MODIFIED LIMITED MEMORY BFGS METHOD WITH NONMONOTONE LINE SEARCH FOR UNCONSTRAINED OPTIMIZATION

Gonglin Yuan, Zengxin Wei, and Yanlin Wu

Abstract. In this paper, we propose two limited memory BFGS algorithms with a nonmonotone line search technique for unconstrained optimization problems. The global convergence of the given methods will be established under suitable conditions. Numerical results show that the presented algorithms are more competitive than the normal BFGS method.

1. Introduction

Consider the following unconstrained optimization problem

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

where \( f : \mathbb{R}^n \to \mathbb{R} \) is continuously differentiable. The line search method is one of the most numerical method, which is defined by

\[ x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, \ldots, \]

where \( \alpha_k \) that is determined by a line search is the steplength, and \( d_k \) which determines different line search methods [35, 36, 37, 39, 40, 43, 44, 45, 46, 48, 50] is a search direction of \( f \) at \( x_k \).

One of the most effective methods for unconstrained optimization (1.1) is Newton method. It normally requires a fewest number of function evaluations, and is very good at handling ill-conditioning. However, its efficiency largely depends on the possibility to efficiently solve a linear system which arises when computing the search \( d_k \) at each iteration

\[ G(x_k)d_k = -g(x_k), \]

Received July 17, 2008.
2000 Mathematics Subject Classification. 65H10, 65K05, 90C26.
Key words and phrases. limited memory BFGS method, optimization, nonmonotone, global convergence.
This work is supported by China NSF grands 10761001 and the Scientific Research Foundation of Guangxi University (Grant No. X081082).

©2010 The Korean Mathematical Society

767
where \( g(x_k) = \nabla f(x_k) \) is the gradient of \( f(x) \) at \( x_k \), and \( G(x_k) = \nabla^2 f(x_k) \) is the Hessian matrix of \( f(x) \) at current iteration. Moreover, the exact solution of the system (1.3) could be too burdensome, or is not necessary when \( x_k \) is far from a solution [31]. Inexact Newton methods [8, 31] represent the basic approach underlying most of the Newton-type large-scale algorithms. At each iteration, the current estimate of the solution is updated by approximately solving the linear system (1.3) using an iterative algorithm. The inner iteration is typically “truncated” before the solution to the linear system is obtained.

The limited memory BFGS (L-BFGS) method (see [3]) is an adaptation of the BFGS method for large-scale problems. The implementation is almost identical to that of the standard BFGS method, the only difference is that the inverse Hessian approximation is not formed explicitly, but defined by a small number of BFGS updates. It is often provided a fast rate of linear convergence, and requires minimal storage.

Since the standard BFGS is wildly used to solve general minimization problems, most of the studies concerning limited memory methods are concentrate on the L-BFGS method. We know that, the BFGS update exploits only the gradient information, while the information of function values available is neglected. Therefore, many efficient attempts have been made to modify the usual quasi-Newton methods using both the gradient and function values information (e.g. [41, 51]). Lately, in order to get a higher order accuracy in approximating the second curvature of the objective function, Wei, Li, and Qi [41], and Zhang, Deng, and Chen [51] proposed modified BFGS-type methods for (1.1), and the reported numerical results show that the average performance is better than that of the standard BFGS method, respectively.

The monotone line search technique is often used to get the stepsize \( \alpha_k \), however monotonicity may cause a series of very small steps if the contours of objective function are a family of curves with large curvature [18]. More recently, the nonmonotonic line search for solving unconstrained optimization is proposed by Grippo et al. in [18]. Han and Liu [21] presented a new nonmonotone BFGS method for (1.1). The global convergence of the convex objective function was established. Numerical results show that this method is more competitive to the normal BFGS method with monotone line search. We [49] proved its superlinear convergence.

Motivated by the above observation, we propose two limited memory BFGS-type method on the basic of Wei et al. [41], Zhang et al. [51], and [21], respectively, which are suitable for solving large-scale unconstrained optimization problems. The major contribution of this paper is an extension of the BFGS-type method in [41], [51], and the nonmonotone line search technique to limited memory scheme. The triple of the standard L-BFGS method \( \{s_i, y_i\} \), \( i = k - \bar{m} + 1, \ldots, k \), is stored, where \( s_i = x_{i+1} - x_i \), \( y_i = g_{i+1} - g_i \), \( g_i = g(x_i) \) and \( g_{i+1} = g(x_{i+1}) \) are the gradient of \( f(x) \) at \( x_i \) and \( x_{i+1} \), respectively, \( \bar{m} > 0 \) is a constant. A distinguishing feature of our proposed L-BFGS method is that,
at each iteration, a triple
\[ \{s_i, y_i, A_i\}, \quad i = k - \tilde{m} + 1, \ldots, k, \]
is stored, where \( A_i \) is a scalar related to function value. Compared with the standard BFGS method, at each iteration, the proposed method requires no more function or derivative evaluations, and hardly more storage or arithmetic operations. Under suitable conditions, we establish the global convergence of the method. The numerical experiments of the proposed method on a set of large problems indicate that it is interesting.

This paper is organized as follows. In the next section, modified BFGS update and nonmonotone line search are stated. The proposed L-BFGS algorithms are given in Section 3. Under some reasonable conditions, the global convergence of the given methods is established in Section 4. Numerical results and a conclusion are presented in Section 5 and in Section 6, respectively.

2. Modified BFGS update and nonmonotone line search

Quasi-Newton methods are iterative methods of the form
\[ x_{k+1} = x_k + \alpha_k d_k, \]
where \( x_k \) is the \( k \)th iteration point, \( \alpha_k \) is a stepsize, and \( d_k \) is a search direction. Now we first state the search direction as follows.

2.1. Some modified BFGS update formulas

The search direction of the quasi-Newton method is defined by
\[ B_k d_k + g_k = 0, \]
where \( g_k = g(x_k) = \nabla f(x_k) \) is the gradient of \( f(x) \) at \( x_k \), \( B_k \) is an approximation of \( \nabla^2 f(x_k) \).

By tradition, \( \{B_k\} \) satisfies the following quasi-Newton equation
\[ B_{k+1} s_k = y_k, \]
where \( s_k = x_{k+1} - x_k = \alpha_k d_k \), \( y_k = g_{k+1} - g_k \). Throughout the paper, we use these notations: \( \| \cdot \| \) is the Euclidean norm, \( g(x_k) \) and \( g(x_{k+1}) \) are replaced by \( g_k \) and \( g_{k+1} \), and \( f(x_k) \) and \( f(x_{k+1}) \) are replaced by \( f_k \) and \( f_{k+1} \) respectively.

The famous update \( B_k \) is the standard BFGS formula
\[ B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \]
Let \( H_k \) be the inverse of \( B_k \). Then the inverse update formula of (2.3) method is represented as
\[ H_{k+1} = H_k - \frac{y_k (s_k - H_k y_k) s_k^T}{(y_k^T s_k)^2} s_k + (s_k - H_k y_k) s_k^T + s_k (s_k - H_k y_k)^T}{(y_k^T s_k)^2}(y_k^T s_k)^T + s_k s_k^T y_k^T s_k, \]
\[ = \left( I - \frac{s_k y_k^T}{y_k^T s_k} \right) H_k \left( I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}. \]
which is the dual form of the DFP update formula in the sense that $H_k \leftrightarrow B_k$, $H_{k+1} \leftrightarrow B_{k+1}$, and $s_k \leftrightarrow y_k$. It has been shown that the BFGS method is very efficient for solving unconstrained optimization problems (1.1) [11, 14, 47]. For convex minimization problems, the BFGS method are global convergence if the exact line search or some special inexact line search is used [1, 2, 4, 12, 16, 32, 33, 38] and its local convergence has been well established [9, 10, 17]. For general function $f$, Dai [5] have constructed an example to show that the standard BFGS method may fail for non-convex functions with inexact line search, Mascarenhas [29] showed that the nonconvergence of the standard BFGS method even with exact line search.

In order to obtain a global convergence of BFGS method without convexity assumption on the objective function, Li and Fukushima [22, 23] made a slight modification to the standard BFGS method. Now we state their works as follows:

(i) A new quasi-Newton equation [22] with following form

$$B_{k+1}s_k = y^*_k,$$

where $y^*_k = y_k + (\max\{0, -\frac{y_k^Ts_k}{\|s_k\|^2}\} + \phi(\|y_k\|))s_k$, and function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ satisfies: (a) $\phi(t) > 0$ for all $t > 0$; (b) $\phi(t) = 0$ if and only if $t = 0$; (c) $\phi(t)$ is bounded if $t$ is in a bounded set.

(ii) A modified BFGS update formula [23] with following form

$$B_{k+1} = \begin{cases} B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y^*_k y^*_k - y_k^T y_k}{\|s_k\|^2}, & \text{if } s_k^T y^*_k \geq \phi(\|y_k\|) \geq 0, \\ B_k, & \text{otherwise.} \end{cases}$$

Then it is not difficult to see that $s_k^T y^*_k > 0$ always holds, which can ensure that the update matrix $B_{k+1}$ inherits the positive definiteness of $B_k$ (see [14]). The global convergence and the superlinear convergence of these two methods for nonconvex have been established under appropriate conditions (see [22, 23] in detail).

In order to get a better approximation of the objective function Hessian matrix, Wei, Li, and Qi [41] and Zhang, Deng, and Chen [51] proposed modified quasi-Newton equations which are given as follows. (i) The equation of Wei, Li, and Qi [41]:

$$B_{k+1}s_k = y^*_k = y_k + A_k s_k,$$

where

$$A_k = \frac{2[f(x_k) - f(x_{k-1})] + [g(x_{k+1}) + g(x_k)]^T s_k}{\|s_k\|^2}.$$

They replaced all the $y_k$ in (2.3), and obtained the following modified BFGS-type update formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y^*_k y^*_k - y_k^T y_k}{\|s_k\|^2} s_k.$$
Note that this quasi-Newton equation (2.6) contains both gradient and function value information at the current and the previous step, one may argue that the resulting methods will really outperform than the original method. In fact, the practical computation shows that this method is better than the normal BFGS method (see [41, 42] for detail) for some given problems [30]. Furthermore, some theoretical advantages of the new quasi-Newton equation (2.6) can be seen from the following two theorems.

**Theorem 2.1 ([42, Lemma 3.1]).** Considering the quasi-Newton equation (2.6). Then we have for all $k \geq 1$

$$f(x_k) = f(x_{k+1}) + g(x_{k+1})^T (x_k - x_{k+1}) + \frac{1}{2}(x_k - x_{k+1})^T B_{k+1}(x_k - x_{k+1}).$$

**Theorem 2.2 ([24, Theorem 3.1]).** Assume that the function $f(x)$ is sufficiently smooth and $\|s_k\|$ is sufficiently small. Then we have

(2.9) $s_k^T G_{k+1} s_k - s_k^T y_k^{2*} - \frac{1}{3} s_k^T (T_{k+1} s_k) s_k = O(\|s_k\|^4)$

and

(2.10) $s_k^T G_{k+1} s_k - s_k^T y_k - \frac{1}{2} s_k^T (T_{k+1} s_k) s_k = O(\|s_k\|^4),$

where $G_{k+1}$ denotes the Hessian matrix of $f$ at $x_{k+1}$, $T_{k+1}$ is the tensor of $f$ at $x_{k+1}$, and

$$s_k^T (T_{k+1} s_k) s_k = \sum_{i,j,l=1}^{n} \frac{\partial^3 f(x_{k+1})}{\partial x^i \partial x^j \partial x^l} s_k^i s_k^j s_k^l.$$

(iii) The equation of Zhang, Deng, and Chen [51]:

(2.11) $B_{k+1} s_k = y_{k}^{3*} = y_k + \bar{A}_k s_k,$

where

(2.12) $\bar{A}_k = \frac{6[f(x_k) - f(x_{k+1})] + 3[g(x_{k+1}) + g(x_{k})]^T s_k}{\|s_k\|^2}.$

They replaced all the $y_k$ in (2.3), and obtained the following modified BFGS-type update formula

(2.13) $B_{k+1} = B_k - \frac{B_k s_k y_k^T}{s_k^T B_k s_k} + \frac{y_k^{3*} y_k^{3* T}}{\|y_k^{3*}\|^2} s_k.$

Similar to equation (2.6), this quasi-Newton equation (2.11) contains both gradient and function value information at the current and the previous step, one may argue that the resulting methods will really outperform than the original method. In fact, the practical computation shows that this method is better than the normal BFGS method (see [51] for detail). Furthermore, some theoretical advantages of the new quasi-Newton equation (2.11) can be seen from the following theorem.
Theorem 2.3 ([51, Theorem 3.3]). Assume that the function $f(x)$ is sufficiently smooth and $\|s_k\|$ is sufficiently small. Then we have
\begin{equation}
    s_k^T (G_{k+1}s_k - y_k^*) = O(\|s_k\|^4)
\end{equation}
and
\begin{equation}
    s_k^T (G_{k+1}s_k - y_k) = O(\|s_k\|^4).
\end{equation}

It is not difficult to deduce that $s_k^T y_k > 0$ holds for the uniformly convex function $f$ (or see [42]). We all know that the condition $s_k^T y_k > 0$ can ensure that the update matrix $B_{k+1}$ from (2.8) inherits the positive definiteness of $B_k$. Similarly, in order to get the positive definiteness of $B_k$ in (2.13) for each $k$, we give a modified BFGS update of (2.13), i.e., the modified update formula is defined by
\begin{equation}
    B_{k+1} = B_k - s_k^T B_k s_k + y_k^* y_k^T
\end{equation}
where $y_k^* = y_k + A_k^* s_k$, $A_k^* = \frac{1}{3} \max \{ A_k, 0 \}$. Then the corresponding quasi-Newton equation is
\begin{equation}
    B_{k+1}s_k = y_k^*.
\end{equation}
From the definition of $y_k^*$, we can obtain $s_k^T y_k^* > 0$ if the objective function $f$ is uniformly convex function (see Lemma 4.1).

There are other modified formulas that can be seen from [7, 34] in detail, here we do not present anymore.

2.2. One nonmonotone line search

Normally the steplength $\alpha_k$ is generated by the following weak Wolfe-Powell (WWP): Find a steplength $\alpha_k$ such that
\begin{equation}
    f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma_1 \alpha_k g_k^T d_k,
\end{equation}
\begin{equation}
    g(x_k + \alpha_k d_k)^T d_k \geq \sigma_2 g_k^T d_k,
\end{equation}
where $0 < \sigma_1 < \sigma_2 < 1$. Many authors analysis the BFGS algorithm from generalizing line search procedures [25, 26]. Recently, the nonmonotone line search technique for unconstrained optimization is proposed by Grippo et. al. [18, 19, 20] and further studied by [27, 28] etc.. Grippo, Lamparillo, and Lucidi [18] proposed the following nonmonotone line search we call GLL line search.

GLL line search: Select steplength $\alpha_k$ satisfying
\begin{equation}
    f(x_{k+1}) \leq \max_{0 \leq j \leq n(k)} f(x_{k-j}) + \varepsilon_1 \alpha_k g_k^T d_k,
\end{equation}
\begin{equation}
    g(x_{k+1})^T d_k \geq \max \{ \varepsilon_2, 1 - (\alpha_k \|d_k\|^p) \} g_k^T d_k,
\end{equation}
where $p \in (-\infty, 1)$, $k = 0, 1, 2, \ldots$, $\varepsilon_1, \varepsilon_2 \in (0, 1)$, $n(k) = \min \{ H, k \}$, $H \geq 0$ is an integer constant. Combining this line search and the normal BFGS formula (2.3), Han and Liu [21] established the global convergence of the convex
objective function. Numerical results show that this method is more competitive than the normal BFGS method with WWP line search. Recently, the superlinear convergence of the new nonmonotone BFGS algorithm for convex function was proved by Yuan and Wei [49].

3. Limited memory BFGS-type method

In this section, we propose new algorithm to solve (1.1). To improve the performance of the line search, it is a better choice to use the GLL line search instead of the WWP line search. Then our method generates a sequence of points \( \{x_k\} \) by

\[
x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, \ldots,
\]

where \( \alpha_k \) is determined by (2.20) and (2.21), \( d_k \) is a descent direction of \( f \) at \( x_k \). In the following, we state the direction \( d_k \) in details.

The limited memory BFGS (L-BFGS) method (see [3]) is an adaptation of the BFGS method for large-scale problems. In the L-BFGS method, matrix \( H_k \) is obtained by updating the basic matrix \( H_0 \bar{m} > 0 \) times using BFGS formula with the previous \( \bar{m} \) iterations. The standard BFGS correction (2.4) has the following form

\[
H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T,
\]

where \( \rho_k = \frac{1}{s_k^T y_k} \), \( V_k = I - \rho_k y_k s_k^T \), \( I \) is the unit matrix. Thus, \( H_{k+1} \) in the L-BFGS method has the following form:

\[
\begin{align*}
H_{k+1} & = V_k^T H_k V_k + \rho_k s_k s_k^T \\
& = V_k^T [V_k^T H_{k-1} V_{k-1} + \rho_{k-1} s_{k-1} s_{k-1}^T] V_k + \rho_k s_k s_k^T \\
& = \ldots \\
& = [V_k^T \cdots V_{k-\bar{m}+1}^T] H_{k-\bar{m}+1} [V_{k-\bar{m}+1} \cdots V_k] \\
& + \rho_{k-\bar{m}+1} [V_{k-\bar{m}+1}^T \cdots V_{k-\bar{m}+2}^T] s_{k-\bar{m}+1} s_{k-\bar{m}+1}^T V_{k-\bar{m}+2} \cdots V_k \\
& + \cdots \\
& + \rho_k s_k s_k^T.
\end{align*}
\]

(3.23)

To improve the performance of the standard limited memory BFGS algorithm, it is a better choice use the modified BFGS-type update instead of the standard BFGS. If we replaced all the \( y_k \) with \( y_k^* \) and \( y_k^* \) in (3.23) respectively, the new limited memory BFGS-type update can be obtained by

\[
\begin{align*}
H_{k+1} & = V_k^* T H_k V_k^* + \rho_k^* s_k s_k^T \\
& = V_k^* T [V_k^* T H_{k-1} V_{k-1}^* + \rho_{k-1}^* s_{k-1} s_{k-1}^T] V_k^* + \rho_k^* s_k s_k^T \\
& = \ldots \\
& = [V_k^* \cdots V_{k-\bar{m}+1}^* T] H_{k-\bar{m}+1} [V_{k-\bar{m}+1}^* \cdots V_k^*] \\
& + \rho_{k-\bar{m}+1} [V_{k-\bar{m}+1}^* T \cdots V_{k-\bar{m}+2}^* T] s_{k-\bar{m}+1} s_{k-\bar{m}+1}^T V_{k-\bar{m}+2} \cdots V_k^* \\
& + \cdots
\end{align*}
\]
Step 6: Let $H_{k+1} = V_k^{**T} H_k V_k^{**} + \bar{\rho}_k s_k s_k^T$
and
\[
H_{k+1} = V_k^{**T} H_k V_k^{**} + \bar{\rho}_k s_k s_k^T \\
= V_k^{**T} [V_k^{**T} H_{k-1} V_k^{**} + \bar{\rho}_{k-1} s_{k-1} s_{k-1}^T] V_k^{**} + \bar{\rho}_k s_k s_k^T \\
= \ldots \\
= [V_k^{**T} \cdots V_{k-m_1+1}^{**T}] H_{k-m_1+1} [V_{k-m_1+1}^{**T} \cdots V_k^{**}] \\
+ \bar{\rho}_k \tilde{m}_1 [V_{k-m_1+1}^{**T} \cdots V_k^{**}] s_{k-m_1+1} s_{k-m_1+1}^T [V_{k-m_1+2}^{**} \cdots V_k^{**}] \\
+ \ldots \\
+ \bar{\rho}_k s_k s_k^T,
\]
where $\rho^*_k = \frac{1}{s_k^T g_k}$, $V_k^{**} = I - \rho^*_k y_k^* s_k^T$, and $\bar{\rho}_k = \frac{1}{s_k^T g_k}$, $V_k^{**} = I - \bar{\rho}_k^* s_k^T$.

Now we state the new limited memory BFGS-type algorithm (L-BFGS-A)
with GLL line search as follows.

**Algorithm 1. (L-BFGS-A1)**
Step 0: Choose an initial point $x_0 \in \mathbb{R}^n$, an basic symmetric positive definite matrix $H_0 \in \mathbb{R}^{n \times n}$, and constants $r, \varepsilon_1, \varepsilon_2 \in (0, 1), p \in (-\infty, 1)$, $H \geq 0$, an positive integer $m_1$. Let $k := 0$;
Step 1: Stop if $\|g_k\| = 0$.
Step 2: Determine $d_k$ by
\[
d_k = -H_k g_k.
\]
Step 3: Find $\alpha_k$ satisfying (2.20) and (2.21).
Step 4: Let the next iterative be $x_{k+1} = x_k + \alpha_k d_k$.
Step 5: Let $\tilde{m} = \min \{k+1, m_1\}$. Update $H_0$ for $\tilde{m}$ times to get $H_{k+1}$ by (3.24).

**Algorithm 11. (L-BFGS-A11)**
Step 5: Let $\tilde{m} = \min \{k+1, m_1\}$. Update $H_0$ for $\tilde{m}$ times to get $H_{k+1}$ by (3.25).
In the following, we assume that the algorithm updates $B_k$—the inverse of $H_k$. We also assume that the basic matrix $B_0$, and its inverse $H_0$, are bounded and positive definite. The Algorithm 1 with $B_k$ can be stated as follows.

**Algorithm 2. (L-BFGS-A2)**
Step 2: Determine $d_k$ by
\[
d_k = -B_k g_k.
\]
Step 5: Let $\tilde{m} = \min \{k+1, m_1\}$. Put $s_k = x_{k+1} - x_k = \alpha_k d_k$, $y_k = g_{k+1} - g_k$.
Update $B_0$ for $\tilde{m}$ times, i.e., for $l = k - \tilde{m} + 1, \ldots, k$ compute
\[
B^{l+1}_k = B^l_k - \frac{B^l_k s_l s_l^T B^l_k}{s_l^T B^l_k s_l} + \frac{y_l^4 y_l^{4*}}{y_l^4 y_l^{4*}}.
\]
where $s_l = x_{l+1} - x_l$, $y_l^4 = y_l + A^*_l s_l$, and $B^{k-\tilde{m}+1}_k = B_0$ for all $k$. 

Algorithm 22. (L-BFGS-A22)

Step 2: Determine $d_k$ by

$$B_k d_k = -g_k.$$  \hfill (3.29)

Step 5: Let $\tilde{m} = \min\{k + 1, m_1\}$. Put $s_k = x_{k+1} - x_k = \alpha_k d_k$, $y_k = g_{k+1} - g_k$.

Update $B_0$ for $\tilde{m}$ times, i.e., for $l = k - \tilde{m} + 1, \ldots, k$ compute

$$B_{k+1}^l = B_k - \frac{B_k^{l-1} s_l s_l^T B_k^{l-1}}{s_l^T B_k^{l-1} s_l} y_l^{2*} y_l^{2*T},$$  \hfill (3.30)

where $s_l = x_{l+1} - x_l$, $y_l^{2*} = y_l + A_l s_l$, and $B_{k-\tilde{m}+1}^k = B_0$ for all $k$.

Note that Algorithms 1 and 2 are mathematically equivalent, and Algorithms 11 and 22 are mathematically equivalent too. In our numerical experiments we implement Algorithms 1 and 11, and Algorithms 2 and 22 are given only for the purpose of analysis. Throughout this paper, we only discuss Algorithms 2 and 22. In the following section, we will concentrate on their global convergence.

4. Convergence analysis

This section is devoted to show that Algorithm 2 is convergent on twice continuously differentiable and uniformly convex function. In order to establish global convergence for Algorithm 2, we need the following assumptions.

**Assumption A.** (i) The level set $\Omega = \{x \mid f(x) \leq f(x_0)\}$ is bounded.

(ii) The function $f$ is twice continuously differentiable on $\Omega$.

(iii) The function $f$ is uniformly convex, i.e., there exist positive constants $m$ and $M$ such that

$$m \|d\|^2 \leq d^T G(x) d \leq M \|d\|^2$$  \hfill (4.1)

holds for all $x \in \Omega$ and $d \in \mathbb{R}^n$, where $G(x) = \nabla^2 f(x)$. These assumptions are the same as those in [42, 51].

It is obvious that Assumption A implies that there exists a constant $M_* > 0$ such that

$$\|G(x)\| \leq M_*, \ x \in \Omega.$$  

Assumption A (ii) implies that there exists a constant $L \geq 0$ satisfying

$$\|g(x) - g(y)\| \leq L \|x - y\|, \ x, y \in \Omega.$$  \hfill (4.2)

**Lemma 4.1.** Let Assumption A hold. Then there exists a positive number $M_1$ such that

$$\left\|y_k^{2*}\right\| \leq M_1, \ k = 0, 1, 2, \ldots.$$  

**Proof.** Following the definition of $y_k^{2*}$ and the Taylor’s formula, we get

$$s_k^T y_k^{2*} = s_k^T y_k + s_k^T A_k^* s_k$$

$$= \max\{2[f_k - f_{k+1}] + 2g_{k+1}^T s_k, s_k^T y_k\}$$
To begin with, we take the determinant in both sides of (2.8)
\[
\det(B_k + \theta(x_{k+1} - x_k)s_k)
\]
where \(\theta, \theta_1 \in (0, 1)\). Combining with Assumption A(iii), it is easy to obtain
\[
m\|s_k\|^2 \leq s_k^* y_k^* \leq M\|s_k\|.
\]
By the definition of \(y_k^*\) and the Taylor’s formula again, we have
\[
\|y_k^*\| = \left\| y_k + \max \left\{ \frac{2[f(x_k) - f(x_{k+1})] + [g(x_{k+1}) + g(x_k)]^T s_k}{\|s_k\|^2}, 0 \right\} s_k \right\|
\]
\[
\leq \max \left\{ \|y_k\| + \frac{\|s_k\|}{\|s_k\|^2} \right\},
\]
\[
\leq 2\|y_k\| + \frac{s_k^T G(x_k + \theta(x_{k+1} - x_k)s_k)}{\|s_k\|^2}
\]
\[
\leq 2\|s_k\| + M\|s_k\|
\]
(4.4) \(= (2L + M)\|s_k\|\),
where \(\theta \in (0, 1)\), the third inequality follows (4.1) and (4.2). By (4.3) and (4.4), we get
\[
\|y_k^*\|^2 \leq \frac{(2L + M)^2\|s_k\|^2}{m\|s_k\|^2} = \frac{(2L + M)^2}{m} = M_1.
\]
The proof is complete. \(\square\)

**Lemma 4.2.** Let \(B_k\) be generated by (3.28). Then we have
\[
\det(B_{k+1}) = \det(B_k^{k+m+1}) \prod_{l=k-m+1}^{k} \frac{s_l^T y_l^*}{s_l^T B_l s_l},
\]
where \(\det(B_k)\) denotes the determinant of \(B_k\).

**Proof.** To begin with, we take the determinant in both sides of (2.8)
\[
\det(B_{k+1}) = \det \left( B_k \left( I - \frac{s_k^T B_k s_k}{s_k B_k s_k} + \frac{B_k^{-1} y_k^* y_k^* T}{s_k y_k^*} \right) \right)
\]
\[
= \det(B_k) \det \left( I - \frac{s_k^T B_k s_k}{s_k^T B_k s_k} + \frac{B_k^{-1} y_k^* y_k^* T}{s_k y_k^*} \right)
\]
\[
= \det(B_k) \left( (1 - s_k^T B_k s_k) \left( 1 + (B_k^{-1} y_k^* y_k^* T) \frac{y_k^*}{y_k^*} \right) \right.
\]
\[
- \left. (s_k^T y_k^* y_k^* T) \frac{(B_k s_k)^T}{s_k B_k s_k} \right)
\]
\[ \det(B_k) \frac{y_k^4 s_k}{s_k^4 B_k s_k}, \]

where the third equality follows from the formula (see, e.g., [9, Lemma 7.6])

\[ \det(I + u_1 u_2^T + u_3 u_4^T) = (1 + u_1^T u_2)(1 + u_3^T u_4) - (u_1^T u_4)(u_2^T u_3). \]

Therefore, there is also a simple expression for the determinant of (3.28)

\[ \det(B_{k+1}) = \det(B_k^{k-m+1}) \prod_{l=k-m+1}^{k} \frac{s_l^T y_l^4}{s_l^4 B_l s_l}. \]

Then we complete the proof. \(\square\)

Define the length of the orthogonal projection of \(-g_k\) on \(d_k\) by

\[ \eta_k = \frac{-g_k^T d_k}{\|d_k\|}. \]

The following Lemmas 4.3-4.6 has been proved in [21], here we only state them as follows, but omit the proof.

**Lemma 4.3.** Let Assumption A be satisfied. Consider GLL line search. Then there exists a positive constant \(b_0\) such that

\[ \|s_k\| \geq b_0 \min\{\eta_k, (\eta_k)^{-\frac{1}{2}}\}, \]

where \(\eta_k\) is defined by (4.6).

**Lemma 4.4.** Denote that

\[ f(x_{l(k)}) = \max_{0 \leq j \leq n(k)} f(x_{k-j}), k - n(k) \leq l(k) \leq k. \]

If \(f_{k+1} \leq f(x_{l(k)}), k = 0, 1, 2, \ldots,\) then the sequence \(\{f(x_{l(k)})\}\) monotonically decreases, and \(x_k \in \Omega\) for all \(k \geq 0.\)

**Lemma 4.5.** If

\[ f_{k+1} \leq f(x_{l(k)}) - t_k, k = 0, 1, 2, \ldots, \]

where \(t_k \geq 0,\) then

\[ \sum_{k=0}^{\infty} \min_{0 \leq j \leq n(k)} t_{k+n(k)-j} < +\infty. \]

**Lemma 4.6.** If the sequence of nonnegative numbers \(m_k(k = 0, 1, \ldots)\) satisfies

\[ \prod_{j=0}^{k} m_j \geq c_1^k, c_1 > 0, k = 1, 2, \ldots, \]

then \(\lim \sup_k m_k > 0.\)
Lemma 4.7. Let \( \{x_k\} \) be generated by Algorithm 2 and Assumption A hold. If
\[
\liminf_{k \to \infty} \|g_k\| > 0,
\]
then there exists a constant \( \epsilon_0 > 0 \) such that
\[
\prod_{j=0}^{k} \eta_j \geq (\epsilon_0)^{k+1} \text{ for all } k \geq 0.
\]

Proof. Assume that \( \liminf_{k} \|g_k\| > 0 \), i.e., there exists a constant \( c_2 > 0 \) such that
\[
(4.10) \quad \|g_k\| \geq c_2, \quad k = 0, 1, 2, \ldots
\]
From Assumption A(iii) and Taylor’s formula, we have
\[
(4.11) \quad m\|s_k\|^2 \leq s_k^T G(x_k + \theta_1 s_k)s_k = s_k^T y_k \leq M\|s_k\|^2,
\]
combining with (4.3), we get
\[
(4.12) \quad m M s_k^T y_k \leq s_k^T y^*_k \leq M m s_k^T y_k.
\]
Taking the trace operation in both sides of (3.28), we get
\[
(4.13) \quad \text{Tr}(B_{k+1}^+) = \text{Tr}(B_{k-m+1}^+) - \sum_{l=k-m+1}^{k} \frac{\|B_l s_l\|^2}{s_l^T B_l s_l} + \sum_{l=k-m+1}^{k} \frac{\|y^*_l\|^2}{s_l^T y^*_l},
\]
where \( \text{Tr}(B_k) \) denotes the trace of \( B_k \). Repeating this trace operation, we have
\[
\text{Tr}(B_{k+1}) = \text{Tr}(B_{k-m+1}^+) - \sum_{l=k-m+1}^{k} \frac{\|B_l s_l\|^2}{s_l^T B_l s_l} + \sum_{l=k-m+1}^{k} \frac{\|y^*_l\|^2}{s_l^T y^*_l} = \ldots
\]
\[
(4.14) \quad = \text{Tr}(B_0) - \sum_{l=0}^{k} \frac{\|B_l s_l\|^2}{s_l^T B_l s_l} + \sum_{l=0}^{k} \frac{\|y^*_l\|^2}{s_l^T y^*_l}.
\]
Combining (4.10), (4.14), (3.26), (3.29), and Lemma 4.1, we obtain
\[
(4.15) \quad \text{Tr}(B_{k+1}) \leq \text{Tr}(B_0) - \sum_{j=0}^{k} \frac{c_2^2}{g_j^T H_j g_j} + (k + 1)M_1.
\]
Using \( B_{k+1} \) is positive definite, we have \( \text{Tr}(B_{k+1}) > 0 \). By (4.15), we obtain
\[
(4.16) \quad \sum_{j=0}^{k} \frac{c_2^2}{g_j^T H_j g_j} \leq \frac{\text{Tr}(B_0) + (k + 1)M_1}{c_2^2}
\]
and
\[
(4.17) \quad \text{Tr}(B_{k+1}) \leq \text{Tr}(B_0) + (k + 1)M_1.
\]
By the geometric-arithmetic mean value formula we get
\[
(4.18) \quad \prod_{j=0}^{k} g_j^T H_j g_j \geq \left[ \frac{(k+1)\varepsilon_2^2}{\text{Tr}(B_0) + (k+1)M_1} \right]^{k+1}.
\]
Using Lemma 4.2, (4.12), and (2.21), we have
\[
\det(B_{k+1}) = \det(B_k^{-\bar{m}+1}) \prod_{l=k-\bar{m}+1}^{k} \frac{s_l^T y_l^*}{s_l^T B_l s_l}
\geq \det(B_k^{-\bar{m}+1}) \prod_{l=k-\bar{m}+1}^{k} \frac{m}{M} \frac{s_l^T y_l}{s_l^T B_l s_l}
\geq \det(B_k^{-\bar{m}+1}) \prod_{l=k-\bar{m}+1}^{k} \frac{m \min\{1-\varepsilon_2, \|s_l\|^p\}}{\alpha_l}
\geq \cdots
\geq \det(B_0) \left[ \frac{m}{M} \right]^{k+1} \prod_{j=0}^{k} \frac{\min\{1-\varepsilon_2, \|s_j\|^p\}}{\alpha_j},
\]
which implies
\[
(4.19) \quad \frac{\det(B_0)}{\det(B_{k+1})} \leq \left[ \frac{m}{M} \right]^{k+1} \prod_{j=0}^{k} \max \left\{ \frac{\alpha_j}{1-\varepsilon_2}, \frac{\alpha_j}{\|s_j\|^p} \right\}.
\]
By using the geometric-arithmetic mean value formula again, we get
\[
(4.20) \quad \det(B_{k+1}) \leq \left[ \frac{\text{Tr}(B_{k+1})}{n} \right]^n.
\]
Using (4.17), (4.19) and (4.20), we obtain
\[
\prod_{j=0}^{k} \max \left\{ \frac{\alpha_j}{1-\varepsilon_2}, \frac{\alpha_j}{\|s_j\|^p} \right\} \geq \left[ \frac{m}{M} \right]^{k+1} \frac{\det(B_0)n^n}{\text{Tr}(B_0) + (k+1)M_1}^{n^n}
\geq \left[ \frac{m}{M} \right]^{k+1} k+1 \left[ \frac{\det(B_0)n^n}{\text{Tr}(B_0) + M_1} \right]^{n^n}
\geq \left[ \frac{m}{M} \right]^{k+1} \left( \frac{1}{\exp(n)} \right)^{k+1} \min \left\{ \frac{\det(B_0)n^n}{\text{Tr}(B_0) + M_1}^{n^n}, 1 \right\}
\geq \left[ \frac{M}{\exp(n)m} \right]^{k+1} \min \left\{ \frac{\det(B_0)n^n}{\text{Tr}(B_0) + M_1}^{n^n}, 1 \right\}
\geq c_3^{k+1},
\]
where \( c_3 \leq \frac{M}{\exp(n)m} \min \{ \frac{\det(B_0)n^n}{\text{Tr}(B_0) + M_1}^{n^n}, 1 \}. \) Let
\[
\cos \theta_j = \frac{-g_j^T d_j}{\|g_j\| \|d_j\|}.
\]
Multiplying (4.18) with (4.21), we get for all \( k \geq 0 \)
\[
\prod_{j=0}^{k} \max \left\{ \frac{\| s_j \| \| g_j \| \cos \theta_j}{1-\varepsilon_2}, \frac{\| g_j \| \cos \theta_j}{\| s_j \|^{p-1}} \right\} \geq c_3^{k+1} \left[ \frac{(k+1)c_2^2}{\text{Tr}(B_0) + (k+1)M_1} \right]^{k+1} \geq \left[ c_3 c_2^2 \frac{\text{Tr}(B_0) + M_1}{M_1} \right]^{k+1}.
\]  
(4.22)

By
\[
\prod_{j=0}^{k} \max \left\{ \frac{\| s_j \| \| g_j \| \cos \theta_j}{1-\varepsilon_2}, \frac{\| g_j \| \cos \theta_j}{\| s_j \|^{p-1}} \right\} \leq \left( \frac{1}{1-\varepsilon_2} \right)^{k+1} \prod_{j=0}^{k} \max \{ \| s_k \|, \| s_k \|^{1-p} \} \| g_j \| \cos \theta_j,
\]
we have
\[
\prod_{j=0}^{k} \max \{ \| s_j \|, \| s_j \|^{1-p} \} \| g_j \| \cos \theta_j \geq \left( \frac{1-\varepsilon_2}{c_2} \right)^{k+1} \left[ \frac{(k+1)c_2^2}{\text{Tr}(B_0) + M_1} \right]^{k+1}.
\]
(4.23)

According to Lemma 4.4 and Assumption A we know that there exists a constant \( M_2' > 0 \) such that
\[
\| s_k \| = \| x_{k+1} - x_k \| \leq \| x_{k+1} \| + \| x_k \| \leq 2M_2'.
\]
(4.24)

Combining (4.23) and (4.24), and noting that \( \| g_j \| \cos \theta_j = \eta_j \), we get for all \( k \geq 0 \)
\[
\prod_{j=0}^{k} \eta_j \geq \left[ \frac{(1-\varepsilon_2) c_3 c_2^2}{\text{Tr}(B_0) + M_1} \max \{ 2M_2', 1, (2M_2')^{1-p} \} \right]^{k+1} = \epsilon_{k+1}^{k+1}.
\]
The proof is complete. \( \square \)

Now we establish the global convergence theorem for Algorithm 2.

**Theorem 4.1.** Let Assumption A hold and the sequence \( \{ x_k \} \) be generated by Algorithm 2. Then we have
\[
\liminf_{k \to \infty} \| g_k \| = 0.
\]
(4.25)

**Proof.** By Lemma 4.3 and (2.20), we get
\[
f_{k+1} \leq f(x_{l(k)}) - \varepsilon_1 \| s_k \| \eta_k \leq f(x_{l(k)}) - \varepsilon_1 b_0 \min \{ \eta_k^2, \eta_k^{\frac{2-p}{p}} \}.
\]
(4.26)

Let \( t_k = \varepsilon_1 b_0 \min \{ \eta_k^2, \eta_k^{\frac{2-p}{p}} \} \). By Lemma 4.5, we have
\[
\sum_{k=0}^{\infty} \min \{ \eta_k^2, \eta_k^{\frac{2-p}{p}} \} \min \{ \eta_{k+n(k)-j}^2, \eta_{k+n(k)-j}^{\frac{2-p}{p}} \} < +\infty.
\]
\[
\sum_{q=1}^{\infty} \min_{0 \leq j \leq n(k)} \min \{ \eta_j^{(n(k)+1)q+n(k)-j}, \eta_j^{(n(k)+1)q+n(k) - j}\} < +\infty.
\]

Denoting the sequence \(\{p(q)\}\) as follows:

\[
\min \{ \eta_j^{p(q)}, \eta_j^{p(q)} \} = \min_{0 \leq j \leq n(k)} \min \{ H_{1j}(q), H_{2j}(q) \},
\]

\[
H_{1j}(q) = \eta_j^{(n(k)+1)q+n(k)-j},
\]

\[
q(n(k)+1) \leq p(q) \leq (q+1)n(k) + q.
\]

Therefore

\[
p(1) < p(2) < p(3) < \cdots < p(q-1) < p(q) < \cdots,
\]

\[
\lim_{q \to \infty} \min \{ \eta_j^{p(q)}, \eta_j^{p(q)} \} = 0,
\]

\[
\lim_{q \to \infty} \eta_j^{p(q)} = 0,
\]

which means that

\[
\lim_{k \in K} \eta_k = 0, K \subset N,
\]

where \(K\) is a subset of \(N = \{1, 2, 3, \ldots\}\). By \(x_k \in \Omega\), and \(\Omega\) is bounded, we can assume that there exists a constant \(b_3 > 0\) such that \(\|g_k\| \leq b_3\). Then we get

\[
\eta_k = -\frac{g_k^T d_k}{\|d_k\|} \leq \|g_k\| \leq b_3.
\]

We prove our theorem by contradiction. Assume that

\[
\liminf_{k \to \infty} \|g_k\| > 0,
\]

so that there exists a constant \(c_2 > 0\) such that

\[
\|g_k\| \geq c_2, k = 0, 1, 2, \ldots.
\]

By Lemma 4.7, we know that there exists a constant \(\epsilon_0\) satisfying

\[
\prod_{j=0}^{k} \eta_j^{k} \geq \epsilon_0^{k+1}.
\]

Combining (4.29) and (4.28), we deduce that for any integer \(k \geq 1\),

\[
\epsilon_0^{(k+1)n(k)+k} \leq \prod_{j=1}^{(k+1)n(k)+k} \eta_j
\]

\[
= \frac{1}{\eta_0} \prod_{q=0}^{k} \prod_{j=(n(k)+1)q} \eta_j
\]

\[
= \frac{1}{\eta_0} \prod_{q=0}^{k} \prod_{0 \leq j \leq n(k)} \eta_j^{(n(k)+1)+n(k)-j}
\]
\[
\leq \frac{1}{\eta_0} \prod_{q=0}^{k} [\eta_p(q) b_3^{n(k)}] = \frac{1}{\eta_0} b^{kn(k)} \prod_{q=0}^{k} \eta_p(q).
\]

Then we have
\[
\prod_{q=0}^{k} \eta_p(q) \geq \eta_0 \epsilon_0^{n(k)} \left[ \frac{\epsilon_0}{b^{n(k)}} \right]^k \geq \left[ \frac{\epsilon_0^{n(k)+1}}{b^{n(k)}} \min\{1, \eta_0 \epsilon_0^{n(k)}\} \right]^k.
\]

Using Lemma 4.6 we have
\[
\limsup_{q \to \infty} \eta_p(q) > 0,
\]
which contradicts (4.27). Therefore, we obtain
\[
\liminf_{k \to \infty} \|g_k\| = 0.
\]
The proof is complete. \(\Box\)

Similar to Algorithm 2, it is not difficult to get the global convergence of Algorithm 22. Here, we only state it as follows but omit the proof.

**Theorem 4.2.** Let Assumption A hold and the sequence \(\{x_k\}\) be generated by Algorithm 22. Then we have (4.25).

### 5. Numerical results

In this section, we report some numerical results on the problems [30] with initial points. All codes were written in MATLAB 7.0 and run on PC with 2.60GHz CPU processor and 256MB memory and Windows XP operation system. The parameters are chosen as: \(\sigma_1 = 0.1, \sigma_2 = 0.9, \epsilon = 10^{-5}, \epsilon_1 = 0.1, \epsilon_2 = 0.01, p = 5, H = 8, m_1 = 5,\) and the initial matrix \(B_0 = I\) is the unit matrix.

The following Himmeblau stop rule is used [47]:

If \(|f(x_k)| > \epsilon_1\), let \(\text{stop}1 = \frac{|f(x_k) - f(x_{k+1})|}{|f(x_k)|}\); Otherwise, let \(\text{stop}1 = |f(x_k) - f(x_{k+1})|\).

For each problem, if \(\|g(x)\| < \epsilon\) or \(\text{stop}1 < \epsilon_2\) was satisfied, the program will be stopped, where \(\epsilon_1 = \epsilon_2 = 10^{-5}\).

Since the line search cannot always ensure the descent condition \(d_k^T g_k < 0\), uphill search direction may occur in the numerical experiments. In this case, the line search rule maybe fails. In order to avoid this case, the stepsize \(\alpha_k\) will be accepted if the searching number is more than twenty five in line search. We also stop the program if the iteration number is more than one thousand, and the corresponding method is considered to be failed.

In Figure 1-3, “BFGS-WP-Ak1” and “BFGS-WP-Ak2” stand for the modified BFGS formula (2.8) with WWP rule and the modified BFGS formula.
(2.16) with WWP rule, respectively. “L-BFGS-A1” and “L-BFGS-A11” stand for Algorithm 1 and Algorithm 2, respectively. The detailed numerical results are listed on the web site


Dolan and Moré [13] gave a new tool to analyze the efficiency of algorithms. They introduced the notion of a performance profile as a means to evaluate and compare the performance of the set of solvers $S$ on a test set $P$. Assuming that there exist $n_s$ solvers and $n_p$ problems, for each problem $p$ and solver $s$, they defined $t_{p,s}$ = computing time (the number of function evaluations or others) required to solve problem $p$ by solver $s$.

Requiring a baseline for comparisons, they compared the performance on problem $p$ by solver $s$ with the best performance by any solver on this problem; that is, using the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$ 

Suppose that a parameter $r_M \geq r_{p,s}$ for all $p, s$ is chosen, and $r_{p,s} = r_M$ if and only if solver $s$ does not solve problem $p$.

The performance of solver $s$ on any given problem might be of interest, but we would like to obtain an overall assessment of the performance of the solver, then they defined

$$\rho_s(t) = \frac{1}{n_p} \text{size}\{p \in P : r_{p,s} \leq t\},$$

thus $\rho_s(t)$ was the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ was within a factor $t \in R$ of the best possible ratio. Then function $\rho_s$ was the (cumulative) distribution function for the performance ratio. The performance profile $\rho_s : R \mapsto [0, 1]$ for a solver was a nondecreasing, piecewise constant function, continuous from the right at each breakpoint. The value of $\rho_s(1)$ was the probability that the solver would win over the rest of the solvers.

According to the above rules, we know that one solver whose performance profile plot is on top right will win over the rest of the solvers.

Figures 1, 2, and 3 show that the performance of these methods is relative to $NI$, $NF$, and $Time$, respectively, where $NI$ denotes the total number of iterations, $NF$ denotes the total number of the function evaluations and the gradient evaluations where $NT = NF + 5NG$ (see [6, 18]), and $Time$ denotes the cpu time that these methods spent. From these three figures it is clear that the L-BFGS-A11 method has the most wins (has the highest probability of being the optimal solver).

Figure 1 shows that L-BFGS-A11 and L-BFGS-A1 outperform BFGS-WP-Ak1 and BFGS-WP-Ak2 about 5% and 8% test problems, respectively. The L-BFGS-A11 method is predominant among the other three methods for $t \leq 5$. 


Moreover, the L-BFGS-A11 and L-BFGS-A1 and solve 100%, and the BFGS-WP-Ak1 and BFGS-WP-Ak2 method solve about 95% and 92% of the test problems successfully, respectively.

Figure 2 shows that L-BFGS-A11 and L-BFGS-A1 are superior to BFGS-WP-Ak1 and BFGS-WP-Ak2 about 15% test problems. The L-BFGS-A11 and L-BFGS-A1 method can solve 100% of the test problems successfully at $t \approx 4.2$ and $t \approx 7.2$, respectively. The BFGS-WP-Ak1 and BFGS-WP-Ak2 method solve about 85% of the test problems successfully.

Figure 3 shows that L-BFGS-A11 outperforms the other three methods. The L-BFGS-A1 method and the BFGS-WP-Ak1 method solve about 95% and 91% of the test problems, respectively, and the BFGS-WP-Ak2 solves about 88% of the test problems successfully.

In summary, the presented numerical results reveal that Algorithm 1 and Algorithm 11, compared with other two methods with WWP line search and BFGS update, have potential advantages for these problems.

6. Conclusion

This paper gives two modified L-BFGS method with one nonmonotone line search technique for solving unconstrained optimization, which include the function value at the current and next iterative point values. The global convergence for the uniformly convex functions are established. The numerical
Performance profiles of these methods (NFG). Figure 2

Performance profiles of these methods (Time). Figure 3
results show that the given methods are competitive to the other standard BFGS methods for the test problems.

For further research, we should study the performance of the new algorithm at different stop rules and different testing environment (such as [15]). Moreover, more numerical experiments for large practical problems should be done in the future.

References


GONGLIN YUAN
College of Mathematics and Information Science
Guangxi University
Nanning, Guangxi, 530004, P. R. China
E-mail address: glyuan@gxu.edu.cn

ZENGXIN WEI
College of Mathematics and Information Science
Guangxi University
Nanning, Guangxi, 530004, P. R. China
E-mail address: zxwei@gxu.edu.cn

YANLIN WU
College of Mathematics and Information Science
Guangxi University
Nanning, Guangxi, 530004, P. R. China
E-mail address: wyl-kk@163.com