UNIVERSITÀ DEGLI STUDI FIRENZE

FLORE
Repository istituzionale dell'Università degli Studi di Firenze

Global convergence technique for the Newton method with periodic Hessian evaluation.

Questa è la Versione finale referata (Post print/Accepted manuscript) della seguente pubblicazione:

Original Citation:

Global convergence technique for the Newton method with periodic Hessian evaluation / F. LAMPARIELLO; M. SCIANDRONE. - In: JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS. - ISSN 0022-3239. - STAMPA. - 111:(2001), pp. 341-358.

Availability:

This version is available at: 2158/256060 since:

Publisher:

Plenum Press:Book Customer Service, 233 Spring Street:New York, NY 10013:(212)620-8471, (212)620-

Terms of use:

Open Access

La pubblicazione è resa disponibile sotto le norme e i termini della licenza di deposito, secondo quanto stabilito dalla Policy per l'accesso aperto dell'Università degli Studi di Firenze (https://www.sba.unifi.it/upload/policy-oa-2016-1.pdf)

Publisher copyright claim:

(Article begins on next page)

12 October 2024

JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 111, No. 2, pp. 341–358, November 2001 (© 2001)

Global Convergence Technique for the Newton Method with Periodic Hessian Evaluation[1]

F. Lampariello[2] and M. Sciandrone[2]

Communicated by O. L. Mangasarian

Abstract. The problem of globalizing the Newton method when the actual Hessian matrix is not used at every iteration is considered. A stabilization technique is studied that employs a new line search strategy for ensuring the global convergence under mild assumptions. Moreover, an implementable algorithmic scheme is proposed, where the evaluation of the second derivatives is conditioned to the behavior of the algorithm during the minimization process and the local convexity properties of the objective function. This is done in order to obtain a significant computational saving, while keeping acceptable the unavoidable degradation in convergence speed. The numerical results reported indicate that the method described may be employed advantageously in all applications where the computation of the Hessian matrix is highly time consuming.

Key Words. Unconstrained optimization, Newton-type methods, periodic Hessian evaluation, global and superlinear convergence, computational savings.

1. Introduction

We consider the unconstrained minimization problem

$$\min_{x \in R^n} f(x), \tag{1}$$

where $f: R^n \to R$. We assume that both the gradient $g(x)$ and the Hessian matrix $H(x)$ of f exist and are continuous on R^n.

[1]The authors are indebted to Prof. Luigi Grippo, Dipartimento di Informatica e Sistemistica, University of Rome—La Sapienza, for helpful advice and suggestions.

[2]Researcher, Istituto di Analisi dei Sistemi ed Informatica, National Research Council, Rome, Italy.

0022-3239/01/1100-0341$19.50/0 © 2001 Plenum Publishing Corporation

As is well known, the Newton method for determining a local solution of problem (1) is defined by the iteration

$$x_{k+1} = x_k - H(x_k)^{-1} g(x_k), \qquad k = 0, 1, \ldots,$$

provided that $H(x_k)$ is nonsingular.

Because of its fast rate of local convergence, typically quadratic, several strategies have been devised for globalizing the Newton method, i.e., for forcing convergence from poor starting points into a neighborhood of a local minimizer. Among them, well-known approaches are the line search and trust region methods, which lead to modifications of the pure Newton iteration that represent some of the most reliable and powerful methods for solving unconstrained optimization problems.

In the design of practical implementations of the Newton method, the need for the Hessian matrix represents the main drawback, since the explicit evaluation of the second-order derivatives is sometimes an expensive process requiring excessive computation time. For this reason, the possibility has been considered in the literature of defining a Newton-type method that avoids the Hessian evaluation at every iteration.

In particular, let us consider the following iterative scheme:

$$\begin{aligned} x_{k+1} = x_k &- H(x_k)^{-1} g(x_k) \\ &- H(x_k)^{-1} g(x_k - H(x_k)^{-1} g(x_k)), \qquad k = 0, 1, \ldots, \end{aligned} \tag{2}$$

where a Newton step is followed by another Newton step calculated using the "old" Hessian matrix.

It has been shown (Ref. 1) that, if f is twice Lipschitz continuously differentiable, the combined iteration (2) converges locally with at least cubic convergence rate.

The extension of (2) to the case where a finite number $m \geq 1$ of approximated Newton steps (i.e., calculated using the same Hessian matrix) are composed with the initial Newton step is known as the Shamanskii modification (Refs. 2–4) of the Newton method,

$$\begin{aligned} x_{k+1} = x_k &- H(x_k)^{-1} g(x_k) \\ &- H(x_k)^{-1} \sum_{j=1}^{m} g(x_k + d_j(x_k)), \qquad k = 0, 1, \ldots, \end{aligned} \tag{3}$$

where

$$\begin{aligned} d_1(x_k) &= -H(x_k)^{-1} g(x_k), \\ d_j(x_k) &= d_{j-1}(x_k) - H(x_k)^{-1} g(x_k + d_{j-1}(x_k)), \qquad j = 2, \ldots, m. \end{aligned}$$

It has been shown (Ref. 1) that even the iterate (3) converges locally and has convergence of order $m + 2$.

This modification of the Newton method may be considerably more efficient computationally than other Newton-type iterative schemes, since the Hessian matrix is recomputed only every $m+1$ steps of the total iterations. This reduction in the computational burden is achieved obviously at the expense of a degradation in speed of convergence, which however may be small or acceptable at any rate. Therefore, the use of this strategy may be advantageous in all applications where the computation of the second-order derivatives is very expensive.

As far as we are aware, techniques for ensuring the global convergence of the Shamanskii modification of the Newton method have not yet been studied.

In this paper, by adopting the line search approach, we present a stabilization technique for the Shamanskii method, that uses a new line search algorithm for ensuring the convergence under mild assumptions. We prove that this line search strategy allows one to obtain superlinear convergence rate. Then, with the aim of keeping the overall computational cost as small as possible, we propose an implementable algorithmic scheme, where the reevaluation of the Hessian matrix is decided in an adaptive manner, i.e., on the basis of the behavior of the algorithm during the minimization process and of the local convexity properties of the objective function. Finally, we report the numerical results obtained by solving a set of standard test problems and those relative to a specific application, which is the solution of neural network training problems.

2. Stabilization Algorithm for the Shamanskii Method

We consider the sequence of points generated by performing a line search along every direction defined by the Shamanskii method (3). In order to consider these points explicitly, we adopt the notation used in Ref. 3, i.e., the iterative scheme

$$x_{k+1} = x_k - \alpha_k D_k g(x_k), \qquad k = 0, 1, \ldots, \tag{4}$$

where

$$D_k = D_{ip+j} = \hat{H}(x_{ip})^{-1}, \qquad j = 0, 1, \ldots, p-1 \text{ and } i = 0, 1, \ldots.$$

The matrix $\hat{H}(x_{ip})$ denotes the Hessian or a suitable modification of it, which is recomputed every p iterations, and the stepsize α_k along the search direction $d_k = -D_k g(x_k)$ is computed by means of the line search algorithm described later.

As regards d_k, we assume obviously that

$$g(x_k)^T d_k < 0, \qquad \text{for all } k.$$

Moreover, denoting by K_N the subset of iterates where the Hessian matrix is computed, we assume that:

(a) for all $k \in K_N = \{ip\}$, $i = 0, 1, \ldots,$

$$\|d_k\| \geq \sigma(|g(x_k)^T d_k| / \|d_k\|),$$

where $\sigma: R^+ \to R^+$ is a forcing function, i.e., such that $\lim_{k\to\infty} \sigma(t_k) = 0$ implies $\lim_{k\to\infty} t_k = 0$;

(b) for every infinite subset $K \subseteq K_N$,

$$\lim_{k\to\infty,\, k\in K} [|g(x_k)^T d_k| / \|d_k\|] = 0 \text{ implies } \lim_{k\to\infty,\, k\in K} \|g(x_k)\| = 0.$$

Essentially, these conditions impose that the subsequence $\{d_k\}_{k\in K_N}$ be gradient-related to $\{x_k\}_{k\in K_N}$. In particular, condition (a) holds by assuming some reasonable boundedness condition on the matrix $\hat{H}(x_{ip}) = D_k^{-1}$. In this case, since D_k^{-1} is not changed outside K_N, condition (a) imposed on K_N will also be satisfied for all k. In order to ensure that condition (b) be satisfied, it is possible to employ a modification of the Cholesky factorization of the Hessian matrix. However, to obtain that the global convergence conditions be consistent with those of the superlinear convergence rate, the Hessian matrix should not be modified in a neighborhood of a strong local minimum. This can be achieved, for instance, by means of the modified Cholesky factorization described in Ref. 3. By this factorization process, the factors are related to the gradient at the current point x_{ip} in such a way that either $\hat{H}(x_{ip})$ is uniformly nonsingular or, whenever it tends to become singular, $\|g(x_{ip})\| \to 0$. Therefore, condition (b) is satisfied. Note that this does not imply that it holds outside K_N, i.e., that the whole sequence $\{d_k\}$ is gradient-related to $\{x_k\}$. Indeed, if the subsequence $\{x_k\}_{k\in K_N}$ converges toward a limit point where the Hessian is singular, the modification performed for $k \in K_N$ does not ensure that the modified Hessian will tend to a positive-definite matrix.

Then, under conditions (a) and (b), if the stepsize α_k is computed by means of a standard line search for ensuring

$$|g(x_k)^T d_k| / \|d_k\| \to 0,$$

we have that

$$\|g(x_k)\| \to 0$$

only for subsequences in K_N.

We remark that a possible way for obtaining a stronger global convergence result together with the superlinear convergence rate is to modify D_k^{-1} outside K_N in such a way that condition (b) holds for all k. However, this would reduce the advantage of computing the Hessian and its factorization only periodically. Therefore, we prefer to adopt a different approach, which takes into account the following result.

Proposition 2.1. Let $\{x_k\}$ be a sequence generated by the scheme (4). Assume that

$$\lim_{k \to \infty,\, k \in K_N} [|g(x_k)^T d_k| / \|d_k\|] = 0 \tag{5}$$

and that

$$\lim_{k \to \infty} \|x_{k+1} - x_k\| = 0. \tag{6}$$

Then, every limit point of $\{x_k\}$ is a stationary point of f.

Proof. By contradiction, let us assume that there exists an infinite subset $K \subseteq \{0, 1, \ldots\}$ such that $\{x_k\}_{k \in K} \to \hat{x}$ and

$$\|g(\hat{x})\| \geq \eta > 0. \tag{7}$$

For each $k \in K$, let $l(k) \in [0, p-1]$ be the integer such that $k + l(k) \in K_N$. Then, since

$$\|x_{k+l(k)} - x_k\| \leq \|x_{k+l(k)} - x_{k+l(k)-1}\| + \cdots + \|x_{k+1} - x_k\|,$$

by (6) we have

$$\lim_{k \to \infty,\, k \in K} \|x_{k+l(k)} - x_k\| = 0,$$

which implies

$$\lim_{k \to \infty,\, k \in K} x_{k+l(k)} = \hat{x}. \tag{8}$$

From assumption (5), we have

$$\lim_{k \to \infty,\, k \in K} [|g(x_{k+l(k)})^T d_{k+l(k)}| / \|d_{k+l(k)}\|] = 0,$$

so that, taking into account condition (b), by (8) and the continuity of g, we obtain $\|g(\hat{x})\| = 0$, which contradicts (7). □

On the basis of this result, the global convergence of the scheme (4) together with a superlinear convergence rate can be obtained by computing the stepsize α_k in such a way that the following properties are ensured:

(i) $f(x_{k+1}) \leq f(x_k)$;
(ii) $|g(x_k)^T d_k| / \|d_k\| \to 0$;
(iii) $\|x_{k+1} - x_k\| \to 0$;
(iv) the stepsize $\alpha_k = 1$ is accepted for k sufficiently large.

We observe that property (iii) would not be guaranteed by means of an Armijo-type line search technique. Property (iii) could be ensured by using line search techniques studied in the context of derivative-free methods (Refs. 5–6). These are based on replacing the Armijo acceptance rule with a condition of the form

$$f(x_k + \alpha_k d_k) \leq f(x_k) - \gamma \alpha_k^2 \|d_k\|^2, \qquad \gamma > 0,$$

that does not require gradient information. However, the use of a rule of this kind would not guarantee property (iv), and hence the superlinear convergence rate. In order to ensure all the properties (i)–(iv), we propose the following line search technique.

Line Search Algorithm (Algorithm LS)

Data. $\gamma \in (0, 1/2)$, $\delta \in (0, 1)$.
Step 0. Set $i = 0$.
Step 1. Set $\alpha = \delta^i$. If

$$f(x_k + \alpha d_k) \leq f(x_k) - \gamma \alpha^3 \|d_k\|^3, \tag{9}$$

set $\alpha_k = \alpha$ and stop.
Step 2. Set $i = i + 1$, and go to Step 1.

It can be shown easily that Algorithm LS terminates in a finite number of steps.

Proposition 2.2. There exists a finite integer i such that $\alpha_k = \delta^i$ satisfies the condition (9).

Proof. By contradiction, let us assume that, for every $i \geq 0$, we have

$$f(x_k + \delta^i d_k) > f(x_k) - \gamma \delta^{3i} \|d_k\|^3.$$

By the mean-value theorem,

$$f(x_k + \delta^i d_k) = f(x_k) + \delta^i g(\xi)^T d_k,$$

where

$$\xi = x_k + \theta\delta^i d_k, \qquad \text{with } \theta \in (0, 1).$$

Then, we have

$$g(\xi)^T d_k > -\gamma\delta^{2i}\|d_k\|^3,$$

and taking limits for $i \to \infty$, so that $\delta^i \to 0$ and $\xi \to x_k$, we obtain

$$g(x_k)^T d_k \geq 0,$$

which contradicts the descent property of d_k, i.e., $g(x_k)^T d_k < 0$. □

Now, we prove the following convergence results.

Proposition 2.3. Let $\{x_k\}$ be the sequence generated by the scheme (4), where the stepsize is computed by means of Algorithm LS, and assume that $\{x_k\}$ is bounded. Then:

(i) the sequence $\{f(x_k)\}$ converges;
(ii) $\lim_{k\to\infty}\|x_{k+1} - x_k\| = 0$;
(iii) $\lim_{k\to\infty}[|g(x_k)^T d_k|/\|d_k\|] = 0$;
(iv) every limit point of $\{x_k\}$ is a stationary point of f.

Proof. Let $\bar{x}$ be any limit point of the sequence $\{x_k\}$. Since $\{f(x_k)\}$ is monotonically decreasing, $\{f(x_k)\}$ either converges to a finite value or diverges to $-\infty$. Since f is continuous, $f(\bar{x})$ is a limit point of $\{f(x_k)\}$ and (i) is proved.

By (i) and condition (9), we have

$$\lim_{k\to\infty} \alpha_k\|d_k\| = 0, \tag{10}$$

which implies (ii).

To prove (iii), let us assume by contradiction that there exists an infinite subset $K \subseteq \{0, 1, \ldots\}$ such that, since $g(x_k)^T d_k < 0$ for all k,

$$\lim_{k\to\infty,\, k\in K} [g(x_k)^T d_k/\|d_k\|] = -\eta < 0. \tag{11}$$

Taking into account that $\{x_k\}$ is bounded, by relabeling the subset K if necessary, there exist vectors $\hat{x}, \hat{d} \in R^n$ such that

$$\{x_k\}_{k\in K} \to \hat{x}, \ \{d_k/\|d_k\|\}_{k\in K} \to \hat{d},$$

and by (11) and the continuity of g, we have

$$g(\hat{x})^T \hat{d} = -\eta < 0. \tag{12}$$

From (10), we have that

$$\text{either } \lim_{k\to\infty} \|d_k\| = 0 \quad \text{or} \quad \lim_{k\to\infty} \alpha_k = 0.$$

In the first case, condition (a) implies

$$\{|g(x_k)^T d_k|/\|d_k\|\}_{k\in K_N} \to 0,$$

which together with (ii), by Proposition 2.1, gives $g(\hat{x}) = 0$, and this contradicts (12).

In the second case, by the instructions of Algorithm LS we have, for sufficiently large $k \in K$,

$$f(x_k + (\alpha_k/\delta)d_k) > f(x_k) - \gamma(\alpha_k/\delta)^3\|d_k\|^3,$$

and by the mean-value theorem,

$$g(x_k + \theta_k(\alpha_k/\delta)d_k)^T d_k/\|d_k\| > -\gamma(\alpha_k/\delta)^2\|d_k\|^2,$$

with $\theta_k \in (0, 1)$. Taking limits for $k \to \infty$, $k \in K$, since $\alpha_k\|d_k\| \to 0$, we have $g(\hat{x})^T\hat{d} \geq 0$, which contradicts (12).

Finally, (iv) follows from (ii), (iii), and Proposition 2.1. □

Proposition 2.4. Let $\{x_k\}$ be the sequence generated by the scheme (4), where the stepsize is computed by means of Algorithm LS. Assume that $\{x_k\}$ converges to x^*, where $g(x^*) = 0$ and $H(x^*)$ is positive definite, and that

$$\lim_{k\to\infty} [\|(D_k - H(x^*)^{-1})g(x_k)\|/\|g(x_k)\|] = 0. \tag{13}$$

Then, there exists an integer $\bar{k} \geq 0$ such that, for $k > \bar{k}$, $\alpha_k = 1$ and the sequence $\{x_k\}$ converges superlinearly.

Proof. It is sufficient to show that, for k large enough, a stepsize α satisfying the Armijo rule with initial unit stepsize and $\gamma < 1/2$, i.e., $\alpha \in [0, 1]$ such that

$$f(x_k + \alpha d_k) \leq f(x_k) - \gamma\alpha|g(x_k)^T d_k|, \tag{14}$$

also satisfies Inequality (9), so that the result follows directly from Proposition 1.15 of Ref. 3.

Let us consider the positive value of α for which the right-hand sides of (9) and (14) are equal, i.e., the value

$$\hat{\alpha}_k = (|g(x_k)^T d_k|/\|d_k\|^3)^{1/2},$$

so that, for any $\alpha \in [0, \hat{\alpha}_k]$, we have

$$f(x_k) - \gamma\alpha|g(x_k)^T d_k| \leq f(x_k) - \gamma\alpha^3 \|d_k\|^3;$$

hence, a stepsize $\alpha_k \leq \hat{\alpha}_k$ satisfying the Armijo rule (14) will satisfy even Inequality (9). Therefore, we have to show only that, for k sufficiently large, $[0, 1] \subset [0, \hat{\alpha}_k]$. Since

$$g(x_k) = -D_k{}^{-1} d_k,$$

we have

$$\hat{\alpha}_k = (|d_k^T D_k^{-1} d_k| / \|d_k\|^3)^{1/2} \geq (\lambda_m(D_k^{-1}) / \|d_k\|)^{1/2},$$

where $\lambda_m(D_k^{-1})$ is the smallest eigenvalue of D_k^{-1}, and then, as $\|d_k\| \to 0$, there exists an index $\bar{k}$ such that, for $k \geq \bar{k}$, we have $\hat{\alpha}_k > 1$. □

We observe that, by assuming the compactness of the level set

$$\Omega_o = \{x \in R^n : f(x) \leq f(x_o)\}$$

for a given $x_o \in R^n$, and using a modified Cholesky factorization for obtaining a positive-definite approximation $\hat{H}(x_k)$ *of* $H(x_k)$, the assumptions (a) and (b) on the search direction d_k are satisfied. Moreover, for each point x^* for which $g(x^*) = 0$ and $H(x^*)$ is positive definite, there exists a scalar ϵ such that, if $\|x_k - x^*\| < \epsilon$, then $D_k = H(x_k)^{-1}$; i.e., the Hessian matrix will not be modified by the Cholesky factorization of Ref. 3 and (13) is satisfied.

3. Adaptive Implementation Algorithm

In order to implement the iterative scheme (4), we have to choose the integer p, i.e., to decide how many iterations will be performed without recomputing the second-order derivatives. It is evident that, in general, when p is taken small, the computational savings could be negligible with respect to the case where $p = 1$ that corresponds to an implementation of the Newton method. On the other hand, with a large p, it is to be expected that the computational advantage could be impaired or canceled by the greater slowness of the minimization process. Obviously, these effects will be more or less significant depending on the problem dimensionality and the specific structure of the objective function.

Therefore, it appears reasonable to consider p as an upper bound by taking a large value for it, say $p = 10$, and to introduce some rule for establishing, at each iteration, whether or not it can be advantageous to keep the old Hessian matrix, on the basis of the behavior of the algorithm in the

preceding iteration and the local convexity properties of the objective function. Thus, the scheme proposed here is adaptive in the sense that the Hessian is recomputed at least (instead of only) every p iterations and whenever the use of the old matrix does not appear advisable. In particular, a reasonable strategy could be that of recomputing the Hessian matrix at a given point, if the stepsize produced by the line search algorithm is relatively small and, at the same time, the objective function has not been reduced significantly.

We note that, in Ref. 7, an algorithm is described that implements, with the line search approach, the iterative scheme (2), where the approximated Newton step is taken or ignored depending on whether or not it produces a reasonable reduction in the function value.

Another element that can be usefully exploited as a further Hessian updating criterion is the amount of perturbation on the true Hessian $H(x_k)$ needed to obtain a positive-definite approximation ensuring the descent property of the search direction.

We refer to the modified Cholesky factorization of a matrix described in Ref. 8. As known, any $n \times n$ symmetric positive-definite matrix H may be expressed as the Cholesky factorization $L\Omega L^T$, where L is unit lower-triangular and Ω is positive diagonal with elements ω_j. We recall that, following the approach of Ref. 8, the Cholesky factors L and Ω can be computed in such a way that, for $j = 1, \ldots, n$,

$$\omega_j \geq \epsilon, \qquad |l_{ij}\sqrt{\omega_j}| \leq \beta, \qquad i > j,$$

where ϵ and β are positive numbers suitably chosen. In particular, the elements of Ω are given by

$$\omega_j = \max\{\epsilon, |v_j|, (\theta_j/\beta)^2\},$$

where

$$v_1 = h_{11}, \qquad v_j = h_{jj} - \sum_{s=1}^{j-1} l_{js}^2 \omega_s, \qquad \text{for } j = 2, \ldots, n,$$

$$\theta_j = \max_{j+1 \leq i \leq n} |h_{ij} - \sum_{s=1}^{j-1} l_{js} l_{is} \omega_s|, \qquad \text{for } j = 1, \ldots, n-1 \quad \text{and} \quad \theta_n = 0.$$

Then, we assume that the matrix H has been perturbed significantly using the modified Cholesky factorization if, for at least an index $j \in \{1, \ldots, n\}$, it results that

$$(\theta_j/\beta)^2 \geq \eta v_j, \tag{15}$$

with $\eta > 1$, since in this case the element ω_j will be considerably different from v_j.

This test, that allows us to quantify the alteration of the true Hessian, combined with that on the behavior of the algorithm in the previous iterate, is used in the following algorithmic scheme.

Adaptive Stabilization Algorithm (Algorithm AS)

Data. $x_o \in R^n$, $\eta > 1$, $c_\alpha \in (0, 1]$, $c_f \in (0, 1)$, $p > 1$.

Step 0. Set $k = 0$, $i = 0$, and $u = 0$.

Step 1. Compute $g(x_k)$. If $\|g(x_k)\| \leq 10^{-6}$, stop.

Step 2. If $k \neq ip$ and $u = 0$, go to Step 3. Otherwise, compute $H(x_k)$ and construct its positive-definite approximation $\hat{H}(x_k)$ by applying the modified Cholesky factorization of Ref. 8. Then, set $u = 1$ or $u = 0$, depending on whether or not, according to (15), $\hat{H}(x_k)$ is too different from $H(x_k)$. Moreover, if $k = ip$, set $i = i + 1$. Go to Step 4.

Step 3. Set $\hat{H}(x_k) = \hat{H}(x_{k-1})$.

Step 4. Compute the search direction d_k by solving the system $\hat{H}(x_k)d_k = -g(x_k)$, and compute the stepsize α_k by means of Algorithm LS. Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 5. If

$$\alpha_k \geq c_\alpha \text{ or } [f(x_k) - f(x_{k+1})]/|f(x_k)| \geq c_f,$$

set $u = 0$. Set $k = k + 1$ and go to Step 1.

Note that by the instruction at Step 5, if the reduction of the function value or the stepsize is judged to be sufficiently large, the same Hessian approximation matrix is used in the subsequent iteration, provided that $k \neq ip$, even if it represents a poor representation of the true Hessian.

We report here the numerical results obtained by applying Algorithm AS to a set of standard test problems (Ref. 9), in order to assess the extent to which we can expect a computational advantage from the proposed approach. The algorithm has been coded in double precision Fortran 77; all the runs were carried out on an IBM RISC System/6000 375. For the parameters in Algorithm AS, we have taken the values

$$\eta = 1.5, \qquad c_\alpha = 0.1, \qquad c_f = 0.25, \qquad p = 10.$$

As regards the parameters in Algorithm LS, we have chosen the values

$$\delta = 0.5, \qquad \gamma = 10^{-9};$$

the low value of γ is justified by the cubic exponent in condition (9).

For each problem of dimension n, we report the numbers n_f, n_g, n_h of function, gradient, and Hessian evaluations, the CPU time in seconds, and

Table 1. Comparison of the results obtained by the Newton method (Algorithm AS with $p = 1$) and Algorithm AS with $p = 10$ (first and second columns).

Function	n	n_f		n_g		n_h		Time		f^*	
Rosenbrock	100	170	2412	149	871	148	105	2.5	2.6	3.98	1.E–15
	200	315	4735	285	1681	284	204	37.3	33.1	3.98	1.E–16
	400	610	9978	561	3361	560	408	3190.2	2451.3	3.98	3.98
Variably dimensioned	100	37	117	32	112	31	12	0.4	0.2	0.E–00	5.E–23
	200	53	151	46	144	45	15	5.0	2.1	0.E–00	1.E–21
	400	1042	1182	1032	1172	1031	118	5735.7	720.0	1.E–27	0.E–00
Oren	100	31	97	26	92	25	10	0.3	0.2	6.E–11	3.E–11
	200	33	103	28	98	27	10	2.8	1.3	4.E–11	6.E–11
	400	35	109	30	104	29	11	155.5	65.2	2.E–11	3.E–11
Trigonometric	100	228	311	30	48	29	20	4.0	4.6	4.E–17	2.E–07
	200	486	548	55	67	54	28	59.2	51.8	1.E–17	8.E–11
	400	747	836	73	95	72	52	3444.1	3476.5	2.E–19	2.E–14
Penalty I	100	44	338	35	163	34	17	0.6	0.4	9.E–04	9.E–04
	200	45	297	36	164	35	19	4.7	3.0	2.E–03	2.E–03
	400	46	244	38	157	37	16	205.1	97.1	4.E–03	4.E–03

the objective function value f^* at the final point reached. In Table 1, these results are compared with those obtained by means of the corresponding implementation of the Newton method, i.e., by applying the same Algorithm AS with $p = 1$. In particular, for each item, the first column refers to the results obtained by the Newton method, while the second column refers to those obtained by Algorithm AS with $p = 10$.

We observe first that, as it can be expected, the number of iterations (given by $n_g - 1$), and consequently that n_f of function evaluations, is much greater for $p = 10$ than for $p = 1$. However, the Newton method computes the Hessian and its factorization at each iteration, while Algorithm AS with $p = 10$ performs a lower number n_h of Hessian evaluations. In the problems considered, the performance of the adaptive stabilization algorithm, measured in terms of the overall CPU time, is comparable with that of the Newton method for $n = 100$; it becomes better as the problem dimensionality increases with a considerable computational saving for $n = 400$ in most cases.

In order to have some indication about the advantage deriving from the introduction of the adaptive criterion in Algorithm AS, we compare in Table 2 the results obtained by Algorithm AS ($p = 10$) with those obtained by the Shamanskii modification of the Newton method, i.e., by the scheme (4), where we have taken $p = 3$.

We note that, using the adaptive Algorithm AS, we have again a significant computational time saving for $n = 400$ in all cases with the exception of the trigonometric function minimization, where the Hessian is

Table 2. Comparison of the results obtained by the Shamanskii method with $p = 3$ and Algorithm AS with $p = 10$ (first and second columns).

Function	n	n_f		n_g		n_h		Time		f^*	
Rosenbrock	100	533	2412	347	871	116	105	2.2	2.6	3.98	1.E–15
	200	983	4735	662	1681	221	204	31.1	33.1	3.98	1.E–16
	400	1829	9978	1274	3361	425	408	2516.5	2451.3	3.98	3.98
Variably dimensioned	100	61	117	56	112	19	12	0.3	0.2	0.E–00	5.E–23
	200	81	151	74	144	25	15	3.0	2.1	1.E–19	1.E–21
	400	1080	1182	1070	1172	357	118	2090.0	720.0	0.E–00	0.E–00
Oren	100	50	97	45	92	15	10	0.2	0.2	8.E–11	3.E–11
	200	54	103	49	98	16	10	1.8	1.3	5.E–11	6.E–11
	400	58	109	53	104	18	11	101.3	65.2	1.E–11	3.E–11
Trigonometric	100	420	311	42	48	14	20	3.7	4.6	3.E–12	2.E–07
	200	366	548	58	67	19	28	41.3	51.8	3.E–07	8.E–11
	400	258	836	33	95	11	52	1019.0	3476.5	2.E–07	2.E–14
Penalty I	100	98	338	71	163	24	17	0.5	0.4	9.E–04	9.E–04
	200	91	297	73	164	24	19	3.4	3.0	2.E–03	2.E–03
	400	94	244	74	157	25	16	145.7	97.1	4.E–03	4.E–03

recomputed more frequently than every three iterations. It is worth noting that, in all the problems considered, the Shamanskii method with $p = 3$ performs better than the Newton method.

Finally, it may be of interest to compare the performance of the line search strategy proposed here (Algorithm LS) with that of a standard technique. To this purpose, we have applied Algorithm AS with $p = 1$ and $p = 10$, where the stepsize is computed by means of the standard Armijo rule with $\delta = 0.5$ and $\gamma = 10^{-3}$. The results obtained for the test problems considered before were in most cases the same as those reported in Table 1. In two runs, we obtained slightly greater values of n_f, n_g, and n_h, while in one case these values were slightly lower. Therefore, it appears that the two line search strategies are practically equivalent in performance.

The results reported in this section confirm the usefulness of the approach described for the solution of problems where the computation of the second-order derivatives turns out to be time-consuming. In Section 4, we report some results relative to a specific and important application, that is, the solution of neural network training problems, whose features appear to be suited for obtaining significant computational advantages.

4. Computational Results in Neural Network Training Problems

The computation of the parameters of a neural network (training problem) can be formulated as an unconstrained minimization problem. In

particular, a neural network implements a nonlinear input–output mapping $\psi(\cdot\,; w)\colon R^{n_i} \to R$, where $w \in R^n$ is the vector of network parameters.

Let us consider a given set of data (input–output pairs),

$$T = \{(u_j, d_j), u_j \in R^{n_i}, d_j \in R, j = 1, \ldots, P\}.$$

The training problem can be formulated as follows:

$$\min_{w \in R^n} \sum_{j=1}^{P} (d_j - \psi(u_j; w))^2 + \tau \|w\|^2, \tag{16}$$

where the sum measures the degree of success in the approximation of the output data and the second term (with $\tau > 0$) is the complexity penalty, introduced to prevent the network from overfitting the training data; e.g., see Ref. 10.

We consider here the class of neural networks known as radial basis function (RBF) networks, that implements an input-output mapping $\psi\colon R^{n_i} \to R$ of the form

$$\psi(u; \lambda_1, \ldots, \lambda_{n_r}, c_1, \ldots, c_{n_r}) = \sum_{i=1}^{n_r} \lambda_i \phi(\|u - c_i\|^2),$$

where $\phi\colon R^+ \to R^+$ is a radially symmetric function [among the various possible choices, we use here the direct multiquadric function $(\|u - c_i\|^2 + \sigma^2)^{1/2}$, with the shift parameter $\sigma = 0.1$], $\lambda_i \in R, i = 1, \ldots, n_r$, are the weights, $c_i \in R^{n_i}$ are the RBF centers, and n_r is the number of hidden nodes of the network. Then, with reference to problem (16), the parameter vector $w \in R^{n_r(n_i+1)}$ is composed of the n_r scalars λ_i and the n_r vectors c_i.

Note that the problem dimensionality may be very large, depending on the dimension n_i of the input space and the number n_r of hidden nodes. Moreover, the computation of the first-order and second-order derivatives may result to be time consuming, since the number P of input–output pairs in the training set T may be very large, so that the structure of the objective function is composed of the sum of several nonlinear terms. Therefore, by applying Algorithm AS, we can expect a significant reduction in the overall computational cost. We observe that, in this kind of applications, the objective function is characterized by the presence of multidimensional plateaus, i.e., regions where the slope is simultaneously shallow in multiple dimensions. For this reason, in Algorithm AS, the parameter that controls the reduction in the function value is set at $c_f = 0.001$ and we have used the stopping criterion $\|g\| \le 10^{-5}$.

We have considered the following two neural network training problems (Ref. 11).

4.1. Building Problem. The aim is to predict the electrical energy consumption in a building based on various elements such as the date, time of day, outside temperature, solar radiation, and so on. The input space dimension is $n_i = 14$, the training set consists of $P = 500$ pairs, and the network is composed of $n_r = 5$ hidden nodes, so that the problem dimension is $n = 75$.

4.2. Heart Problem. The task is to predict the heart disease, i.e., to decide whether at least one of four major vessels is reduced in diameter by more than 50%. The decision is made based on personal data and results of various medical examination. The input space dimension is $n_i = 35$, the training set consists of $P = 303$ pairs, and the network is composed of $n_r = 3$ hidden nodes, so that the problem dimension is $n = 108$.

We compare in Table 3 the results obtained by Algorithm AS with $p = 1$ (Newton method) and with $p = 10$, starting from five different initial points (a, b, c, d, e), obtained by choosing randomly the weights λ_i in the interval $[-0.5, 0.5]$ and the centers c_i among the input vectors in T.

We note that the number n_h of Hessian evaluations is relatively higher than that for Hessians recomputed only every $p = 10$ iterations $[n_h > (n_g - 1)/p]$. This is due to the presence of the plateau regions and hence to the inherent ill-conditioning of the Hessian matrix which is perturbed more frequently by the modified Cholesky factorization. In spite of this fact, which derives from the specific feature of network training problems, the computational savings, in terms of the CPU time, are significant for both problems in all runs. In particular, for the building problem, we have a mean time value of 4247 sec ($p = 10$) against 6031 sec ($p = 1$) with an average saving of 29.6%; for the heart problem, the mean time values are 2706 sec ($p = 10$) and 4382 sec ($p = 1$) with an average saving of 38.2%.

Table 3. Comparison of the results obtained by the Newton method (Algorithm AS with $p = 1$) and Algorithm AS with $p = 10$ (first and second columns), starting from five different initial points.

Problem	Initial point	n_f		n_g		n_h		Time		f^*	
Building	a	8065	9930	550	705	549	333	4700	3065	0.5748	0.5748
	b	12577	18278	850	1216	849	585	7272	5435	0.5748	0.5478
	c	10017	15174	675	1032	674	474	5775	4404	0.5748	0.5478
	d	8886	10941	616	752	615	330	5258	3084	0.5748	0.5478
	e	11870	16669	838	1125	837	572	7148	5245	0.5748	0.5478
Heart	a	3638	4583	265	373	264	186	3613	2622	5.6909	5.6909
	b	3559	4230	245	348	244	157	3346	2227	5.6617	5.6909
	c	5406	4121	382	341	381	156	5218	2212	5.6909	5.6909
	d	5012	5055	337	366	336	204	4611	2875	5.6617	5.6617
	e	5327	7455	375	554	374	252	5124	3595	5.6941	5.6941

Table 4. Comparison of the results obtained by the Shamanskii method with $p = 3$ and Algorithm AS with $p = 10$ (first and second columns), starting from five different initial points.

Problem	Initial point	n_f		n_g		n_h		Time		f^*	
Building	a	17326	9930	1043	705	348	333	3544	3065	0.5748	0.5748
	b	21472	18278	1293	1216	431	585	4423	5435	0.5478	0.5478
	c	22149	15174	1293	1032	431	474	4585	4404	0.5478	0.5478
	d	15485	10941	960	752	320	330	3257	3084	0.5748	0.5478
	e	32367	16669	2044	1125	681	572	6901	5245	0.5479	0.5478
Heart	a	7436	4583	494	373	165	186	2494	2622	5.6909	5.6909
	b	8418	4230	518	348	173	157	2634	2227	5.6617	5.6909
	c	9086	4121	604	341	201	156	3039	2212	5.6909	5.6909
	d	14593	5055	867	366	289	204	4416	2875	5.6617	5.6617
	e	13857	7455	882	554	294	252	4463	3595	5.6941	5.6941

In Table 4, we compare the results obtained by the adaptive Algorithm AS ($p = 10$) with those obtained by the Shamanskii method ($p = 3$). We note that, in eight runs over ten we have an appreciable saving in computational time. Thus, these results indicate that the introduction of the adaptive criterion in Algorithm AS is beneficial.

As regards the comparison of the performance of the two line search strategies (Armijo rule vs Algorithm LS), by applying Algorithm AS with $p = 10$ we obtained the same results in six runs; the results relative to the remaining four runs are reported in Table 5. In all these four cases, the performance of Algorithm AS using Algorithm LS is better than using the Armijo rule. Although this computational experience is very limited, it appears that the line search strategy proposed in this paper allows some computational advantage, at least in the specific application considered.

Table 5. Comparison of the results obtained by Algorithm AS ($p = 10$) using the Armijo rule and using Algorithm LS (first and second columns).

Problem	Initial point	n_f		n_g		n_h		Time		f^*	
Building	a	13354	9930	896	705	521	333	4801	3065	0.5748	0.5748
	c	18304	15174	1206	1032	613	474	5755	4404	0.5478	0.5478
Heart	a	5308	4583	423	373	230	186	3295	2622	5.6909	5.6909
	b	4154	4230	308	348	176	157	2523	2227	5.6617	5.6909

5. Conclusions

The results given in this paper show that the global convergence of the Newton method can be enforced even when a finite number of iterations is performed using the same Hessian matrix, thus avoiding its evaluation at each step. Moreover, the line search technique proposed here ensures the superlinear convergence of the stabilization method.

From the numerical results obtained for a set of standard test problems and from those relative to the specific application of neural network training, it appears that the adaptive implementation scheme described, where the Hessian matrix is only periodically evaluated, may allow considerable computational savings when the calculation of the second-order derivatives is expensive.

Finally, we observe that the proposed strategy for obtaining significant computational advantages may be employed in connection with truncated Newton schemes for solving large scale optimization problems.

References

1. Ortega, J. M., and Rheinboldt, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, 1970.
2. Shamanskii, V. E., *On a Modification of Newton's Method*, Ukrainskyi Matematychnyi Zhurnal, Vol. 19, pp. 133–138, 1967 (in Russian).
3. Bertsekas, D. P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, New York, NY, 1980.
4. Kelley, C. T., *Iterative Methods for Optimization*, SIAM, Philadelphia, Pennsylvania, 1999.
5. De Leone, R., Gaudioso, M., and Grippo, L., *Stopping Criteria for Linesearch Methods without Derivatives*, Mathematical Programming, Vol. 30, pp. 285–300, 1984.
6. Grippo, L., Lampariello, F., and Lucidi, S., *Global Convergence and Stabilization of Unconstrained Methods without Derivatives*, Journal of Optimization Theory and Applications, Vol. 56, pp. 385–406, 1988.
7. Wright, S. J., *Primal–Dual Interior-Point Methods*, SIAM, Philadelphia, Pennsylvania, 1997.
8. Gill, P. E., and Murray, W., *Newton-Type Methods for Unconstrained and Linearly Constrained Optimization*, Mathematical Programming, Vol. 7, pp. 311–350, 1974.
9. Moré, J. J., Garbow, B. S., and Hillstrom, K. E., *Testing Unconstrained Optimization Software*, ACM Transactions on Mathematical Software, Vol. 7, pp. 17–41, 1981.
10. Haykin, S., *Neural Networks*, 2nd Edition, Prentice-Hall International, Upper Saddle River, New Jersey, 1999.

11. PRECHELT, L., *Proben 1: A Set of Neural Network Benchmark Problems and Benchmarking Rules*, Technical Report 21/94, Fakultät für Informatik, Universität Karlsruhe, Karlsruhe, Germany, 1994.