leq 0.$$

Hence,

$$0 = \lim_{k\to\infty} \lambda_{m(k)-1} \|p_{m(k)-1}\|^2 \geq \lim_{k\to\infty} \lambda_{m(k)-1}^2 \|p_{m(k)-1}\|^2,$$

since $\lambda_{m(k)-1} \leq 1$. Since $\{L(k)\} \subset \{m(k)\}$, it follows that (2.14) is true for $j = 1$.

Assume that (2.14) is true for some j. To avoid negative indices, we assume that $k \geq j-1$. Again from GAI,

$$f_{L(k)-j} - f_{m(L(k)-(j+1))} < \delta \lambda_{L(k)-(j+1)} \nabla f_{L(k)-(j+1)}^T p_{L(k)-(j+1)}. \tag{2.15}$$

Now $m(k) \leq k \; \Rightarrow m\big(L(k)-(j+1)\big) \leq (L(k)-(j+1))$. Since $\{m\big(L(k)-(j+1)\big)\} \subset \{L(k)-j\}$, taking limits in (2.15), which implies, as before,

$$\lambda_{L(k)-(j+1)} \|p_{L(k)-(j+1)}\| = 0,$$

concluding the induction and the proof. □

3. Computational Experience

In this section we present our computational experience using Algorithm 2.1 with each of the three relaxing functions in (2.1)-(2.3) and for comparison purposes, also that of Algorithm (1.8) as in [5]. In each case, we use the modified Newton's direction $p_k = -(A_k)^{-1}\nabla f_k$, where A_k is defined by (1.4). We chose $\bar{\lambda} = 1$. Since the performances of $R(x) = f(x_{l(k)})$ and $R(x) = f(x_{m(k)})$ are dependent on the choice of M, we have tested each of them for two values of M, $M = 5$ and $M = 11$. Likewise, the performance of $R(x) = C(k)$ and $R(x) = G(k)$ are dependent on the values of the parameter α_k and we show results for several values of α_k.

Our computational results are given in Table 1 where in columns A, B, C, D we list the results obtained when $R(k) = C(k), G(k), f(x_{m(k)}), f(x_{l(k)})$ respectively. Since $R(k) = C(k)$ and $R(k) = G(k)$ depend on α_k, the value we have chosen for the latter is given in parenthesis. Likewise both $R(k) = f(x_{m(k)})$ and $R(k) = f(x_{l(k)})$ are dependent on the value of M which is given in the respective columns.

We tested our algorithms on the following well known test problems, most of which are from [8].

1. *Six-hump camelback function* $(n = 2)$

$$f(x) = x_1^2(4 - 2.1x_1^2 + (x_1^4/3)) + x_1x_2 + x_2^2(-4 + 4x_2^2)$$
$$x_0 = (-0.5, 0.2), \quad x^* = (-0.0898, 0.7126), (0.0898, -0.7126).$$

2. *Beale function* $(n = 2)$

$$f(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$$
$$x_0 = (-0.5, -0.6), \quad x^* = (3, 0.5).$$

3. *Box three-dimensional function* $(n = 3)$

$$f(x) = \sum_{i=1}^{3} f_i^2(x), \text{ where}$$
$$f_i(x) = e^{-t_ix_1} - e^{-t_ix_2} - x_3(e^{-t_i} - e^{-10t_i}), \ t_i = (0.1)i$$
$$x_0 = (0, 10, 20), \quad x^* = (1, 10, 1).$$

4. *Helical valley function* $(n = 3)$

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{3} f_i^2(x), \text{ where} \\
f_1(x) &= 10[x_3 - 10\theta(x_1, x_2)] \\
f_2(x) &= 10[(x_1^2 + x_2^2)^{1/2} - 1] \\
f_3(x) &= x_3 \\
\theta(x_1, x_2) &= \begin{cases} (1/2\pi)\arctan(x_2/x_1), & \text{if } x_1 > 0 \\ (1/2\pi)\arctan(x_2/x_1) + 0.5, & \text{if } x_1 < 0. \end{cases} \\
x_0 &= (-5, 10, -10), \quad x^* = (1, 0, 0).
\end{aligned}
$$

5. *Trigonometric function* $(n = 8)$

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{n} f_i^2(x), \text{ where} \\
f_i(x) &= n - \sum_{j=1}^{n} \cos x_j + i(1 - \cos x_i) - \sin x_i, \ i = 1, \ldots, n. \\
x_0 &= (1/n, \ldots, 1/n), \\
x^* &= (0.067, 0.070, 0.073, 0.077, 0.081, 0.240, 0.179, 0.116).
\end{aligned}
$$

6. *Variably dimensioned function* $(n = 8)$

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{n} f_i^2(x); \quad f_i(x) = x_i - 1, i = 1, \ldots, n \\
f_{n+1}(x) &= \sum_{j=1}^{n} j(x_j - 1), \ f_{n+2}(x) = \Big(\sum_{j=1}^{n} j(x_j - 1) \Big)^2 \\
x_0 &= \big(1 - (j/n)\big), j = 1, \ldots, n, \quad x^* = (1, \ldots, 1).
\end{aligned}
$$

7. *Penalty function I* $(n = 10)$

$$
\begin{aligned}
f_i(x) &= a^{1/2}(x_i - 1), \ 1 \le i \le n; \ a = 10^{-5} \\
f_{n+1}(x) &= \Big(\sum_{j=1}^{n} x_j^2 \Big) - (1/4). \\
x_0 &= (1, 2, \ldots, n) \\
x^* &= (0.16, \ldots, 0.16).
\end{aligned}
$$

8. *Penalty function II* $(n = 10)$

$$f_1(x) = x_1 - 0.2$$
$$f_i(x) = a^{1/2}\Big(\exp\Big[\frac{x_i}{10}\Big] + \exp\Big[\frac{x_{i-1}}{10}\Big] - y_i\Big),\ 2 \le i \le n$$
$$f_i(x) = a^{1/2}\Big(\exp\Big[\frac{x_{i-n+1}}{10}\Big] - \exp\Big[\frac{-1}{10}\Big]\Big),\ n < i < 2n$$
$$f_{2n}(x) = \Big(\sum_{j=1}^{n}(n-j+1)x_j^2\Big) - 1,$$
$$\text{where } a = 10^{-5} \text{ and } y_i = \exp\Big[\frac{i}{10}\Big] + \exp\Big[\frac{i-1}{10}\Big]$$
$$x_0 = (1, \ldots, 1)$$
$$x^* = (0.20, 0.02, 0.03, 0.04, 0.05, 0.08, 0.12, 0.19, 0.34, 0.36).$$

9. *Discrete boundary value function* $(n = 10)$

$$f_i(x) = 2x_i - x_{i-1} - x_{i+1} + h^2(x_i + t_i + 1)^3/2,$$
$$\text{where } h = 1/(n+1),\ t_i = ih, \text{ and } x_0 = x_{n+1} = 0;$$
$$x_0 = (-10, -2, 3, -4, 55, 6, -7, 8, -90, 10)$$
$$x^* = (-0.04, -0.08, -0.11, -0.14, -0.16,$$
$$-0.17, -0.17, -0.16, -0.13, -0.08).$$

10. *Broyden tridiagonal function* $(n = 10)$

$$f_i(x) = (3 - 2x_i)x_i - x_{i-1} - 2x_{i+1} + 1,$$
$$\text{where } x_0 = x_{n+1} = 0,$$
$$x_0 = (-10, 1, 1, 1, 1, 10, 1, 1, 1, -10),$$
$$x^* = (-0.45, -0.38, 0.02, 0.76, 1.16, 0.42, 0.31, 0.76, 1.21, 0.26)$$
$$\text{or } x^* = (1.57, 0.38, 0.17, 0.61, 1.05, 0.59, 0.42, 0.77, 1.17, 0.27).$$

4. Concluding Remarks

In this paper we have given examples of three other relaxing functions for nonmonotone convergence together with proofs of convergence of the corresponding algorithms. Our computational experience suggests all are very viable.

Test problems	**(A)**	**(B)**	**(C)**	**(D)**
Six-hump camelback	5 (0.25)	5 (0.25)	6 (5)	5 (5)
	5 (0.85)	5 (0.85)	11 (11)	5 (11)
Beale	8 (0.25)	8 (0.25)	8 (5)	8 (5)
	8 (0.85)	8 (0.85)	8 (11)	8 (11)
Box 3-dimensional	13 (0.25)	7 (0.25)	8 (5)	13 (5)
	16 (0.85)	9 (0.85)	8 (11)	16 (11)
Helical valley	20 (0.25)	16 (0.25)	14 (5)	23 (5)
	20 (0.85)	20 (0.85)	12 (11)	67 (11)
Trigonometric	12 (0.25)	12 (0.25)	12 (5)	12 (5)
	12 (0.85)	12 (0.85)	12 (11)	12 (11)
Variably dimensioned	13 (0.25)	13 (0.25)	13 (5)	13 (5)
	13 (0.85)	13 (0.85)	13 (11)	13 (11)
Penalty function I	28 (0.25)	32 (0.25)	27 (5)	23 (5)
	23 (1)	21 (1)	21 (11)	23 (11)
Penalty function II	86 (0.25)	93 (0.25)	85 (5)	65 (5)
	28 (6)	28 (6)	61 (11)	28 (11)
Discrete boundary value	17 (0.25)	17 (0.25)	17 (5)	20 (5)
	20 (0.85)	17 (0.85)	17 (11)	20 (11)
Broyden tridiagonal	19 (0.25)	19 (0.25)	19 (5)	19 (5)
	50 (0.85)	19 (0.85)	18 (11)	50 (11)

Table 1: Numerical comparisons

We should point out that besides $R(k) = f(x_{l(k)})$ and $R(k) = f(x_{m(k)})$ other possibilities exist. Given $M = 2s + 1$, consider once again the function values $\{f(x_{k-j})\}_{j=0}^{M-1}$ after iteration k, listed as in the proof for Case 3 of Theorem 2.1:

$$f_{(2s)'} \leq \ldots \leq f_{(s+1)'} \leq f_{(s)'} \leq f_{(s-1)'} \leq \cdots \leq f_{(0)'}.$$

In this listing, $R(k) = f(x_{l(k)}) = f_{(0)'}$ while $R(k) = f(x_{m(k)}) = f_{(s)'}$. *In fact, we can choose* $R(k) = f(x_{(2s-j)'})$, $0 \leq j \leq 2s$, *the function value at any arbitrary but fixed position* $(2s-j)'$ *in this list.* The proof of Theorem 2.1 for such a choice is very similar to that given in the case when $R(k) = f(x_{m(k)})$. However, we surmise that the optimal value of M as well as the optimally located function value for $R(k)$ are problem dependent.

The use of modified Newton direction is preferable to the regular Newton direction $p_k = -(\nabla^2 f_k)^{-1}\nabla f_k$ or the Cauchy direction $p_k = -\nabla f_k$ (when

the Hessian is not positive definite, as suggested in [5]). In many cases the Hessian may be indefinite and cause the algorithms to fail. This is the case in the *Beale, Six-hump camelback, Box three-dimensional* and *Trigonometric functions*, where with the given starting points, Algorithm (1.8) with regular Newton direction either converges to a saddle point or does not converge due to very small stepsizes. With the modified Newton direction, however, it converges with the number of iterations comparable to algorithms (A)-(C).

We note that in several of the test problems, such as *Beale* and *Trigonometric functions*, all four algorithms choose the maximum stepsize $\lambda_{\max} = 1$ throughout. Thus, their performances were identical regardless of α_k and M chosen.

References

[1] J.M. Ortega, W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, SIAM, Philadelphia (2000).

[2] M. Hendrata, P.K. Subramanian, Some Remarks on Newton's Algorithm, *International Journal of Pure and Applied Mathematics*, **63** (2010), 223-241.

[3] M. Hendrata, P.K. Subramanian, Some Remarks on Newton's Algorithm - II, *International Journal of Pure and Applied Mathematics*, **74** (2012), 373-391.

[4] L. Armijo, Minimization of functions having Lipschitz continuous first partial derivatives, *Pacific J. Math*, **16** (1966), 1-3.

[5] L. Grippo, F. Lampariello, S. Lucidi, A Nonmonotone Line Search Technique for Newton's Method, *SIAM Journal on Numerical Analysis*, **23** (1986), 707-716.

[6] H. Zhang, W. Hager, A Nonmonotone Line Search Technique and Its Application to Unconstrained Optimization, *SIAM Journal of Optimization*, **14** (2004), 1043-1056.

[7] S. Hu, Z. Huang, N. Lu, A Non-monotone Line Search Algorithm for Unconstrained Optimization, *Journal of Scientific Computing*, **42** (2010), 38-53.

[8] J. Moré, B. Garbow, K. Hillstrom, Testing Unconstrained Optimization Software, *ACM Transactions on Mathematical Software*, **7** (1981), 17-41.