SULTAN QABOOS UNIVERSITY

Department of Mathematics and Statistics

Extra-Updates Criterion for the Limited Memory BFGS Algorithm for Large Scale Nonlinear Optimization

by

M. Al-Baali

December 2000

Extra-Updates Criterion for the Limited Memory BFGS Algorithm for Large Scale Nonlinear Optimization *

M. Al-Baali †

December 7, 2000

Abstract

This paper studies recent modifications of the limited memory BFGS (L-BFGS) method for solving large scale unconstrained optimization problems. Each modification technique attempts to improve the quality of the L-BFGS Hessian by employing (extra) updates in certain sense. Because at some iterations these updates might be redundant or worsen the quality of this Hessian, this paper proposes an updates criterion to measure this quality. Hence, extra updates are employed only to improve the poor approximation of the L-BFGS Hessian. The presented numerical results illustrate the usefulness of this criterion and show that extra updates improve the performance of the L-BFGS method substantially.

Key Words: large scale optimization, quasi-Newton methods, limited memory BFGS method.
AMS subject classification: 90C30, 65K05, 49.

1 Introduction

Finding a solution to a general large scale nonlinear optimization problem

$$\min f(x), \tag{1}$$

where f is a smooth function of n variables, by low storage quasi-Newton methods is considered. It is assumed that n is large so that a small number (at least 7) of vectors of n components can be stored, but an $n \times n$ symmetric matrix cannot. Three of these vectors are required to store the current iterate $x^{(k)}$, the gradient vector $g^{(k)} = \nabla f(x^{(k)})$ and the search direction $s^{(k)}$. The other vector pairs are used to store the information of the Hessian approximation $H^{(k)-1}$ so that the product $H^{(k)}v$, for any vector v, can be obtained without calculating this matrix explicitly.

*Presented at the 2000 AMS-IMS-SIAM Joint Summer Research Conference in Algorithms, Computational Complexity, and Models of Computation for Nonlinear and Multivariate Problems, Mount Holyoke College, USA, July 16-20, 2000.

†Department of Mathematics and Statistics, Sultan Qaboos University, P.O. Box 36, Al-Khod 123, Muscat, Sultanate of Oman. E-mail: albaali@squ.edu.om.

In the limited memory BFGS (L-BFGS) method of Nocedal [16], these pairs are defined by the differences

$$\delta^{(i)} = x^{(i+1)} - x^{(i)}, \quad \gamma^{(i)} = g^{(i+1)} - g^{(i)}, \qquad k - \hat{m} + 1 \leq i \leq k, \tag{2}$$

where

$$\hat{m} = \min(m, k) \tag{3}$$

and for convenience $m \geq 2$. These pairs and the positive scalars $\delta^{(i)T}\gamma^{(i)}$, for $i = k - \hat{m} + 1, \ldots, k$, are stored during the previous $\hat{m}$ iterations so that the most recent pair replaces the oldest one when $k > m$. The inverse L-BFGS Hessian $H^{(k+1)}$ is defined implicitly as the outcome of updating a matrix $D^{(k)}$ $\hat{m}$ times in terms of the pairs (2), using the BFGS updating formula. Note that this Hessian is equivalent to the BFGS Hessian if $m = \infty$ which is not possible in practice. Thus, the latter Hessian depends on more information than the L-BFGS one. Hence, under mild conditions on convex functions, the BFGS method converges superlinearly (Powell [17]), while the L-BFGS algorithm converges linearly (Liu and Nocedal [12]).

In practice, the L-BFGS method is attractive because of its low storage requirement (usually, $3 \leq m \leq 20$) and its useful performance which has been observed when the method was applied with $D^{(k)}$ defined by a multiple of the unit matrix to general large scale optimization problems (e.g. [2], [3], [6], [7], [10], [12], [14], and [18]). Nevertheless, this method can be very slow on a certain type of problems (e.g. [2], [6] and [15]), because the approximation of the L-BFGS Hessian is sometimes poor compared to the BFGS Hessian, particularly for ill-conditioned problems. Thus, there is room for improving the quality of the Hessian approximation. In particular, Al-Baali [2] and Byrd, Nocedal and Zhu [6] suggest some modifications of the L-BFGS Hessian, which are based on using a fixed number (say, p) of extra updates to $D^{(k)}$ before using the normal $\hat{m}$ updates. Although the authors reported encouraging numerical results in terms of the number of function and gradient evaluations required to solve some test problems, the total number of updates is usually larger than that required by the L-BFGS method.

Since this drawback is due to the fact that some extra updates become redundant in some cases, this paper attempts to avoid such redundancy. It proposes an updates criterion to distinguish between poor and acceptable quality of the L-BFGS Hessian. Hence extra updates are employed to improve only the poor quality one. In Section 2, we provide some details about the storage locations and the number of updates employed by the methods of this paper. In Section 3, further details concerning certain modified L-BFGS algorithms are outlined. Section 4 derives the updates criterion and shows that a number of extra updates $p_k \leq p$ can be varied from one iteration to another. Section 5 gives some numerical results. It is shown that the choice $p_k = 0$ occurs at several iterations, and that the performance of a modified L-BFGS algorithm with the new criterion is substantially better than that of the L-BFGS method.

2 Motivations for Extra Updates

Since the BFGS Hessian contains more information than the L-BFGS Hessian, it is expected that the quality of the latter Hessian improves as m increases. Although this is

not always the case (see Fletcher [7]); increasing m also means increasing the storage location for storing the pairs (2). Therefore, the L-BFGS Hessian is need to be updated particularly as follows.

In the modified L-BFGS algorithm of Byrd, Nocedal and Zhu [6], p extra updates are employed in terms of p vector pairs. Thus, the storage location is increased from approximately $2m$ to $2(m+p)$ vectors, as well as the number of updates from $\hat{m}$ to

$$m_k = p + \min(m, k - k_i + 1), \tag{4}$$

where $k_i, i = 1, 2, \ldots,$ denote certain iterations. Note that the p pairs do not have the difference form as in (2), since they are generated at iteration k_i by the inner conjugate gradient iterations of one step of the discrete-truncated Newton method.

The authors show that this technique improves the performance of the L-BFGS method when applied to a certain type of problems. It is thus clear that the extra updates improve the quality of the L-BFGS Hessian at some iterations. However, when we examined certain numerical results, we observed that the quality of the L-BFGS Hessian seems good so that employing extra updates are not needed. Therefore extra updates should be employed when only satisfy certain properties. In fact, Morales and Nocedal [13] consider this possibility and improve the performance of the above modified algorithm by employing a number of extra updates when a value of the steplength (see Section 3) satisfies a certain condition. Although this condition seems useful in practice, it is not directly related to the quality of the L-BFGS matrices. This paper proposes an updates criterion to measure the quality of these matrices and hence considers employing at most p extra updates whenever necessary.

We note that the modified L-BFGS algorithm of Byrd, Nocedal and Zhu [6] or that of Morales and Nocedal [13] have the disadvantages that it increases the storage location as well as the cost of evaluating the p extra pairs, but it also has the advantage that it preserves the simple form of the L-BFGS matrix. To avoid these drawbacks and maintain this useful property, Al-Baali [2] introduces certain modification techniques to the L-BFGS matrix similar to that of Byrd, Nocedal and Zhu [6], except that the p extra pairs are chosen from the set of the stored pairs (2) in several ways. In particular, we consider the following technique. Once $k \geq m$, the first $q = [p/m]$ updates are employed in terms of (2) with m replaced by q. The other updates, say mr, depend on re-using the pairs (2) r times. (Further details can be seen in the next section).

Al-Baali [2] reported encouraging numerical results and showed that this modified L-BFGS algorithm with fixed values of m and p works better than the L-BFGS method for general problems. In certain cases, however, we observed that employing extra updates does not improve (or even worsens) the performance of the L-BFGS method. To avoid this drawback, the number of extra updates need not be fixed for all iterations, a subject that is the main aim of this paper. For example, let the number of updates be defined by

$$m_k = \begin{cases} k & \text{if } k < m \\ m + p_k & \text{if } k \geq m, \end{cases} \tag{5}$$

where $p_k \in [0, p]$ is a parameter. As mentioned above the value of p_k depends on the quality of the Hessian approximation. Thus, the choice $p_k = 0$ is possible and extra updates might be employed at some iterations rather than at all, as in the modified algorithm of Al-Baali [2].

3 Modified L-BFGS Algorithms

We now describe the L-BFGS method of Nocedal [16] and certain modification techniques which preserve the inverse Hessian approximations $\{H^{(k)}\}$ in the L-BFGS matrix form.

The L-BFGS algorithm resembles the line search BFGS method which generates a sequence of points $\{x^{(k)}\}$ by the equation

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} s^{(k)}, \tag{6}$$

where $x^{(1)}$ is given, $\alpha^{(k)}$ is a steplength parameter and

$$s^{(k)} = -H^{(k)} g^{(k)} \tag{7}$$

is the search direction. At every iteration, the matrix $H^{(k)}$ is updated to

$$H^{(k+1)} = \mathrm{bfgs}(H^{(k)}, \delta^{(k)}, \gamma^{(k)}), \tag{8}$$

where the function

$$\mathrm{bfgs}(H, \delta, \gamma) = (I - \frac{\delta\gamma^T}{\delta^T\gamma}) H (I - \frac{\gamma\delta^T}{\delta^T\gamma}) + \frac{\delta\delta^T}{\delta^T\gamma} \tag{9}$$

is the BFGS updating formula and $\delta^{(k)}$ and $\gamma^{(k)}$ are the difference vector pairs

$$\delta^{(k)} = x^{(k+1)} - x^{(k)}, \quad \gamma^{(k)} = g^{(k+1)} - g^{(k)} \tag{10}$$

which is the most recent pair of the set of pairs (2).

The BFGS method has several useful properties (e.g., see Fletcher [8]). In particular, it ensures the quasi-Newton condition

$$H^{(k+1)}\gamma^{(k)} = \delta^{(k)} \tag{11}$$

and preserves positive definiteness of the matrices $\{H^{(k)}\}$ if $\alpha^{(k)}$ is chosen to satisfy the Wolfe conditions

$$f^{(k)} - f^{(k+1)} \geq c_1 |\delta^{(k)T} g^{(k)}|, \quad \delta^{(k)T}\gamma^{(k)} \geq (1 - c_2)|\delta^{(k)T} g^{(k)}|, \tag{12}$$

where $f^{(k)}$ denotes $f(x^{(k)})$, $c_1 \in (0, 1/2)$ and $c_2 \in (c_1, 1)$. Note that the second condition in (12) guarantees that $\delta^{(k)T}\gamma^{(k)} > 0$ whenever $g^{(k)} \neq 0$.

The L-BFGS method is defined as above, except that the inverse L-BFGS Hessian contains less information than the inverse BFGS Hessian (8) and the product in (7) is obtained without computing the matrix $H^{(k)}$ explicitly. To illustrate this computation, we state the following. The single update (8) is equivalent to the k updates

$$H^{(k+1)} = \mathrm{lbfgs}(k, H^{(1)}, \delta^{(1)}, \ldots, \delta^{(k)}, \gamma^{(1)}, \ldots, \gamma^{(k)}), \tag{13}$$

where for any matrix H and any positive integer $m \leq k$,

$$\begin{aligned} &\mathrm{lbfgs}(m, H, \delta^{(k-m+1)}, \ldots, \delta^{(k)}, \gamma^{(k-m+1)}, \ldots, \gamma^{(k)}) = \\ &\quad \mathrm{lbfgs}(m-1, \mathrm{bfgs}(H, \delta^{(k-m+1)}, \gamma^{(k-m+1)}), \\ &\quad \delta^{(k-m+2)}, \ldots, \delta^{(k)}, \gamma^{(k-m+2)}, \ldots, \gamma^{(k)}), \end{aligned} \tag{14}$$

and

$$\text{lbfgs}(1, H, \delta, \gamma) = \text{bfgs}(H, \delta, \gamma). \tag{15}$$

Expression (14) defines the L-BFGS updating formula which employs m BFGS updates to H in terms of the vector pairs (2). For simplicity, these m updates are denoted by $L_m^{(k)}(H)$ which satisfies the property

$$L_m^{(k)}(H) = L_{m-p}^{(k)}(L_p^{(k-m+p)}(H)), \tag{16}$$

where the zero update $L_0^{(k)}(H) = H$. Note that the notation $L_m^{(k)}$ is used with $m \leq k$, but if the inequality does not hold, then the subscript m is replaced by k. Note also that we will omit the parentheses whenever confusion does not occur.

Because expression (16) can be written as a sum of symmetric matrices of a certain structure, Nocedal [16] proposes a two loop recursive formula which computes the product $L_m^{(k)}(H)g$ for any vector g efficiently. This recursive formula is described by the following algorithm (see also Averick and Moré [3] and Byrd, Nocedal and Schnabel [5], for instance).

Algorithm A (for computing $v \leftarrow L_m^{(k)}(H)g$).

Given H and $\delta^{(i)}, \gamma^{(i)}$ and $\rho^{(i)} = 1/\delta^{(i)T}\gamma^{(i)}$, for $i = k - m + 1, \ldots, k$.

step 0. $v = g$

step 1. for $l = m, \ldots, 1$
$i = l + k - m$
$\sigma_l = \rho^{(i)}\delta^{(i)T}v$ (stored)
$v \leftarrow v - \sigma_l\gamma^{(i)}$
end for

step 2. $v \leftarrow Hv$

step 3. for $l = 1, \ldots, m$
$i = l + k - m$
$\bar{\sigma} = \rho^{(i)}\gamma^{(i)T}v$
$v \leftarrow v + (\sigma_l - \bar{\sigma})\delta^{(i)}$
end for

We observe that the storage requirements for this algorithm are $2mn$ spaces to hold the pairs (2) and $2m$ spaces for the scalars. Excluding step 2, we note that each update requires $4n$ multiplications. If H is diagonal and the scalar product $\delta^{(k)T}g$ is already computed (which is the case for the L-BFGS method), then computing $L_m^{(k)}(H)g$ requires only $4mn$ multiplications. Note that the scalars $g^T L_l^{(k)}(H)g$, for $l = 1, \ldots, m$, can be computed not only before the "end for" of step 3, but also before the "end for" of step 1. This useful property plays an important role for measuring the quality of the L-BFGS matrices before calculating them implicitly at the "end for" of step 3. Hence, extra updates might be made before and after step 2.

Now it is clear that the next BFGS search direction $s^{(k+1)}$ (defined by (7) with k replaced by $k+1$) can be computed without storing the matrix (13) explicitly, but the storage

location is approximately $2kn$ spaces. Since storing k vector pairs is not possible (particularly for large scale optimization problems) when k is sufficiently large, Nocedal [16] recommends updating a diagonal matrix $D^{(k)}$ $\hat{m}$ times to

$$H^{(k+1)} = L_m^{(k)}(D^{(k)}) \tag{17}$$

in terms of the pairs (2). Liu and Nocedal [12] recommend the choice of the multiple of the unit matrix

$$D^{(k)} = \nu^{(k)} I, \qquad \nu^{(k)} = \frac{\delta^{(k)T}\gamma^{(k)}}{\gamma^{(k)T}\gamma^{(k)}} \tag{18}$$

which depends on the most recent pair of (2). For simplicity, we consider this choice here, though another desirable one has been suggested by Al-Baali [2].

It is worth noting that a comparison of the number of multiplications required to obtain the search direction by the L-BFGS algorithm (defined by (7), (17) and (18)) to that required by the BFGS method shows that the cost of the former direction is cheaper than the latter if $k < n/2m - 1$. For certain problems with sufficiently large value of n, this inequality is usually satisfied. This case provides further motivation for using L-BFGS rather than BFGS.

However, since a small number of m is usually used, the L-BFGS method with the choice (18) can be very slow in certain cases. Although this difficulty is sometimes overcome by repeating the run with a larger value of m, the repeating procedure is time consuming and one may not be able to increase the value of m since the memory is limited (see, also, Fletcher [7] and Nocedal [15]). Therefore certain techniques have been introduced to the L-BFGS method.

In particular, Al-Baali [2] replaces the L-BFGS matrix (17) by

$$H^{(k+1)} = L_m^{(k)} L_p^{(k)}(D^{(k)}), \tag{19}$$

where $p \leq m$ is a fixed number, which employs $\hat{m} + p$ updates to the matrix in (18). Although this number is larger than $\hat{m}$, these updates only depend on the pairs (2) and the storage location is increased by only p spaces.

The author shows that this modified L-BFGS algorithm with some fixed values of m and p works well in practice. Although it improves the performance of the L-BFGS method in terms of the function and gradient evaluations required to solve general problems, it may worsen its performance in terms of the number of updates. Therefore we are concerned to keep number of times the p extra updates are employed at a minimum because they are time consuming. In the next section, we suggest a criterion to guess the number of extra updates, which is not necessarily smaller than m (p will be used to denote the maximum number of extra updates). Note that the choice (19) can be generalized to

$$H^{(k+1)} = (L_m^{(k)})^r L_q^{(k)}(D^{(k)}), \tag{20}$$

where $r \geq 0$ and $q \in [1, \hat{m}]$ such that $rm + q \leq m + p$. Note that $r = 0$ and $q = \hat{m}$ if $m \leq k$. The storage requirements for this algorithm is only $(r-1)m+q$ (which defines p_k in (5)) spaces more than L-BFGS, which can be ignored compared to the other requirement spaces. Although the number of updates required to obtain (20) is usually larger than m, the total number of updates required to solve certain problems is still small compared to that required by the L-BFGS method (see Section 5, for details).

Finally it is also worth noting that the matrix (19) differs from the choice

$$H^{(k+1)} = L_m^{(k)} L_p^{[p]}(D^{(k)}), \tag{21}$$

where the square brackets are used to denote that the p extra pairs are different from the pairs (2). The choice (21) defines the modified L-BFGS algorithm of Byrd, Nocedal and Zhu [6]. Although further modification has been considered recently by Morales and Nocedal [13], the storage of approximately $2(m+p)n$ spaces are still required rather than $2mn+p$ as for (19); and obtaining the p pairs is not required for (19).

4 Conditions for Employing Extra Updates

We now study the behaviour of the L-BFGS matrices (17) and the modified matrices (20) by measuring their quality on the basis of certain useful properties that usually hold for the BFGS matrices.

It is obvious that these matrices maintain the positive definiteness property and satisfy the quasi-Newton condition (11), since the last update in both (17) and (20) depends on the pair (10) which satisfies the inequality $\delta^{(k)T}\gamma^{(k)} > 0$.

Another useful property is that if the objective function f is strictly convex quadratic, then the BFGS method with exact line searches terminates in at most n iterations (e.g., see Fletcher [8]). This termination property has been extended to the L-BFGS method with at least one update employed at each iteration by Kolda, O'Leary and Nazareth [11] who also suggest several modifications of the L-BFGS method. Because the reported numerical results show slight improvement of the modified algorithms over the L-BFGS method in certain cases only, it seems that the latter termination property may not hold when approximate line searches are performed. Therefore, we study the possible convergence of the L-BFGS matrices in the context of inexact line searches as follows.

We begin with the result of Ge and Powell [9] who show that the BFGS matrices $\{H^{(k)}\}$ converge to a matrix, say H^*, if f is strictly convex quadratic and the difference vectors $\{\delta^{(k)}\}$ satisfy certain conditions. Although the authors state that this result does not hold for general twice continuously differentiable functions, it implies the following. If the difference $H^{(k+1)} - H^{(k)}$ is not sufficiently close to the zero matrix, then $H^{(k+1)}$ approximates H^* poorly and, hence, modifying $H^{(k+1)}$ might be useful. In fact, Al-Baali [1] updates the BFGS Hessian in terms of a few pairs of (2), and shows the usefulness of extra updates.

To motivate this improvement and generalize it to the extra-updates matrices of this paper, we consider a particular result of Boggs and Tolle [4] who analyze the behaviour of BFGS matrices independently of the optimization setting. To avoid confusion with the above BFGS matrices, let $\{H_l\}$ be generated by the BFGS formula

$$H_{l+1} = \mathrm{bfgs}(H_l, \delta_l, \gamma_l), \tag{22}$$

where H_1 is positive definite. Here δ_l are given vectors in R^n, that are not related directly to its generation by an optimization process and γ_l are vectors that depend on δ_l in some way. In particular, let

$$\gamma_l = G\delta_l, \tag{23}$$

where G is the Hessian of a strictly convex quadratic f. Note that equation (23) includes the definition of δ_l and γ_l in (10) with the superscript (k) replaced by the subscript l as a special case. The authors establish the convergence of $\{H_l\}$ under certain restrictions on $\{\delta_l\}$.

An examination of this result shows that it also holds if the sequence $\{\delta_l\}$ is a subset of the sequence of vectors $\{\delta^{(i)}\}_{i=k-\hat{m}+1}^{k}$, defined in (2), assuming the vectors satisfy certain conditions. In this case the sequence of matrices $\{H_l\}$ depends on re-using these vectors, as well as the corresponding vectors $\{\gamma^{(i)}\}_{i=k-\hat{m}+1}^{k}$, several times. For convenience, let

$$\delta_l = \delta^{(k-\hat{m}+q)},\ \gamma_l = \gamma^{(k-\hat{m}+q)}, \quad l = \hat{m}r + q,\ q = 1, 2, \ldots, \hat{m},\ r = 0, 1, 2, \ldots \tag{24}$$

Hence, for the choice $H_1 = D^{(k)}$, the BFGS matrix (22) can be written as follows

$$H_{l+1} = (L_m^{(k)})^r L_q^{(k)} (D^{(k)}) \tag{25}$$

which is equivalent to the modified L-BFGS Hessian (20) if r is bounded.

In practice, one would like to stop updating when the difference $H_{l+1} - H_l$ is sufficiently close to the zero matrix. Because this difference matrix cannot be calculated explicitly, an alternative scheme to quantify this difference should be tested instead. In particular, we check the relative error

$$\frac{|d_{l+1} - d_l|}{d_{l+1}} \leq \epsilon, \tag{26}$$

where $\epsilon > 0$ and

$$d_l = g^T H_l g \tag{27}$$

which can be calculated before the "end for" of step 1 of Algorithm A, defined in Section 3. In this paper, g denotes the most recent gradient vector $g^{(k+1)}$. The cost of computing the scalar (27) is equal to approximately n multiplications if $D^{(k)}$ is defined as a multiple of the unit matrix (e.g. (18)). Although the limit $d_{l+1} - d_l \to 0$ does not imply convergence of $\{H_{l+1} - H_l\}$, we use the criterion (26) for nonconvergence as follows. If the inequality (26) is satisfied for some $l = j \geq m$, then we stop updating and define r and q so that $mr + q = j$. Otherwise, the inequality does not hold and hence further updates are needed.

It is worth mentioning that Boggs and Tolle [4] also show that the sequence of matrices $\{H_l\}$ converges linearly. Because of this result and that this sequence may not converge for general functions, a value of r and hence a maximum number of extra updates should be given small.

5 Numerical Experiments

We now present results of some modified L-BFGS algorithms on a set of standard test problems used by Al-Baali [2] to examine the updates criterion (26). We consider the class of modified L-BFGS methods which defines the inverse Hessian approximation by (25) (or (20)), using (24) and (18), for some values of $l \leq m + p$. The values of m, p and ϵ are defined a priori, while the values of r, q and hence l are defined at each iteration depending on the criterion (26). All runs were performed in double precision using a software routine that implements all algorithms but differ only in the choices of m, p and ϵ. The routine

finds a value of the steplength $\alpha^{(k)}$ that satisfies the strong Wolfe conditions which imply (12) with the choices $c_1 = 10^{-4}$ and $c_2 = 0.9$. The runs were terminated when

$$f^{(k)} - f^{(k+1)} \leq \epsilon_1 \max(1, |f^{(k+1)}|), \tag{28}$$

where $\epsilon_1 \approx 2.22 \times 10^{-16}$, and

$$\|g^{(k+1)}\| \leq 10\sqrt{\epsilon_1} \max(1, |f^{(k+1)}|), \tag{29}$$

where $\|.\|$ denotes the Euclidean norm. (Other details can be seen in Al-Baali [2].)

The variables nfe and nu in the tables below denote, respectively, the number of function evaluations and the number of updates required to solve the problem. Because the number of multiplications for testing the criterion (26) and employing an update are approximately n and $4n$, respectively, nu was calculated by dividing the number of scalar products by 4. Since the number of line searches as well as the number of gradient evaluations were a little less than nfe, we do not present them here.

We firstly applied the above algorithms to the quartic function Q20 with number of variables $n = 100$, which belongs to the class of 28 tests of Byrd, Nocedal and Zhu [6]. A desirable feature of this test is that some improvements in terms of nfe can be seen when more information is introduced to the L-BFGS Hessian (see [2] and [6]). It thus illustrates the behaviour of the modified L-BFGS algorithms when the number of extra updates are increased without increasing the number of the stored vector pairs.

For the choices

$$m = 5, 10, 20, 40, \quad p = 0, m, 2m, 4m, 9m, \quad \epsilon = 0, 10^{-10}, 10^{-6}, 10^{-4}, 10^{-2} \tag{30}$$

the results are given in Table 1 through Table 4 in the form nfe/nu required to solve Q20. Note that the choice $p = 0$ yields the normal L-BFGS method of Nocedal [16] and $\epsilon = 0$ yields the modified L-BFGS algorithm with fixed p extra updates at each iteration (as proposed by Al-Baali [2]).

We observe from Tables 1–4 that for all values of m, nfe decreases as p increases or ϵ decreases, although this improvement increases nu. However, both nfe and nu are decreased in some cases (e.g., see in particular the first column of the results in Tables 3 and 4).

Except for $m = 5$, the choices $\epsilon = 10^{-10}$ and $\epsilon = 0$ yield the same nfe which indicates that all extra updates are employed for the former choice. Although the latter choice for ϵ is smaller than the former one, it yields smaller nu due to the following reason. We do not examine condition (26) which requires the cost of $(m + p)/4$ updates per iteration, because this condition never satisfied for $\epsilon = 0$. This observation suggests that examining this condition for sufficiently small value of ϵ is not necessary after each extra update. Instead, however, it can be checked after every m updates. We observe that large values of ϵ yield a little reduction in nfe, even for large values of p (see in particular the first column of the results in Tables 3 and 4). However, differences can be seen in Tables 1–4 when ϵ is sufficiently small and, hence, large values of p reduce nfe substantially.

To compare this result with the best possible one, we considered an application of the L-BFGS method with m sufficiently large so that the method became equivalent to the BFGS method. For $m = 200$, we found that the nfe/nu required to solve the Q20 function are 180/15975. Comparing this result to those of Table 4, we observe that the modified

L-BFGS algorithm with most choices of p and ϵ performs better than the BFGS method in terms of nfe. In addition, it is preferable in terms of nu for the values of $p = m, 2m$ and $\epsilon \leq 10^{-6}$.

Therefore, it seems desirable to define m as large as possible, $p \approx 2m$ and $\epsilon \approx 10^{-6}$, and examine the criterion (26) only r times at each iteration.

Table 1: nfe/nu required to solve Q20 ($n = 100$), using $m = 5$

$p\backslash\epsilon$	10^{-2}	10^{-4}	10^{-6}	10^{-10}	0
0	655/ 3229	655/ 3229	655/ 3229	655/ 3229	655/ 3229
m	451/ 3668	548/ 5844	603/ 6494	612/ 6653	612/ 5684
$2m$	563/ 4996	567/ 8946	543/ 8974	541/ 8868	541/ 7412
$4m$	588/ 5948	588/14269	546/14908	528/14795	515/11833
$9m$	539/ 5660	470/18320	525/28592	476/26230	442/19764

Table 2: nfe/nu required to solve Q20 ($n = 100$), using $m = 10$

$p\backslash\epsilon$	10^{-2}	10^{-4}	10^{-6}	10^{-10}	0
0	630/ 6053	630/ 6053	630/ 6053	630/ 6053	630/ 6053
m	483/ 6361	427/ 8511	508/10900	511/10892	511/ 9494
$2m$	534/ 7118	426/12308	420/13261	404/13035	404/11038
$4m$	463/ 6059	391/16019	398/20854	383/20713	383/17142
$9m$	493/ 6449	463/28499	360/37478	362/39375	362/32037

Table 3: nfe/nu required to solve Q20 ($n = 100$), using $m = 20$

$p\backslash\epsilon$	10^{-2}	10^{-4}	10^{-6}	10^{-10}	0
0	353/ 6553	353/ 6553	353/ 6553	353/ 6553	353/ 6553
m	298/ 6461	260/ 9547	275/11313	254/10280	254/ 9070
$2m$	289/ 6165	274/13647	238/14371	235/13941	235/11903
$4m$	289/ 6165	240/15498	232/22260	225/23279	225/19362
$9m$	289/ 6165	209/17419	203/38572	187/38061	187/31053

To see if these observations apply to general functions, we applied the modified algorithm with $m = 10$, $p = 21$ and $\epsilon = 10^{-6}$ to the set of 27 tests of Al-Baali [2]. This choice of p (as well as $p = mr + 1$) imply that the first extra update depends on the most recent pair of (2) which seems useful for solving general problems (see Al-Baali [2]). The results are given in Table 5 in the form nfe/nu required to solve each test. For comparison, we also include the results for $p = 0$ and $p = 21$ with $\epsilon = 0$. We observe that, generally, the performance of the modified algorithm with $\epsilon = 10^{-6}$ is substantially better than that of the L-BFGS method in terms of nfe (the average reduction is about 30%). Although the maximum number of updates per iteration is equal to 3.1 times m, the average increase of nu is 2.14 approximately. This result shows the usefulness of the criterion (26) which also prevents using extra updates when L-BFGS works well on some tests (e.g. VARDIM), though the modified algorithm worsens L-BFGS in a few cases. A comparison of the results in the last two columns of Table 5 shows that the performance of the modified algorithm with the choice $\epsilon = 0$ is slightly less efficient than that with the choice $\epsilon = 10^{-6}$.

Table 4: nfe/nu required to solve Q20 ($n = 100$), using $m = 40$

$p\backslash\epsilon$	10^{-2}	10^{-4}	10^{-6}	10^{-10}	0
0	232/ 8195	232/ 8195	232/ 8195	232/ 8195	232/ 8195
m	210/ 7799	185/11734	171/11989	171/11993	171/10700
$2m$	210/ 7799	171/13232	155/16159	165/16735	165/14418
$4m$	210/ 7799	172/14261	152/25605	147/25146	147/21055
$9m$	210/ 7799	167/17717	150/42051	139/46961	139/38453

Depending on these results, we recommend the above modified L-BFGS algorithm with m as large as possible and $p = 2m + 1$, though further experiments are required to choose typical values for m, p, ϵ and the number of times for testing the condition (26).

For future experiments, it is worth introducing the criterion (26) to the modified L-BFGS algorithm of Byrd, Nocedal and Zhu [6], and comparing its performance with that of the modified method of Morales and Nocedal [13].

Acknowledgements. I would like to thank Anton Purnama and Emilio Spedicato for the comments made on a draft of this paper. I would also like to thank Emilio and Lucio Grandinetti for providing Italian university grants (including MURST 1999 cofinanziamento funds and CNR/GNIM funds) which partially supported this research.

References

[1] M. Al-Baali. Extra updates for the BFGS method. *Optimization Methods and Software*, 13:159–179, 2000.

[2] M. Al-Baali. Improved Hessian approximations for the limited memory BFGS methods. *Numerical Algorithms*, 22:99–112, 1999.

[3] B.M. Averick and J.J. Moré. Evaluation of large-scale optimization problems on vector and parallel architectures. *SIAM J. Optimization*, 4:708–721, 1994.

[4] P.T. Boggs and J.W. Tolle. Convergence properties of a class of rank-two updates. *SIAM J. Optimization*, 4:262–287, 1994.

[5] R.H. Byrd, J. Nocedal, and R.B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Math. Programming*, 63:129–156, 1994.

[6] R.H. Byrd, J. Nocedal, and C. Zhu. Towards a discrete Newton method with memory for large scale optimization. In G. Di Pillo and F. Giannessi, editors, *Nonlinear Optimization and Applications*, pages 1–12, Plenum, 1996.

[7] R. Fletcher. Low storage methods for unconstrained optimization. In E.L. Allgower and K. George, editors, *Computational Solution of Nonlinear Systems of Equations, Lectures in Applied Mathematics*, pages 165–179, AMS, Providence, RI, 1990.

[8] R. Fletcher. *Practical Methods of Optimization*. John Wiley, Chichester, England, second edition, 1987.

[9] R. Ge and M.J.D. Powell. The convergence of variable metric matrices in unconstrained optimization. *Math. Programming*, 27:123–143, 1983.

[10] J.C. Gilbert and C. Lemaréchal. Some numerical experiments with variable storage quasi-Newton algorithms. *Math. Programming*, 45:407–436, 1989.

Table 5: nfe/nu, using $m = 10$

Problem	n	$p = 0$	$p = 21$ $\epsilon = 10^{-6}$	$p = 21$ $\epsilon = 0$
EX-ROSBRK	1000	51/ 375	46/ 467	46/ 870
EX-ROSBRKx1	1000	468/ 4321	191/ 5203	191/ 4963
EX-FRDRTH	1000	24/ 170	24/ 264	20/ 307
CH-FRDRTH	1000	54/ 365	37/ 604	43/ 776
EX-POW-SR	1000	80/ 713	60/ 1327	65/ 1213
EX-POW-SRx1	1000	1748/17205	815/24719	929/ 26870
VAR-DIM	1000	53/ 477	53/ 502	53/ 1338
PENLTY I	1000	77/ 662	76/ 887	72/ 1745
EX-ENGVL1	1000	26/ 170	20/ 312	27/ 307
WR-EX-WOOD	1000	44/ 262	33/ 541	37/ 557
TRIGS	1000	84/ 744	73/ 1895	78/ 1932
TRIDB	1000	40/ 293	37/ 781	37/ 745
TRIDT	1000	430/ 4280	386/11262	387/ 11682
SP-MXSQRT	1000	205/ 1943	183/ 5390	183/ 5245
LMS	961	257/ 2425	225/ 6458	223/ 6526
TRIGT	100	144/ 1318	133/ 3720	132/ 3588
VAR(-3.4)	100	696/ 6935	535/16369	528/ 15963
MXSQRT 1	100	443/ 4403	423/12896	392/ 11807
MXSQRT 2	100	534/ 5326	422/12986	439/ 13245
Q12	100	800/ 7991	753/23872	699/ 21338
Q20	100	630/ 6053	395/11806	424/ 12057
CH-ROSBRK	100	560/ 5428	598/17997	598/ 17182
PE	100	1717/17257	992/30882	1356/ 41463
BVP	100	6158/62859	3142/99412	3605/112338
QOR	50	46/ 354	47/ 967	49/ 963
GOR	50	131/ 1195	121/ 3553	129/ 3495
PSP	50	125/ 1082	106/ 2691	110/ 2651

[11] T.G. Kolda, D.P. O'Leary, and L. Nazareth. BFGS with update skipping and varying memory. *SIAM J. Optimization*, 8:1060–1083, 1998.

[12] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming (Series B)*, 45:503–528, 1989.

[13] J.L. Morales and J. Nocedal. *Enriched Methods for Large-Scale Unconstrained Optimization.* Tech. Report, Dept. of Electrical Engineering and Computer Science, Northwestern University, Evanston, USA, 2000.

[14] S.G. Nash and J. Nocedal. A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. *SIAM J. Optimization*, 1:358–372, 1991.

[15] J. Nocedal. *Large Scale Unconstrained Optimization.* Tech. Report, Dept. of Electrical Engineering and Computer Science, Northwestern University, Evanston, USA, 1996.

[16] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comput.*, 35:773–782, 1980.

[17] M.J.D. Powell. Some global convergence properties of a variable metric algorithm for minimization without exact line searches. In R.W. Cottle and C.E. Lemke, editors, *Nonlinear Programming*, pages 53–72, SIAM-AMS Proceedings vol. IX, SIAM Publications, 1976.

[18] X. Zou, I.M. Navon, M. Berger, K.H. Phua, T. Schlick, and F.X. Le Dimet. Numerical experience with limited-memory quasi-Newton and truncated Newton methods. *SIAM J. Optimization*, 3:582–608, 1993.