GLOBAL CONVERGENCE OF A MODIFIED LIU-STOREY CONJUGATE GRADIENT METHOD

Min LI[1], Yu CHEN[2], Ai-Ping QU[3]

In this paper[4], we make a modification to the LS conjugate gradient method and propose a descent LS method. The method can generates sufficient descent direction for the objective function. We prove that the method is globally convergent with an Armijo-type line search. Moreover, under mild conditions, we show that the method is globally convergent if the Armijo line search or the Wolfe line search is used. The numerical results show that the proposed methods are efficient

Keywords: LS conjugate gradient method; Sufficient descent property; Global convergence.

1. Introduction

In this paper, we consider the unconstrained problem

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

where $f: R^n \to R$ is continuously differentiable. Nonlinear conjugate gradient methods are efficient for problem (1). The nonlinear conjugate gradient methods generate iterates by letting

$$x_{k+1} = x_k + \alpha_k d_k,$$

with

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_k d_{k-1}, & k \geq 1, \end{cases}$$

where α_k is the step-length, $g_k = g(x_k)$ denotes the gradient of f at x_k, and β_k is a suitable scalar. Well-known conjugate methods include the HS, FR, PRP, CD, LS and DY methods [1-7]. In the survey paper [10], Hager and Zhang reviewed the development of different versions of nonlinear gradient methods, with special

[1] Prof., Department of Mathematics and Applied Mathematics, Huaihua University, Huaihua, 418008, China
[2] Prof., Department of Mathematics and Applied Mathematics, Huaihua University, Huaihua, 418008, China
[3] Prof., Department of Mathematics and Applied Mathematics, Huaihua University, Huaihua, 418008, China
[4] This work is supposed by NSF grant .HHUQ2009-01and HHUY2010-04 of HuaiHua University, China.
Corresponding author: Min Li, Email:liminmath@hotmail.com.

attention given to global convergence properties. We refer to [10] for more details.

Recently, there is a growing interest in the development of descent conjugate gradient methods. The first one was due to the CG_DESCENT method proposed by Hager and Zhang [11]. They calculated β_k by

$$\beta_k^{\mathrm{N}} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}} - 2\frac{\| y_{k-1} \|^2 \, g_k^T d_{k-1}}{(d_{k-1}^T y_{k-1})^2} \ , \tag{2}$$

here and throughout this paper, $\|\bullet\|$ stands for the Euclidean norm of a vector and $y_{k-1} = g_k - g_{k-1}$. An attractive property of the CG_DESCENT method is that the directions d_k generated by the CG_DESCENT method satisfy the sufficient descent condition $g_k^T d_k \le -\frac{7}{8} \| g_k \|^2$. The method is globally convergent if the Wolfe line search is used [11].

Zhang and Zhou [12] made a modification to the CG_DESCENT method and propose a so-called cautious CG_DESCENT method. It was proved that the cautious CG_DESCENT method with the standard Armijo line search is also globally convergent.

However, just as Hager and Zhang [10] pointed out that the research about the LS method [6] is very few. The purpose of this paper is to develop a descent LS method and establish its global convergence.

In the next section, we propose the method. In section 3, we prove the global convergence of the proposed method with an Armijo-type line search. In section 4, we establish the global convergence of the proposed method with Armijo line search and Wolfe line search. In section 5, we do some numerical experiments to test the proposed methods and compare their performance with some existing methods.

2. The algorithms

The standard LS method [6] specifies the β_k^{LS} by

$$\beta_k^{\mathrm{LS}} = -\frac{g_k^T y_{k-1}}{g_{k-1}^T d_{k-1}} \ ,$$

Inspired by the CG_DESCENT method, we give the following modified formula to β_k^{MLS}

$$\beta_k^{\mathrm{MLS}} = -\frac{g_k^T y_{k-1}}{g_{k-1}^T d_{k-1}} - t\frac{\| y_{k-1} \|^2 \, g_k^T d_{k-1}}{(g_{k-1}^T d_{k-1})^2} \ , \tag{3}$$

where $t > 1/4$ is a constant. Obviously, if exact line search is used, then β_k^{MLS} will reduce to β_k^{LS}. The theorem below shows an attractive property of the modified LS method that the search directions d_k will always be sufficiently descent if $g_{k-1}^T d_{k-1} \neq 0$.

Theorem 2.1. Let $\{d_k\}$ be generated by

$$d_k = -g_k + \beta_k^{\mathrm{MLS}} d_{k-1}, \quad d_0 = -g_0 \ , \tag{4}$$

If $g_{k-1}^T d_{k-1} \neq 0$, then the following inequality holds

$$g_k^T d_k \leq \left(\frac{1}{4t} - 1\right) \| g_k \|^2 \tag{5}$$

In other words, the directions d_k are sufficiently descent directions for function f if $t > 1/4$.

Proof. It is clear that (5) holds for $k = 0$. Suppose that (5) holds for some $k > 0$. We are going to show that it holds for $k+1$. Multiplying both sides of the first equation in (4) by g_k, we get

$$\begin{aligned}
g_k^T d_k &= -\| g_k \|^2 + \beta_k^{\mathrm{MLS}} g_k^T d_{k-1} \\
&= -\| g_k \|^2 - \left(\frac{g_k^T y_{k-1}}{g_{k-1}^T d_{k-1}} + t \frac{\| y_{k-1} \|^2 \, g_k^T d_{k-1}}{(g_{k-1}^T d_{k-1})^2} \right) g_k^T d_{k-1} \\
&= \frac{-\| g_k \|^2 (g_{k-1}^T d_{k-1})^2 - (g_k^T y_{k-1})(g_{k-1}^T d_{k-1})(g_k^T d_{k-1}) - t \| y_{k-1} \|^2 (g_k^T d_{k-1})^2}{\left(g_{k-1}^T d_{k-1} \right)^2} \\
&\leq \frac{-\| g_k \|^2 (g_{k-1}^T d_{k-1})^2 + \frac{1}{2} \frac{1}{2t} \| g_k \|^2 (g_{k-1}^T d_{k-1})^2}{\left(g_{k-1}^T d_{k-1} \right)^2} \\
&\quad + \frac{\frac{1}{2} 2t \| y_{k-1} \|^2 (g_k^T d_{k-1})^2 - t \| y_{k-1} \|^2 (g_k^T d_{k-1})^2}{\left(g_{k-1}^T d_{k-1} \right)^2} \\
&= \left(\frac{1}{4t} - 1 \right) \| g_k \|^2 .
\end{aligned}$$

The proof is complete.

The above theorem shows that the directions generated by (4) are sufficient descent directions. This feature is independent of the line search used. Based on the above process, we present concrete MLS method with an Armijo-type line search as follows

Algorithm 2.1. (MLS Method with an Armijo-type line search.)

Step 0. Given constant $\varepsilon > 0$. Given $x_0 \in R^N$. Set $k = 0$.

Step 1. Stop if $\| g_k \|_\infty < \varepsilon$.

Step 2. Compute d_k by (4).

Step 3. Determine the steplength α_k by the following Armijo-type search. Namely determines $\alpha_k = \max\{\rho^j,\ j = 0,1,2,\cdots\}$ satisfying

$$f(x_k + \alpha_k d_k) - f(x_k) \le -\delta_1 \alpha_k^2 \| d_k \|^4 , \tag{6}$$

where $\delta_1 > 0$ and $0 < \rho < 1$.

Step 4. Let $x_{k+1} = x_k + \alpha_k d_k$. If $\| g_k \|_\infty < \varepsilon$, then stop.

Step 5. Set $k = k + 1$, go to Step 2.

As we have shown in Theorem 2.1 that d_k is a descent of f at x_k, it is not difficult to see that the above algorithm is well defined. Moreover, if f is bounded from below, we have from (6) that

$$\sum_{k=0}^{\infty} \alpha_k^2 \| d_k \|^4 .$$

This implies

$$\lim_{k\to\infty} \alpha_k^2 \| d_k \|^4 = 0 \qquad \text{and} \qquad \lim_{k\to\infty} \alpha_k \| d_k \|^2 = 0 . \tag{7}$$

3. The global convergence of Algorithm 2.1

In this section, we will focus on the global convergence of the Algorithm 2.1. We first make the following assumptions.

Assumption 3.1

I. The level set $\Omega = \{x \mid f(x) \le f(x_0),\ x \in R^n\}$ is bounded.

II. In some neighborhood N of Ω, function f is continuously differentiable and its gradient $g(x)$ is Lipschitz continuous, namely, there exists a constant L such that

$$\| g(x) - g(y) \| \le L \| x - y \|, \qquad \forall x, y \in N . \tag{8}$$

From now on, throughout this paper, we always suppose that the conditions in this assumption hold. It follows directly from the Assumption 3.1 that there exist two positive constants B and γ_1 such that

$$\| x \| \le B \qquad \text{and} \qquad \| g(x) \| \le \gamma_1, \qquad \forall x \in \Omega. \tag{9}$$

In the later part of this section, we will prove the global convergence of Algorithm 2.1. At first, we give the following lemma about the boundness of the directions d_k.

Lemma 3.1. Let the conditions in Assumption 3.1 hold, and $\{d_k\}$ generated by Algorithm 2.1. If there exists a constant $\gamma > 0$ such that

$$\| g_k \| > \gamma, \qquad \forall k \geq 0, \tag{10}$$

then there exists a constant $M > 0$ such that

$$\| d_k \| \leq M, \qquad \forall k \geq 0. \tag{11}$$

Proof. We get from (4) , (9) and (10) that

$$\begin{aligned}
\| d_k \| &\leq \| g_k \| + | \beta_k^{\mathrm{MLS}} | \| d_{k-1} \| \\
&\leq \| g_k \| + \frac{\| g_k \| \| y_k \| \| d_{k-1} \|}{| g_{k-1}^T d_{k-1} |} + t\frac{\| y_{k-1} \|^2 \| g_k \| \| d_{k-1} \|}{| g_{k-1}^T d_{k-1} |^2} \| d_{k-1} \| \\
&\leq \gamma_1 + \frac{4t\gamma_1 L\alpha_{k-1} \| d_{k-1} \|^2}{(4t-1)\gamma^2} + \frac{t(4t)^2 2\gamma_1^2 L\alpha_{k-1} \| d_{k-1} \|^2}{(4t-1)^2\gamma^4} \| d_{k-1} \| \\
&\leq \gamma_1 + \frac{4t\gamma_1 L}{(4t-1)\gamma^2}\alpha_{k-1} \| d_{k-1} \|^2 + \frac{32t^3\gamma_1^2 L}{(4t-1)^2\gamma^4}\alpha_{k-1} \| d_{k-1} \|^2 \| d_{k-1} \|.
\end{aligned}$$

(7) implies that for any constant $b \in (0,1)$, there exists a index k_0 such that

$$\frac{32t^3\gamma_1^2 L}{(4t-1)^2\gamma^4}\alpha_{k-1} \| d_{k-1} \|^2 < b, \qquad \forall k > k_0.$$

Then

$$\| d_k \| \leq \gamma_1 + \frac{(4t-1)\gamma^2 b}{8t^2\gamma_1} + b \| d_{k-1} \| = C + b \| d_{k-1} \|,$$

where $C = \gamma_1 + (4t-1)\gamma^2 b / (8t^2\gamma_1)$ is a constant. For any $k \geq k_0$, we have

$$\| d_k \| \leq C(1 + b + b^2 + \cdots + b^{k-k_0+1}) + b^{k-k_0} \| d_{k_0} \| \leq \frac{C}{1-b} + \| d_{k_0} \|.$$

So, we can let

$$M = \max\{\| d_1 \|, \| d_2 \|, \cdots, \| d_{k_0} \|, \tfrac{C}{1-b} + \| d_{k_0} \|\}$$

to get (11). The proof is completed.

Based on Lemma 3.1, we give the next global convergence theorem for Algorithm 2.1 with the Armijo-type line search.

Theorem 3.2. Let the conditions in Assumption 3.1 hold, $\{x_k\}$ and $\{d_k\}$ be generated by Algorithm 2.1, then either $\| g_k \| = 0$ for some k or

$$\liminf_{k \to \infty} \| g_k \| = 0. \tag{12}$$

Proof. We suppose for contradiction that neither $\| g_k \| = 0$ nor $\liminf_{k \to \infty} \| g_k \| = 0$, then there exists a constant $\gamma > 0$ such that (10) holds. We

consider the case that $\liminf_{k\to\infty}\alpha_k > 0$. (7) implies that $\liminf_{k\to\infty}\| d_k \| = 0$. This together with (5) implies $\liminf_{k\to\infty}\| g_k \| = 0$, which contradicts (10). Suppose that $\liminf_{k\to\infty}\alpha_k = 0$, then there exists a infinite index set K such that

$$\liminf_{k\in K,\ k\to\infty}\alpha_k = 0. \tag{13}$$

From the Step 3 of the Algorithm 2.1, $\rho^{-1}\alpha_k$ does not satisfy (6), which implies

$$f(x_k + \rho^{-1}\alpha_k d_k) - f(x_k) > -\delta_1\rho^{-1}\alpha_k^2 \| d_k \|^4 \tag{14}$$

By the Lipschitz condition (8) and the mean value theorem, there is a $\xi_k \in [0,1]$, such that

$$\begin{aligned} f(x_k + \rho^{-1}\alpha_k d_k) - f(x_k) &= \rho^{-1}\alpha_k g(x_k + \xi_k\rho^{-1}\alpha_k d_k)^T d_k \\ &= \rho^{-1}\alpha_k g_k^T d_k + \rho^{-1}\alpha_k \left(g(x_k + \xi_k\rho^{-1}\alpha_k d_k) - g_k \right)^T d_k \\ &\le \rho^{-1}\alpha_k g_k^T d_k + L\rho^{-2}\alpha_k^2 \| d_k \|^2 . \end{aligned}$$

This together with (5), (11) and (14) gives

$$\begin{aligned} \frac{4t-1}{4t}\| g_k \|^2 \le -g_k^T d_k &\le \alpha_k\rho^{-1}(\delta_1 \| d_k \|^4 + L \| d_k \|^2) \\ &\le \alpha_k\rho^{-1}(\delta_1 M^4 + LM^2). \end{aligned}$$

This together with $\lim_{k\in K,\ k\to\infty}\alpha_k = 0$ implies $\lim_{k\in K,\ k\to\infty}\| g_k \| = 0$, which yields a contradiction and completes the proof.

4. The global convergence of MLS method with Armijo line search and Wolfe line search

In this section, we will prove the global convergence of the MLS method with the Armijo line search and the Wolfe line search. The Armijo line search condition is

$$f(x_k + \alpha_k d_k) - f(x_k) \le \delta\alpha_k g_k^T d_k \tag{15}$$

where $\delta \in (0, 1)$. We determine the steplength α_k by letting it be the largest scale in the set $\{\rho^j,\ j = 0,1,2,\cdots\}$, where $\rho \in (0, 1)$.

The Wolfe line search conditions are the following two inequalities

$$\begin{cases} f(x_k + \alpha_k d_k) - f(x_k) \le \delta\alpha_k g_k^T d_k, \\ d_k^T g(x_k + \alpha_k d_k) \ge \sigma g_k^T d_k, \end{cases} \tag{16}$$

where $0 < \delta \le \sigma < 1$. However, it seems not easy to establish the global convergence of the relative method. So, we introduce a cautious update rule to (4) and let d_k be determined by the following cautious rule

$$d_k = \begin{cases} -g_k, & \text{if} \quad |\, g_{k-1}^T d_{k-1} \,| < \varepsilon_1 \,\| d_{k-1} \|\,, \\ -g_k + \beta_k^{\mathrm{MLS}} d_{k-1}, & \text{else}\,, \end{cases} \tag{17}$$

where $\varepsilon_1 > 0$ is a constant. Such a cautious update rule was proposed by D. Li and M. Fukushima in [13] and was used to CG_DESCENT method by Zhang and Zhou in [12]. Now we present the cautious MLS method with the Armijo line or the Wolfe line search as follows:

Algorithm 4.1 (The cautious MLS method.)

Step 0. Given constants $\varepsilon > 0, \varepsilon_1 > 0$. Given $x_0 \in R^N$. Set $k = 0$.

Step 1. Stop if $\| g_k \|_\infty < \varepsilon$.

Step 2. Compute d_k by (17).

Step 3. Determine the steplength α_k by Armijo line search or Wolfe line search.

Step 4. Let $x_{k+1} = x_k + \alpha_k d_k$. If $\| g_k \|_\infty < \varepsilon$, then stop.

Step 5. Set $k = k + 1$, go to Step 2.

From Theorem 2.1 we have that d_k is a descent direction of f at x_k. It is easy to see that the above algorithm is well defined. We simply call the algorithm CMLS method in the later part of this paper.

To prove the global convergence of CMLS method, we first show the following useful lemma, which was essentially proved by Zoutendijk [14] and Wolfe [15,16]

Lemma 4.1. Let the conditions in Assumption 3.1 hold. and $\{x_k\}$ be generated by the CMLS method with Armijo line search or Wolfe line search. Then we have

$$\sum_{k=0}^{\infty} \frac{\| g_k \|^4}{\| d_k \|^2} < +\infty \ . \tag{18}$$

Proof. Consider the case where the Armijo line search is used. We first show that there is a constant c such that

$$\alpha_k \geq c \frac{\| g_k \|^2}{\| d_k \|^2}\,. \tag{19}$$

If $\alpha_k \geq 1$, we get from (4) that

$$1 \geq \left(1 - \frac{1}{4t}\right) \frac{\| g_k \|}{\| d_k \|}\,.$$

This implies (19) with $c = (1 - \frac{1}{4t})^2$.

If $\alpha_k < 1$, by the line search rule, $\rho^{-1}\alpha_k$ will not satisfy (15). This means

$$f(x_k + \rho^{-1}\alpha_k d_k) - f(x_k) > \delta\rho^{-1}\alpha_k g_k^T d_k . \tag{20}$$

By the Lipschitz condition (8) and the mean value theorem, there is a $\xi_k \in [0,1]$, such that

$$\begin{aligned} & f(x_k + \rho^{-1}\alpha_k d_k) - f(x_k) \\ &= \rho^{-1}\alpha_k g(x_k + \xi_k \rho^{-1}\alpha_k d_k)^T d_k \\ &= \rho^{-1}\alpha_k g_k^T d_k + \rho^{-1}\alpha_k \left(g(x_k + \xi_k \rho^{-1}\alpha_k d_k) - g_k \right)^T d_k \\ &\le \rho^{-1}\alpha_k g_k^T d_k + L\rho^{-2}\alpha_k^2 \| d_k \|^2 . \end{aligned}$$

The last inequality together with inequality (20) implies (19) with

$$c = \min\left\{ \left(1 - \frac{1}{4t}\right)^2, \left(1 - \frac{1}{4t}\right)\frac{(1-\delta)\rho}{L} \right\}$$

On the other hand, we get from the Armijo condition (15) and the boundness of $\{x_k\} \subset \Omega$ that

$$-\sum_{k=0}^{\infty} \alpha_k g_k^T d_k < +\infty . \tag{21}$$

This together with (5) and (19) implies

$$\sum_{k=0}^{\infty} \frac{\| g_k \|^4}{\| d_k \|^2} < +\infty .$$

Consider the case where the Wolfe line search is used. From the second inequality of (16), we have

$$g_{k+1}^T d_k \ge \sigma g_k^T d_k .$$

This together with the Lipschtiz condition implies

$$(\sigma - 1) g_k^T d_k \le (g_{k+1} - g_k)^T d_k \le L\alpha_k \| d_k \|^2 .$$

Consequently, we get

$$\alpha_k \ge \frac{(\sigma - 1)}{L} \frac{g_k^T d_k}{\| d_k \|^2} = \frac{(1-\sigma)}{L} \frac{| g_k^T d_k |}{\| d_k \|^2} .$$

Comparing this with the sufficient descent condition (5) we have

$$\alpha_k \ge c \frac{\| g_k \|^2}{\| d_k \|^2}, \; c = \frac{(1-\sigma)(4t-1)}{4Lt}, \; t > \frac{1}{4} . \tag{22}$$

In a way similar to the proof for the case of Armijo line search, we can get (18). The proof is complete.

The following theorem establishes the convergence of the CMLS method with Armijo line search.

Theorem 4.2. Let $\{x_k\}$ be generated by the CMLS method with Armijo line search. If the conditions in Assumption 3.1 hold, then we have either $\| g_k \| = 0$ for some k, or $\liminf_{k\to\infty} \| g_k \| = 0$.

Proof. We suppose for the sake of contradiction that $\| g_k \| \neq 0, \forall k \geq 0$, and $\liminf_{k\to\infty} \| g_k \| > 0$. Denote $\gamma = \inf\{\| g_k \| : k \geq 0\}$. It is clear that $\gamma > 0$ and

$$\| g_k \| \geq \gamma, \quad \forall k \geq 0. \tag{23}$$

Define the index set $K = \{i \mid d_i = -g_i\}$. It is not difficult to see from the Zoutendijk condition (18) that the index set K must be finite. By (3), (9), (17) and (23), we derive

$$\begin{aligned}
\left|\beta_k^{\mathrm{MLS}}\right| &= \left| -\frac{g_k^T y_{k-1}}{g_{k-1}^T d_{k-1}} - t\frac{\| y_{k-1} \|^2 g_k^T d_{k-1}}{(g_{k-1}^T d_{k-1})^2} \right| \\
&\leq \left| \frac{g_k^T y_{k-1}}{g_{k-1}^T d_{k-1}} \right| + \left| \frac{\| y_{k-1} \|^2 g_k^T d_{k-1}}{(g_{k-1}^T d_{k-1})^2} \right| \\
&\leq \frac{\| g_k \| (\| g_k \| + \| g_{k1-1} \|)}{\varepsilon_1 \| d_{k-1} \|} + t\frac{(\| g_k \| + \| g_{k-1} \|)^2 \| g_k \| \| d_{k-1} \|}{(\varepsilon_1 \| d_{k-1} \|)^2} \\
&\leq \frac{\| g_k \| 2\gamma_1}{\varepsilon_1 \| d_{k-1} \|} + t\frac{4\gamma_1^2 \| g_k \|}{\varepsilon_1^2 \| d_{k-1} \|} \\
&\leq \left(\frac{2\gamma_1}{\varepsilon_1} + \frac{4t\gamma_1^2}{\varepsilon_1^2} \right) \frac{\| g_k \|}{\| d_{k-1} \|}.
\end{aligned}$$

This together with (17) implies

$$\| d_k \| \leq \| g_k \| + |\beta_k^{\mathrm{MLS}}| \| d_{k-1} \| \leq \left(1 + \frac{2\gamma_1}{\varepsilon_1} + \frac{4t\gamma_1^2}{\varepsilon_1^2} \right) \| g_k \|.$$

Therefore, we get from the Zoutendijik condition (18)

$$\sum_{k=0}^{\infty} \| g_k \|^2 < +\infty.$$

which yields a contradiction. The proof is completed.

In a way similar to Theorem 4.2, it is not difficult to establish the following convergence for the CMLS method with the Wolfe line search.

Theorem 4.3. Let $\{x_k\}$ be generated by the CMLS method with Wolfe line search. If the conditions in Assumption 3.1 hold, then we have either $\| g_k \| = 0$ for some k, or $\liminf_{k\to\infty} \| g_k \| = 0$.

4. Numerical results

In this section, we do some numerical experiments to test the CMLS method. We compare the performance with some existing conjugate gradient methods including the PRP+ method developed by Nocedal [9] and the CG_DESCENT method [11] . The PRP+ code was obtained from Nocedal's web page at *http:// www.ece.northwestern.edu.nocedalsoftware.html*
and the CG_DESCENT code from Hager's web page at
http://www. math.ufl.edu/hager/papers/CG.

Table 1

Melting points and elemental analyses

N	Prob	Dim	N	Prob	Dim	N	Prob	Dim
1	PENALTY1	1000	37	GENHUMPS	1000	73	BROYDN7D	5000
2	NONDQUAR	10000	38	MSQRTBLS	1024	74	DIXMAANF	9000
3	SENSORS	100	39	WOODS	10000	75	DIXMAANG	3000
4	VARDIM	200	40	QUARTC	5000	76	CHAINWOO	10000
5	FMINSRF2	5625	41	CURLY20	1000	77	FLETCBV2	1000
6	TQUARTIC	5000	42	WOODS	4000	78	DIXMAAND	9000
7	BRYBND	10000	43	DIXMAANI	9000	79	POWER	5000
8	VAREIGVL	1000	44	SPMSRTLS	4999	80	GENROSE	500
9	COSINE	10000	45	DIXMAANG	9000	81	POWELLSG	1000
10	FREUROTH	5000	46	SCHMVETT	10000	82	DQDRTIC	5000
11	DIXMAANJ	9000	47	SROSENBR	10000	83	FMINSRF2	1024
12	GENROSE	100	48	PENALTY2	200	84	CRAGGLVY	5000
13	TOINTPSP	50	49	ERRINROS	50	85	BRYBND	5000
14	VAREIGVL	50	50	NONDQUAR	5000	86	EG2	1000
15	CURLY30	1000	51	DIXMAANA	9000	87	EDENSCH	2000
16	GENHUMPS	5000	52	CHNROSNB	50	88	LIARWHD	10000
17	DQDRTIC	1000	53	DIXMAANK	1500	89	DIXMAANE	9000
18	MANCINO	100	54	DIXON3DQ	1000	90	SPMSRTLS	10000
19	TQUARTIC	1000	55	SCHMVETT	5000	91	ARGLINC	100
20	DIXMAANC	9000	56	LIARWHD	5000	92	MANCINO	50
21	ENGVAL1	1000	57	FLETCBV2	5000	93	DIXMAANB	3000
22	DQRTIC	5000	58	MOREBV	5000	94	POWELLSG	5000
23	MSQRTALS	1024	59	CURLY20	100	95	DIXMAANL	9000
24	SINQUAD	1000	60	DIXMAANB	9000	96	SINQUAD	5000
25	CURLY10	100	61	TOINTGOR	50	97	ENGVAL1	5000
26	FLETCHCR	1000	62	QUARTC	10000	98	COSINE	1000
27	MOREBV	1000	63	FREUROTH	1000	99	TOINTQOR	50
28	POWER	10000	64	FLETCHCR	100	100	ARGLINA	100
29	BDQRTIC	1000	65	DECONVU	61	101	FMINSURF	5625
30	ARGLINB	100	66	SROSENBR	5000	102	DIXMAANE	3000
31	ARWHEAD	5000	67	CURLY10	1000	103	DIXMAAND	3000
32	TESTQUAD	5000	68	CURLY30	100	104	DIXMAANA	3000
33	NONDIA	10000	69	PENALTY1	500	105	FMINSURF	1024
34	NONDIA	5000	70	DIXMAANJ	3000	106	TOINTGSS	10000
35	TRIDIA	10000	71	DIXMAANI	3000	107	SPARSQUR	5000
36	DIXON3DQ	10000	72	DIXMAANH	9000	108	DIXMAANH	3000

All the test problems are the unconstrained problems in the CUTEr [19] library with dimensions varying from 50 to 10000. We stop the iteration if $\| g_k \|_\infty \leq 10^{-6}$ is satisfied. All codes were written in Fortran and run on a PC with 2.8 GHZ CPU processor and 2GB RAM memory and Linux operation system. Table 1 lists all the problems (Prob) and their dimensions (Dim). All the result are listed in Table 2, which include the total number of iterations (Iter), the total number of function evaluations (Nf), the total number of gradient evaluations (Ng), the CPU time (Time) in seconds, respectively. In Table 2, "$-$" means the method failed.

Table 2

Melting points and elemental analyses

N	CMLS method	CG_DESCENT method	PRP+ method
	Iter/Nf/Ng/Time	Iter/Nf/Ng/Time	Iter/Nf(Ng)/Time
1	55/133/86/0.021	50/121/77/0.018	42/173/0.025
2	10007/20027/10089/27.5	10007/20025/10429/23.9	-/-/-
3	21/61/45/0.522	25/57/44/0.506	27/66/0.491
4	28/57/29/0.002	28/57/29/0.002	8/44/0.001
5	323/647/324/0.992	363/729/366/1.07	350/707/1.446
6	18/46/33/0.044	21/52/38/0.049	9/32/0.03
7	31/64/34/0.261	29/60/32/0.234	69/154/0.773
8	76/207/131/0.087	93/243/150/0.097	30/65/0.031
9	14/32/26/0.12	12/32/28/0.12	9/28/0.104
10	43/86/77/0.198	65/126/95/0.25	-/-/-
11	356/713/357/1.485	295/591/296/1.152	293/593/1.381
12	293/613/333/0.014	305/641/347/0.013	288/603/0.015
13	128/262/180/0.003	155/327/211/0.004	-/-/-
14	64/175/111/0.004	60/164/104/0.005	25/57/0.001
15	10740/17083/16943/11.02	9765/15713/15122/9.3	-/-/-
16	6832/13923/7135/41.858	9412/18948/9575/54.02	7442/15241/46.93
17	6/13/7/0.006	7/15/8/0.004	5/15/0.004
18	12/25/13/0.148	11/23/12/0.136	11/27/0.172
19	13/47/40/0.01	24/64/46/0.012	11/37/0.007
20	11/23/12/0.052	10/21/11/0.043	6/25/0.058
21	26/48/32/0.013	26/49/33/0.013	-/-/-
22	50/101/51/0.057	33/67/34/0.033	17/66/0.034
23	3629/7265/3638/18.25	3393/6793/3402/16.73	2934/5873/20.98
24	89/188/144/0.082	84/184/151/0.08	-/-/-
25	918/1677/1259/0.043	1013/1797/1508/0.046	-/-/-
26	4741/9599/4881/2.069	6828/14236/7479/2.90	4371/8767/2.173
27	425/851/426/0.157	425/851/426/0.143	425/851/0.18
28	371/743/372/0.685	369/739/370/0.565	355/719/0.669
29	479/991/699/0.325	628/1296/1025/0.434	-/-/-
30	4/9/8/0.004	6/12/13/0.005	-/-/-
31	10/23/16/0.036	3763/6992/8726/18.26	-/-/-
32	1718/3437/1719/1.583	1715/3431/1716/1.301	1590/3183/1.409
33	9/20/12/0.05	9/22/16/0.057	6/26/0.065

Table 2 continuous

N	CMLS method	CG_DESCENT method	PRP+ method
	Iter/Nf/Ng/Time	Iter/Nf/Ng/Time	Iter/Nf(Ng)/Time
34	12/29/21/0.037	8/27/22/0.035	5/26/0.03
35	1115/2231/1116/2.714	1115/2231/1116/2.37	1112/2227/2.78
36	10000/20001/10002/24.0	10000/20001/10002/21.0	10000/20006/24.673
37	2471/5094/2658/3.047	2697/5568/2908/3.233	2116/5025/2.909
38	2208/4423/2217/11.102	2318/4642/2325/11.391	2396/4797/17.161
39	227/491/294/0.983	187/426/257/0.787	232/487/1.04
40	50/101/51/0.056	33/67/34/0.032	17/66/0.034
41	9775/15435/15148/7.415	9757/15481/15084/7.111	-/-/-
42	226/491/286/0.359	148/342/214/0.236	190/393/0.316
43	3627/7255/3628/15.258	2687/5375/2688/10.343	3542/7091/16.534
44	202/411/211/0.676	218/443/227/0.698	212/430/0.872
45	261/523/262/1.098	266/533/267/1.026	404/816/1.905
46	47/80/63/0.94	39/65/54/0.783	44/102/0.982
47	9/20/13/0.033	12/26/17/0.039	8/26/0.04
48	190/225/347/0.122	199/234/365/0.126	-/-/-
49	1505/2978/2161/0.037	1013/2023/1444/0.024	-/-/-
50	5008/10029/5131/6.486	5014/10053/5154/5.636	-/-/-
51	8/17/9/0.043	9/19/10/0.04	7/23/0.054
52	252/506/255/0.006	272/545/273/0.005	314/636/0.008
53	1399/2799/1400/0.879	1434/2869/1435/0.835	1404/2818/1.017
54	1000/2001/1002/0.23	1000/2001/1002/0.199	1000/2005/0.235
55	44/75/59/0.437	39/66/53/0.38	41/91/0.439
56	25/56/41/0.079	21/48/32/0.062	16/46/0.062
57	0/1/1/0.004	0/1/1/0.004	4101/8203/17.226
58	167/335/169/0.328	147/295/149/0.264	161/323/0.369
59	910/1664/1350/0.067	899/1670/1310/0.063	-/-/-
60	10/21/11/0.045	9/19/10/0.04	7/26/0.063
61	119/218/147/0.006	122/224/154/0.006	-/-/-
62	53/107/54/0.125	35/71/36/0.073	16/69/0.073
63	98/201/128/0.068	85/173/113/0.058	-/-/-
64	796/1670/895/0.036	782/1664/895/0.035	798/1610/0.041
65	306/613/308/0.022	457/916/462/0.03	696/1398/0.062
66	9/20/13/0.016	12/26/16/0.018	10/29/0.022
67	8887/13894/13325/4.241	9431/14475/14406/4.18	-/-/-
68	927/1750/1323/0.084	982/1843/1441/0.088	-/-/-
69	49/114/72/0.01	45/107/69/0.008	29/113/0.009
70	1239/2479/1240/1.56	297/595/298/0.349	360/728/0.531
71	3193/6387/3194/3.988	2552/5105/2553/2.996	2399/4804/3.437
72	261/523/262/1.1	263/527/264/1.016	530/1069/2.497
73	1426/2840/1443/13.54	1502/2988/1524/14.05	6007/12369/83.27
74	270/541/271/1.156	269/539/270/1.054	242/491/1.141
75	167/335/168/0.211	170/341/171/0.199	159/327/0.235
76	313/628/403/2.305	358/696/448/2.429	10001/20701/80.686
77	1174/2349/1177/0.882	1052/2105/1055/0.758	942/1886/0.76
78	13/27/14/0.061	12/25/13/0.051	8/26/0.061
79	266/533/267/0.236	258/517/259/0.195	252/514/0.229

Table 2 continuous

N	CMLS method	CG_DESCENT method	PRP+ method
	Iter/Nf/Ng/Time	Iter/Nf/Ng/Time	Iter/Nf(Ng)/Time
80	1193/2416/1237/0.26	1259/2559/1309/0.255	1121/2269/0.277
81	97/195/109/0.025	241/490/274/0.051	99/238/0.03
82	7/15/8/0.018	7/15/8/0.019	5/15/0.021
83	246/493/247/0.12	276/558/282/0.129	257/517/0.163
84	119/231/162/0.711	116/208/150/0.64	-/-/-
85	30/61/31/0.115	36/74/39/0.141	26/66/0.161
86	4/9/6/0.004	4/9/6/0.004	-/-/-
87	30/54/41/0.04	33/61/43/0.041	-/-/-
88	21/51/33/0.141	25/60/41/0.162	15/46/0.131
89	366/733/367/1.519	359/719/360/1.378	361/727/1.71
90	217/441/226/1.509	225/457/234/1.498	217/440/1.829
91	4/9/8/0.003	5/11/11/0.005	-/-/-
92	9/19/10/0.028	9/19/10/0.031	10/24/0.038
93	10/21/11/0.015	9/19/10/0.014	6/23/0.016
94	97/195/109/0.123	162/332/187/0.176	148/346/0.201
95	269/539/270/1.122	240/481/241/0.932	336/680/1.586
96	84/179/132/0.432	46/111/108/0.311	-/-/-
97	25/46/33/0.071	26/46/37/0.073	-/-/-
98	12/29/23/0.01	12/28/24/0.01	9/29/0.01
99	31/59/38/0.001	32/61/41/0	29/60/0.001
100	1/3/2/0.002	1/3/2/0.002	1/4/0.002
101	442/885/443/1.424	492/985/493/1.502	471/949/2.033
102	228/457/229/0.29	225/451/226/0.263	228/462/0.329
103	13/27/14/0.018	12/25/13/0.015	7/25/0.018
104	8/17/9/0.012	9/19/10/0.015	7/20/0.019
105	210/421/211/0.108	236/474/238/0.116	226/455/0.151
106	4/9/5/0.047	4/9/5/0.047	4/20/0.108
107	37/78/44/0.133	21/43/22/0.065	24/76/0.168
108	197/395/198/0.248	167/335/168/0.195	257/523/0.375

We used the profiles by [29] to compare the performance of those methods. Figures 1-4 show the performance of the above methods related to the CPU time (in second), the total number of iterations, the total number of function evaluations, and the total number of gradient evaluations, respectively. The curves in the figures have the following meanings:

- "CG_DESCENT" stands for the performance of the CG_DESCENT method with the approximate Wolfe line search proposed in [11]. we used the Fortran77 (version 1.4) code and the default parameters there.
- "PRP+" means the PRP+ method with Wolfe line search proposed in [17].
- CMLS" stands for the performance of the CMLS method with $t = 2.55,\ \varepsilon_1 = 10^{-15}$ and the same line search as "CG_DESCENT" method.

We see from figures 1-4 that the performance of the CMLS method and the CG_DESCENT method are much better than the performance of the PRP+ method. The curves "CG_DESCENT" and "CMLS" are very close. We also noted

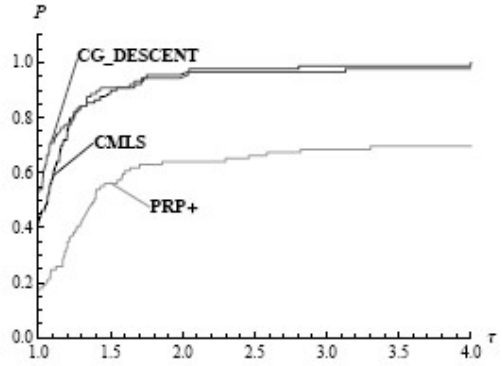

Fig. 1. Performance profiles relative to the CPU time.

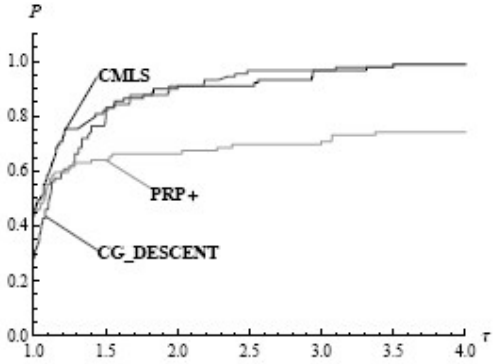

Fig. 2. Performance profiles relative to the number of iterations.

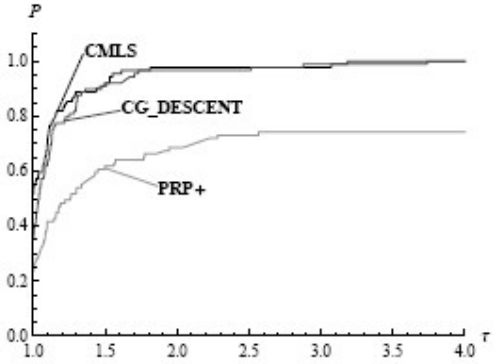

Fig. 3. Performance profiles relative to the number of function evaluated.

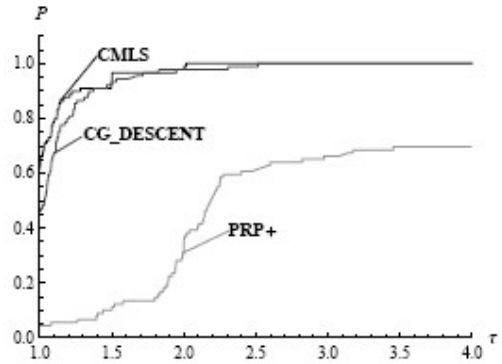

Fig. 4. Performance profiles relative to the number of gradient evaluated.

Acknowledgment:

The author would like to thank Prof. DongHui Li for giving us many valuable suggestions and comments, which improves this paper greatly. We are very grateful to Prof. W. W. Hager and Prof. J. Nocedal for providing their codes.

R E F E R E N C E S

[1] *M. R. Hestenes, E. L. Stiefel*, Methods of conjugate gradients for solving linear systems. J. Research Nat. Bur. Standards, 49(1952), 409-436.

[2] *R. Fletcher, C. Reeves*, Function minimization by conjugate gradients, Comput. J. 7(1964), 149-154

[3] *B. Polak, G. Ribi\`{ere*, Note sur la convergence de méthodes de directions conjuguées, Rev. Francaise Informat. Recherche Opérationnelle, 16(1969), 35-43.

[4] *B. T. Polyak*, The conjugate gradient method in extreme problems, USSR Comput. Math. Math. Phys., 9(1969), 94-112.

[5] *R. Fletcher*, Practical Method of Optimization I: Unconstrained Optimization, John Wiley & Sons. New York, 1987.

[6] *Y. Liu, C. Storey*, Efficient generalized conjugate gradient algorithms, Part 1: Theory, J. Optim. Theory Appl., 69(1991), 177-182.

[7] Y. *H. Dai, Y. Yuan*, A nonlinear conjugate gradient method with a strong global convergence property, SIAM J. Optim., 10(2000), 177-182.

[8] *M. J. D Powell*, Nonconvex minimization calculations and the conjugate gradient method, in: Lecture Notes in Mathematics, Springer-Verlag, Berlin, 1984.

[9] *J. C Gilbert, J. Nocedal*, Global convergence properties of conjugate gradient methods for optimization. SIAM. J. Optim., 2(1992), 21-42.

[10] *W. W. Hager, H. Zhang*, A survey of nonlinear conjugate gradient methods, Pacific J. Optim., 2(2006), 35-58.

[11] *W. W. Hager, H. Zhang*, A new conjugate gradient method with guaranteed descent and an efficient line search, SIAM J. Optim., 16(2005), 170-192.

[12] *L. Zhang, W.J. Zhou*, On the Global Convergence of the Hager-Zhang Conjugate Gradient Method with Armijo Line Search, Acta Mathematics Scientia (Chinese), 28(2008), 840-845.

[13] *D. Li, M. Fukushima*, A modified {BFGS method and its global convergence in nonconvex minimization, J. Comput. Appl. Math., 129(2001), 15-35.
[14] *G. Zoutendijk*, Nonlinear Programming, Computational Methods, in: J. Abadie(ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970.
[15] *P. Wolfe*, Convergence conditions for ascent methods, SIAM Rev., 11(1969), 226-235.
[16] *P. Wolfe*, Convergence conditions for ascent methods. II: Some corrections, SIAM Rev. 13(1969), 185-188.
[17] *J. J Moré, D.J Thuente*, Line search algorithms with guaranteed sufficient decrease, ACM Trans. Math. Software, 20(1995), 286-307.
[18] *I. Bongartz, A. R. Conn , N. I. M Gould, P. L. Toint*, CUTE: Constrained and unco-nstrained testing environments, ACM Trans. Math. Software, 21(1995), 123-160.
[19] *E. D. Dolan, J. J. Morv*, Benchmarking optimization software with performance profiles Math. Program., 91(2002), 201-213.