Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 825958, 14 pages
http://dx.doi.org/10.1155/2014/825958

Research Article

Several Guaranteed Descent Conjugate Gradient Methods for Unconstrained Optimization

San-Yang Liu and Yuan-Yuan Huang

School of Mathematics and Statistics, Xidian University, Xi'an 710071, China

Correspondence should be addressed to San-Yang Liu; liusanyang@126.com

Received 26 July 2013; Accepted 10 December 2013; Published 8 January 2014

Academic Editor: Kazutake Komori

This paper investigates a general form of guaranteed descent conjugate gradient methods which satisfies the descent condition $g_k^T d_k \leq -(1 - 1/(4\theta_k))\|g_k\|^2$ $(\theta_k > 1/4)$ and which is strongly convergent whenever the weak Wolfe line search is fulfilled. Moreover, we present several specific guaranteed descent conjugate gradient methods and give their numerical results for large-scale unconstrained optimization.

1. Introduction

Consider the following unconstrained optimization problem:

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

where R^n is the n-dimensional Euclidean space, $f : R^n \rightarrow R$ is continuously differentiable, and its gradient $g(x)$ is available.

Conjugate gradient methods are very efficient to solve problem (1) due to their simple iteration and their low memory requirements. For any given starting point $x_0 \in R^n$, they generate a sequence $\{x_k\}$ by the following recursive relation:

$$x_{k+1} = x_k + \alpha_k d_k, \quad (2)$$

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \quad (3)$$

where $g_k = g(x_k)$, α_k is a step length obtained by means of a one-dimensional search, and β_k is a scalar that characterizes the method. In general, the step length α_k in (2) is obtained by fulfilling the following weak Wolfe conditions [1, 2]:

$$\begin{gathered} f(x_k + \alpha_k d_k) - f(x_k) \leq \delta \alpha_k g_k^T d_k, \\ g_{k+1}^T d_k \geq \sigma g_k^T d_k, \end{gathered} \quad (4)$$

where $0 < \delta \leq \sigma < 1$. And different choices for the scalar β_k in (3) result in different nonlinear conjugate gradient methods. Well-known formulas for β_k are the Fletcher-Reeves (FR), Hestenes-Stiefel (HS), Polak-Ribiére-Polyak (PRP), Dai-Yuan (DY), and Liu-Storey (LS) formulas (see [3], [4], [5], [6], [7], and [8], resp.) and are given by

$$\beta_k^{\text{FR}} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \qquad \beta_k^{\text{HS}} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}},$$

$$\beta_k^{\text{PRP}} = \frac{g_k^T y_{k-1}}{\|g_{k-1}\|^2}, \qquad \beta_k^{\text{DY}} = \frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}}, \quad (5)$$

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

where $\|\cdot\|$ means the Euclidean norm and $y_k = g_{k+1} - g_k$. Their corresponding conjugate gradient methods are viewed as basic conjugate gradient methods. Among these basic conjugate gradient methods, the PRP and HS methods perform very similarly and perform better than other basic conjugate gradient methods [9]. While Powell [10] utilized an example illustrating that the PRP and HS methods may cycle without approaching any solution point, then modified versions of the PRP and HS methods were presented by many researchers (see, e.g., [11–16]).

The following (sufficient) descent condition,

$$g_k^T d_k \le -c\|g_k\|^2, \quad \forall k \ge 0,\ c > 0, \tag{6}$$

is very important for conjugate gradient methods, so we are particularly interested in the conjugate gradient methods with sufficient descent conditions. Up to now, there are many descent conjugate gradient methods proposed by researchers; please see [12, 16–19] and references therein.

One well-known guaranteed descent conjugate gradient method was proposed by Hager and Zhang [16, 20, 21] with

$$\beta_k^{\mathrm{HZ}} = \beta_k^{\mathrm{HS}} - \frac{2\|y_{k-1}\|^2}{\left(y_{k-1}^T d_{k-1}\right)^2} g_k^T d_{k-1}. \tag{7}$$

The method is designed based on the HS method and satisfies the sufficient descent condition (6) with $c = 7/8$ for any (inexact) line search. In [18], Zhang and Li proposed a general case of the HZ method with

$$\beta_k^{\mathrm{ZL}} = \frac{g_k^T\left(y_{k-1} - 2\left(\|y_{k-1}\|^2/\max\left\{h^2\|d_{k-1}\|^2, z_k\right\}\right)d_{k-1}\right)}{\max\left\{h^2\|d_{k-1}\|^2, z_k\right\}}, \tag{8}$$

where $h > 0$ and z_k is a scalar to be specified. It also satisfies the sufficient descent condition (6) with $c = 7/8$, and it is globally convergent in the sense of $\liminf_{k\to\infty}\|g_k\| = 0$. For $z_k = \|g_{k-1}\|^2$ and $z_k = -d_{k-1}^T g_{k-1}$, it becomes a descent PRP type method and a descent LS type method, respectively.

A more general form of the scalar β_k was suggested by Dai [22] and was defined as

$$\beta_k^D = \frac{g_k^T v_k}{\xi_k} - \frac{C\|v_k\|^2}{\xi_k^2} g_k^T d_{k-1}, \tag{9}$$

where $\xi_k \in R$, $v_k \in R^n$, and $C > 1/4$, while its convergence has not been given in [22]. More recently, Nakamura et al. [19] proved that the method is globally convergent in the sense of $\liminf_{k\to\infty}\|g_k\| = 0$ with the weak Wolfe conditions. Moreover, we say that a conjugate gradient method is strongly convergent if $\lim_{k\to\infty} g_k = 0$. Obviously, the later is stronger than the former, that is, the global convergence indicates that there exists at least one cluster point which is a stationary point of f, while the strong convergence means that every cluster point of $\{x_k\}$ will be a stationary point of f.

Observe formulas (8) and (9); we find that although β_k^{ZL} is a special case of formula (9), it has its own feature; that is, its denominator is lower bounded by $h^2\|d_{k-1}\|^2$. Motivated by this, we consider the general formula (9) by

$$\beta_k^{\mathrm{CGM}} = \frac{g_k^T v_k}{\max\{\xi_k, \epsilon\|d_{k-1}\|\}} - \frac{\theta_k\|v_k\|^2 g_k^T d_{k-1}}{\left(\max\{\xi_k, \epsilon\|d_{k-1}\|\}\right)^2}, \tag{10}$$

where $\theta_k > 1/4$ and $\epsilon > 0$, and prove that the general conjugate gradient method with β_k^{CGM} has better convergence properties; that is, it is strongly convergent. Another difference between the two formulas (9) and (10) is their choices of v_k. In order to guarantee convergence, the choices of v_k and ξ_k in (9) must satisfy the assumption that for all $k \ge 0$, there exist positive constants τ_1 and τ_2 such that $\|v_k\|^2|g_k^T d_{k-1}|/\xi_k^2 \le \tau_2\|s_{k-1}\|^2$ and $|g_k^T v_k/\xi_k| \le \tau_1\|s_{k-1}\|$ hold. If we choose $v_k = g_k$ and $\xi_k = 0.5(\|g_{k-1}\|^2 + |g_k^T d_{k-1}|)$, then whether the above assumption is satisfied is difficult to verify, while the requirement of v_k in (10) only is norm-bounded.

The rest of this paper is organized as follows. In Section 2, we describe the general form of guaranteed descent conjugate gradient methods with (10) and establish that the corresponding search directions always yield descent condition $g_k^T d_k \le -(1 - (1/4\theta_k))\|g_k\|^2$ ($\theta_k > 1/4$) independently of choices of the parameters v_k and ξ_k. And under some mild conditions, we prove its strong convergence with the weak Wolfe conditions. Moreover, we specifically design several efficient descent conjugate gradient methods combined with the features of the basic conjugate gradient methods above. In Section 3, we test the proposed conjugate gradient methods using the large-scale unconstrained problems in the CUTEr test library and compare them with the ZL method. Finally, we give some conclusions in Section 4.

2. Algorithm and Convergence

In this section, we describe the conjugate gradient method with (10) and show its strong convergence. And we give several specific conjugate gradient methods by combining formula (10) with some basic conjugate gradient methods. Firstly, we make the following assumption.

Assumption 1. Assume that $f : R^n \to R$ is bounded below in the level $\mathscr{L} = \{x \in R^n : f(x) \le f(x_0)\}$. And its gradient $g : R^n \to R^n$ is L-Lipschitz continuous in $X \subset R^n$; that is, there exists a constant $L > 0$ such that

$$\|g(x) - g(y)\| \le L\|x - y\|, \quad \forall x, y \in X. \tag{11}$$

Assumption 1 implies that there exists a positive constant $\widehat{\gamma}$ such that

$$\|g(x)\| \le \widehat{\gamma}, \quad \forall x \in \mathscr{L}. \tag{12}$$

Algorithm 2.
Step 0. Choose $\epsilon > 0$, $\varepsilon > 0$. Set $d_0 = -g_0$ and $k := 0$.

Step 1. If $\|g_k\|_\infty \le \varepsilon$, then stop; otherwise find α_k such that the weak Wolfe conditions (4) hold.

Step 2. Compute the new iterate by (2). Then generate the new search direction by (3) with β_k from (10). Set $k := k + 1$ and go to Step 1.

Next, we analyze the convergence properties of Algorithm 2. Under Assumption 1, we state the following *Zoutendijk condition*, which is originally given by Zoutendijk

[23] and Wolfe [1, 2] and is used to prove global convergence of nonlinear conjugate gradient methods.

Theorem 3. *Suppose that x_0 is a starting point for which Assumption 1 holds. Consider any iterative method in the form* (2)*, where d_k is a descent direction and α_k satisfies the weak Wolfe conditions* (4)*; then*

$$\sum_{k\geq 0} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} < +\infty. \tag{13}$$

The following lemma shows that the directions of Algorithm 2 satisfy the sufficient descent condition.

Lemma 4. *If d_k is generated by* (3) *with β_k from* (10) *and $\theta_k > 1/4$, then for every $k \geq 0$,*

$$g_k^T d_k \leq -\left(1 - \frac{1}{4\theta_k}\right) \|g_k\|^2. \tag{14}$$

Proof. Since $d_0 = -g_0$, then $g_0^T d_0 = -\|g_0\|^2$ which satisfies (14). For every $k \geq 1$, multiplying (3) by g_k, we have

$$\begin{aligned} g_k^T d_k &= -\|g_k\|^2 + \beta_k g_k^T d_{k-1} \\ &= -\|g_k\|^2 + \frac{g_k^T v_k}{\max\{\xi_k, \epsilon\|d_{k-1}\|\}} g_k^T d_{k-1} \\ &\quad - \frac{\theta_k \|v_k\|^2}{(\max\{\xi_k, \epsilon\|d_{k-1}\|\})^2} \left(g_k^T d_{k-1}\right)^2. \end{aligned} \tag{15}$$

Denote $u_k = g_k/\sqrt{2\theta_k}$ and $w_k = \sqrt{2\theta_k}(g_k^T d_{k-1})/\max\{\xi_k, \epsilon\|d_{k-1}\|\} v_k$. By applying the inequality $u_k^T w_k \leq 1/2(\|u_k\|^2 + \|w_k\|^2)$ to the second term in (15), we obtain the desired result. □

The lemma above is similar to Theorem 1.1 in [16]. And from this lemma, we can see that the descent property is independent of any line search and choices of the parameters v_k and ξ_k, while different choices of the parameters v_k, ξ_k, and θ_k may yield very different numerical behaviors.

Theorem 5. *Consider Algorithm 2, where α_k satisfies the weak Wolfe conditions* (4) *and β_k is defined by* (10) *with $\|v_k\|$ being bounded. Then, either $g_k = 0$ for some k or*

$$\lim_{k\to\infty} g_k = 0. \tag{16}$$

Proof. Suppose that $g_k \neq 0$ for all k. Utilizing (13) and (14), we have

$$\left(1 - \frac{1}{4\theta_k}\right) \sum_{k\geq 0} \frac{\|g_k\|^4}{\|d_k\|^2} < +\infty. \tag{17}$$

Since $\|v_k\|$ is bounded, then there must exist a large number $M < \infty$ such that $\|v_k\| \leq M$ for all k. By using the definition of β_k, we have

$$\begin{aligned} |\beta_k| &= \left| \frac{g_k^T v_k}{\max\{\xi_k, \epsilon\|d_{k-1}\|\}} - \frac{\theta_k \|v_k\|^2}{(\max\{\xi_k, \epsilon\|d_{k-1}\|\})^2} g_k^T d_{k-1} \right| \\ &\leq \left| \frac{g_k^T v_k}{\max\{\xi_k, \epsilon\|d_{k-1}\|\}} \right| + \frac{\theta_k \|v_k\|^2}{(\max\{\xi_k, \epsilon\|d_{k-1}\|\})^2} \left|g_k^T d_{k-1}\right| \\ &\leq \left(\frac{\|v_k\| \|d_{k-1}\|}{\max\{\xi_k, \epsilon\|d_{k-1}\|\}} + \frac{\theta_k \|v_k\|^2 \|d_{k-1}\|^2}{(\max\{\xi_k, \epsilon\|d_{k-1}\|\})^2} \right) \frac{\|g_k\|}{\|d_{k-1}\|} \\ &\leq \left(\frac{M}{\epsilon} + \frac{\theta_k M^2}{\epsilon^2} \right) \frac{\|g_k\|}{\|d_{k-1}\|}, \end{aligned} \tag{18}$$

where the second inequality is obtained using the Cauchy-Schwary inequality. Then, we have

$$\begin{aligned} \|d_k\| &\leq \|g_k\| + |\beta_k| \|d_{k-1}\| \\ &\leq \left(1 + \frac{M}{\epsilon} + \frac{\theta_k M^2}{\epsilon^2}\right) \|g_k\|. \end{aligned} \tag{19}$$

Inserting this upper bound for d_k in (17) yields

$$\sum_{k\geq 0} \|g_k\|^2 < \infty, \tag{20}$$

which implies (16). □

Now, we propose several specific versions of Algorithm 2. Since hybrid conjugate gradient methods are regarded as better performing conjugate gradient methods in practice, then the specific methods are designed as hybrid versions based on some basic conjugate gradient methods. As mentioned in Section 1, the PRP and HS methods are two efficient methods, so the first specific hybrid method is designed using the features of the PRP and HS methods with

$$\begin{aligned} \beta_k^{\mathrm{CGM1}} &= \frac{g_k^T y_{k-1}}{\max\left\{\max\left\{\|g_{k-1}\|^2, d_{k-1}^T y_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}} \\ &\quad - \frac{2\|y_{k-1}\|^2 g_k^T d_{k-1}}{\left(\max\left\{\max\left\{\|g_{k-1}\|^2, d_{k-1}^T y_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}\right)^2}. \end{aligned} \tag{21}$$

Since the LS method has a similar structure to the PRP method, then the second hybrid method is proposed based on the PRP and LS methods with

$$\begin{aligned} \beta_k^{\mathrm{CGM2}} &= \frac{g_k^T y_{k-1}}{\max\left\{\max\left\{\|g_{k-1}\|^2, -g_{k-1}^T d_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}} \\ &\quad - \frac{2\|y_{k-1}\|^2 g_k^T d_{k-1}}{\left(\max\left\{\max\left\{\|g_{k-1}\|^2, -g_{k-1}^T d_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}\right)^2}. \end{aligned} \tag{22}$$

The third one is derived from the FR and DY methods with

$$\beta_k^{\mathrm{CGM3}} = \frac{\|g_k\|^2}{\max\left\{\max\left\{\|g_{k-1}\|^2, d_{k-1}^T y_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}} - \frac{2\|g_k\|^2 g_k^T d_{k-1}}{\left(\max\left\{\max\left\{\|g_{k-1}\|^2, d_{k-1}^T y_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}\right)^2}. \quad (23)$$

And the last one is proposed with

$$\beta_k^{\mathrm{CGM4}} = \frac{g_k^T y_{k-1}^*}{\max\left\{\max\left\{\|g_{k-1}\|^2, d_{k-1}^T y_{k-1}^*\right\}, \epsilon\|d_{k-1}\|\right\}} - \frac{2\|y_{k-1}^*\|^2 g_k^T d_{k-1}}{\left(\max\left\{\max\left\{\|g_{k-1}\|^2, d_{k-1}^T y_{k-1}^*\right\}, \epsilon\|d_{k-1}\|\right\}\right)^2}, \quad (24)$$

where $y_{k-1}^* = y_{k-1} + \epsilon\|g_{k-1}\|\alpha_{k-1}d_{k-1}$ is similar to that of [24] and utilizes some secant condition. In addition, many conjugate gradient methods have been proposed based on different secant conditions; please refer to [15, 25–28] for further information.

From Assumption 1 and inequality (19), we have that g_k and d_k are norm-bounded for all k; then global convergence properties of the four new hybrid descent conjugate gradient methods can be given following the proof of Algorithm 2.

Here, the parameter θ_k in (10) is chosen to be the constant number 2. It also could have other choices, such as $\theta_k = \max\{1/4+\epsilon, |\xi_k|/\|v_k\|^2\}$, while, in most cases, $\theta_k = 2$ performs better than other choices.

3. Numerical Experiments

In this section, we did some numerical experiments to test the performances of the proposed methods and compared them with the ZL method. Numerical results reported in [18] showed that the ZL method with $z_k = -d_{k-1}^T g_{k-1}$ in (8), denoted by TDLS method, performs better than the HZ method and the descent PRP type method, so we only compared the proposed methods with the TDLS method. All codes were written in Matlab and run on a desktop computer with an Intel(R) Xeon(R) 2.40 GHZ CPU, 6.00 GB of RAM, and Linux operating system Ubuntu 8.04. All test problems were drawn from the CUTEr test library [29, 30] and were accessed from within Matlab R2012a by using Matlab interface. We were particularly interested in large-scale problems, so the dimension of each test problem was at least 100.

For all the implemented methods, the step size α_k satisfied the weak Wolfe conditions (4) with $\sigma = 0.1$ and $\delta = 0.9$ and its initial guess was generated by the rules in [21], the value of h in TDLS method was taken to be 10^{-5} following [18], and the stopping criterion was

$$\|g_k\|_\infty \le \max\{\epsilon, \epsilon(1+f_k)\}, \quad (25)$$

where $\epsilon = 10^{-6}$ and $f_k = f(x_k)$.

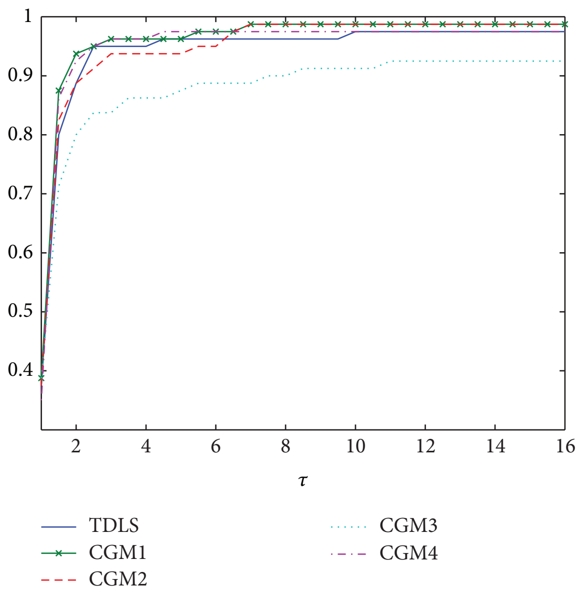

FIGURE 1: Performance profile based on the number of function evaluations.

The numerical results were reported in Table 1, where Problem, Dim, Iter, Nf, Ng, and CPU represent the name of the test problems, the dimension, the number of iterations, the number of function evaluations, the number of gradient evaluations, and the CPU time elapsed in seconds, respectively, and "–" means that the method failed to achieve a prescribed accuracy when the number of iterations exceeded 50,000 or the cost function generated a "NaN."

The performances of all methods were evaluated using the profiles of Dolan and Morè [31]. That is, we plotted the fraction P of the test problems for which each of the methods was within a factor τ. Obviously, the top curve represented the most roust one within the same factor τ. And the left curve represented the fastest one to solve the same percentage of the test problems. Figures 1, 2, and 3 showed the performance profiles referring to the number of function evaluations, the number of gradient evaluations, and CPU time, respectively. These figures revealed that all the test methods were efficient and the CGM1, CGM2, and CGM4 methods were comparable with the TDLS method, while the CGM3 method performed relatively bad. It is worth noting that the CGM1, CGM2 and CGM4 methods are hybrid versions related to the PRP method, so they inherit the good numerical performance of the PRP method. Among the three methods and the TDLS method, the CGM1 method performed more efficiently than the CGM2 method and more robustly than the TDLS and CGM4 methods, so the CGM1 method was the winner of these test methods.

4. Conclusions

This paper has studied a general form of guaranteed descent conjugate gradient methods and has proven that whenever

Table 1: Numerical results for test problems from the CUTEr library.

Name (Dim)	Method	Iter/Nf/Ng/CPU
ARGLINA (200)	TDLS	1/3/2/0.002
	CGM1	1/3/2/0.001
	CGM2	1/3/2/0.002
	CGM3	1/3/2/0.001
	CGM4	1/3/2/0.001
ARGLINB (100)	TDLS	7/275/273/0.109
	CGM1	6/259/257/0.099
	CGM2	7/275/273/0.106
	CGM3	7/132/130/0.052
	CGM4	6/259/257/0.100
ARGLINC (100)	TDLS	8/213/212/0.084
	CGM1	6/104/103/0.040
	CGM2	8/306/304/0.117
	CGM3	8/232/231/0.091
	CGM4	6/105/104/0.041
ARWHEAD (10000)	TDLS	10/311/304/0.492
	CGM1	8/17/9/0.025
	CGM2	9/118/110/0.186
	CGM3	30/351/332/0.569
	CGM4	20/527/520/0.882
BDQRTIC (5000)	TDLS	1156/2348/1226/2.650
	CGM1	1599/3226/1654/3.742
	CGM2	1399/2827/1452/3.237
	CGM3	495/1089/623/1.166
	CGM4	1477/2994/1547/3.541
BIGGSB1 (5000)	TDLS	2500/5001/2501/2.990
	CGM1	2500/5001/2501/3.349
	CGM2	2500/5001/2501/3.141
	CGM3	2500/5001/2501/2.957
	CGM4	2501/5003/2503/3.602
BOX (10000)	TDLS	10/28/22/0.074
	CGM1	9/25/20/0.065
	CGM2	8/22/16/0.054
	CGM3	41/177/144/0.428
	CGM4	9/24/19/0.063
BROWNAL (200)	TDLS	50/107/64/0.043
	CGM1	9/25/19/0.011
	CGM2	12/26/17/0.011
	CGM3	56/114/64/0.045
	CGM4	25/60/44/0.027
BROYDN7D (1000)	TDLS	342/691/354/0.461
	CGM1	327/655/328/0.456
	CGM2	320/644/327/0.455
	CGM3	311/623/312/0.417
	CGM4	319/639/320/0.448
BRYBND (5000)	TDLS	443/859/498/1.150
	CGM1	52/109/60/0.166
	CGM2	275/550/290/0.740
	CGM3	127/260/134/0.337
	CGM4	41/86/48/0.133

Table 1: Continued.

Name (Dim)	Method	Iter/Nf/Ng/CPU
CHAINWOO (4000)	TDLS	300/591/345/0.625
	CGM1	276/540/321/0.629
	CGM2	248/484/284/0.545
	CGM3	4107/158741/162278/1.605
	CGM4	260/515/299/0.591
COSINE (5000)	TDLS	5/17/14/0.025
	CGM1	5/17/14/0.025
	CGM2	5/17/14/0.024
	CGM3	6/19/15/0.025
	CGM4	5/17/14/0.024
CRAGGLVY (5000)	TDLS	56/113/57/0.221
	CGM1	58/117/59/0.234
	CGM2	57/115/58/0.223
	CGM3	63/130/67/0.246
	CGM4	57/115/58/0.235
CURLY10 (1000)	TDLS	1621/3256/1637/0.836
	CGM1	1583/3179/1598/0.923
	CGM2	1626/3266/1642/0.862
	CGM3	2404/4821/2419/1.146
	CGM4	1583/3179/1598/0.874
CURLY20 (600)	TDLS	1804/3619/1817/0.889
	CGM1	1797/3605/1810/1.007
	CGM2	1735/3481/1748/0.872
	CGM3	3463/6937/3476/1.573
	CGM4	1809/3629/1822/0.978
CURLY30 (1000)	TDLS	2573/5158/2587/1.910
	CGM1	2587/5186/2601/1.914
	CGM2	2629/5270/2643/1.892
	CGM3	18710/37432/18724/12.670
	CGM4	2586/5184/2600/1.942
DIXMAANA (9000)	TDLS	7/15/8/0.018
	CGM1	8/17/9/0.021
	CGM2	7/15/8/0.019
	CGM3	18/37/19/0.044
	CGM4	8/17/9/0.021
DIXMAANB (9000)	TDLS	9/19/10/0.024
	CGM1	9/19/10/0.025
	CGM2	9/19/10/0.025
	CGM3	10/21/11/0.025
	CGM4	9/19/10/0.024
DIXMAANC (6000)	TDLS	10/21/11/0.020
	CGM1	10/21/11/0.020
	CGM2	10/21/11/0.020
	CGM3	10/21/11/0.018
	CGM4	10/21/11/0.021
DIXMAAND (9000)	TDLS	11/23/12/0.030
	CGM1	12/25/13/0.033
	CGM2	11/23/12/0.032
	CGM3	13/27/14/0.033
	CGM4	12/25/14/0.033

Table 1: Continued.

Name (Dim)	Method	Iter/Nf/Ng/CPU
DIXMAANE (9000)	TDLS	340/681/341/0.878
	CGM1	337/675/338/0.900
	CGM2	340/681/341/0.898
	CGM3	413/827/414/1.036
	CGM4	337/675/338/0.909
DIXMAANF (9000)	TDLS	252/505/253/0.675
	CGM1	253/507/254/0.682
	CGM2	252/505/253/0.671
	CGM3	250/501/251/0.609
	CGM4	253/507/254/0.711
DIXMAANG (9000)	TDLS	251/503/252/0.636
	CGM1	248/497/249/0.666
	CGM2	251/503/252/0.658
	CGM3	238/477/239/0.578
	CGM4	248/497/249/0.660
DIXMAANH (9000)	TDLS	246/493/247/0.621
	CGM1	247/495/248/0.663
	CGM2	246/493/247/0.646
	CGM3	246/493/247/0.597
	CGM4	247/495/248/0.689
DIXMAANI (9000)	TDLS	1856/3713/1857/4.770
	CGM1	1801/3603/1802/4.932
	CGM2	1912/3825/1913/4.981
	CGM3	1586/3173/1587/3.904
	CGM4	1801/3603/1802/4.965
DIXMAANJ (1500)	TDLS	651/1303/652/0.396
	CGM1	519/1039/520/0.349
	CGM2	651/1303/652/0.426
	CGM3	474/949/475/0.288
	CGM4	518/1037/519/0.359
DIXMAANK (3000)	TDLS	194/389/195/0.252
	CGM1	207/415/208/0.322
	CGM2	194/389/195/0.252
	CGM3	198/397/199/0.232
	CGM4	207/415/208/0.291
DIXMAANL (3000)	TDLS	168/337/169/0.226
	CGM1	186/373/187/0.277
	CGM2	168/337/169/0.238
	CGM3	195/391/196/0.227
	CGM4	186/373/187/0.281
DIXON3DQ (1000)	TDLS	2001/4003/2004/0.810
	CGM1	1227/2455/1230/0.577
	CGM2	1225/2451/1228/0.535
	CGM3	2003/4007/2005/0.744
	CGM4	1065/2131/1068/0.481
DQDRTIC (10000)	TDLS	7/15/8/0.025
	CGM1	7/15/8/0.025
	CGM2	7/15/8/0.025
	CGM3	6/13/7/0.020
	CGM4	11/23/13/0.042

Table 1: Continued.

Name (Dim)	Method	Iter/Nf/Ng/CPU
DQRTIC (1000)	TDLS	29/59/30/0.013
	CGM1	29/59/30/0.014
	CGM2	29/59/30/0.014
	CGM3	29/59/30/0.012
	CGM4	29/59/30/0.014
EG2 (1000)	TDLS	3/7/4/0.003
	CGM1	3/7/4/0.003
	CGM2	3/7/4/0.002
	CGM3	3/7/4/0.002
	CGM4	3/7/4/0.003
EIGENALS (420)	TDLS	6491/12989/6500/5.960
	CGM1	6620/13247/6629/6.377
	CGM2	6683/13373/6692/6.238
	CGM3	7358/14723/7367/6.544
	CGM4	6632/13271/6641/6.398
EIGENBLS (110)	TDLS	391/791/401/0.149
	CGM1	340/683/343/0.151
	CGM2	356/722/373/0.153
	CGM3	355/714/359/0.134
	CGM4	379/761/382/0.173
EIGENCLS (132)	TDLS	543/1101/565/0.226
	CGM1	540/1090/551/0.259
	CGM2	564/1144/586/0.259
	CGM3	595/1191/596/0.242
	CGM4	586/1177/592/0.287
ENGVAL1 (10000)	TDLS	12/25/13/0.040
	CGM1	13/27/14/0.045
	CGM2	12/25/13/0.041
	CGM3	12/25/13/0.039
	CGM4	13/27/14/0.046
EXTROSNB (10000)	TDLS	10044/20294/10301/23.500
	CGM1	—
	CGM2	—
	CGM3	—
	CGM4	9342/18870/9573/23.720
FLETCBV2 (500)	TDLS	591/1183/593/0.321
	CGM1	542/1085/544/0.329
	CGM2	542/1085/544/0.321
	CGM3	582/1165/584/0.315
	CGM4	540/1081/542/0.336
FLETCBV3 (10000)	TDLS	2/22/21/0.087
	CGM1	2/22/21/0.083
	CGM2	2/22/21/0.081
	CGM3	2/21/20/0.077
	CGM4	2/22/21/0.083
FLETCHBV (10000)	TDLS	2/21/20/0.078
	CGM1	2/21/20/0.077
	CGM2	2/21/20/0.081
	CGM3	2/20/19/0.071
	CGM4	2/21/20/0.076

Table 1: Continued.

Name (Dim)	Method	Iter/Nf/Ng/CPU
FLETCHCR (1000)	TDLS	7312/15021/7844/3.960
	CGM1	6944/14415/7514/4.052
	CGM2	7754/16085/8474/4.494
	CGM3	4255/8519/4267/2.125
	CGM4	6847/14134/7330/4.121
FMINSRF2 (961)	TDLS	240/481/241/0.131
	CGM1	245/491/246/0.150
	CGM2	240/481/241/0.142
	CGM3	254/510/256/0.139
	CGM4	246/493/247/0.154
FMINSURF (121)	TDLS	76/154/78/0.025
	CGM1	71/146/75/0.027
	CGM2	77/158/82/0.028
	CGM3	90/184/94/0.028
	CGM4	71/146/75/0.029
FREUROTH (500)	TDLS	36/79/46/0.021
	CGM1	32/69/39/0.018
	CGM2	15/36/23/0.009
	CGM3	32/70/40/0.016
	CGM4	32/69/39/0.020
GENHUMPS (200)	TDLS	2659/5502/2894/1.160
	CGM1	2508/5123/2631/1.207
	CGM2	2633/5410/2819/1.146
	CGM3	62/191/146/0.037
	CGM4	2376/4887/2542/1.111
GENROSE (1000)	TDLS	2540/5140/2615/1.390
	CGM1	2400/4836/2446/1.395
	CGM2	2548/5163/2635/1.395
	CGM3	2105/4240/2143/1.059
	CGM4	2391/4820/2440/1.470
HILBERTA (100)	TDLS	197/395/203/0.174
	CGM1	197/395/203/0.188
	CGM2	197/395/203/0.183
	CGM3	105/211/113/0.094
	CGM4	242/485/250/0.234
HILBERTB (100)	TDLS	5/11/6/0.005
	CGM1	5/11/6/0.005
	CGM2	5/11/6/0.005
	CGM3	5/11/6/0.005
	CGM4	5/11/6/0.005
LIARWHD (5000)	TDLS	28/67/47/0.075
	CGM1	21/44/25/0.048
	CGM2	31/67/44/0.067
	CGM3	996/1994/999/1.794
	CGM4	21/44/25/0.047
MANCINO (150)	TDLS	11/23/12/0.198
	CGM1	11/23/12/0.198
	CGM2	11/23/12/0.198
	CGM3	10/21/11/0.180
	CGM4	11/23/12/0.199

Table 1: Continued.

Name (Dim)	Method	Iter/Nf/Ng/CPU
MODBEALE (2000)	TDLS	523/1042/702/0.970
	CGM1	3455/6821/3664/5.597
	CGM2	3352/6735/3392/5.255
	CGM3	666/1348/693/1.121
	CGM4	851/1715/896/1.455
MOREBV (1000)	TDLS	425/851/426/0.208
	CGM1	391/783/392/0.216
	CGM2	391/783/392/0.208
	CGM3	363/727/364/0.176
	CGM4	391/783/392/0.226
MSQRTALS (100)	TDLS	310/629/321/0.112
	CGM1	309/627/320/0.132
	CGM2	305/619/316/0.124
	CGM3	358/725/369/0.129
	CGM4	309/627/320/0.137
NCB20 (1010)	TDLS	276/557/290/0.573
	CGM1	255/514/270/0.547
	CGM2	303/612/313/0.641
	CGM3	407/817/415/0.785
	CGM4	255/513/266/0.499
NCB20B (1000)	TDLS	45/91/48/0.093
	CGM1	60/121/62/0.126
	CGM2	60/121/62/0.125
	CGM3	55/111/57/0.112
	CGM4	60/121/62/0.127
NONCVXU2 (1000)	TDLS	1032/2065/1033/0.695
	CGM1	848/1697/849/0.622
	CGM2	1024/2049/1025/0.728
	CGM3	813/1627/814/0.498
	CGM4	829/1659/830/0.575
NONCVXUN (1000)	TDLS	1577/3155/1578/1.070
	CGM1	1571/3143/1572/1.154
	CGM2	1174/2349/1175/0.776
	CGM3	12594/25189/12595/7.857
	CGM4	1493/2987/1494/1.042
NONDIA (10000)	TDLS	14/40/31/0.061
	CGM1	22/52/36/0.081
	CGM2	17/43/32/0.068
	CGM3	16/35/23/0.051
	CGM4	10/22/14/0.034
NONDQUAR (5000)	TDLS	13250/26511/13362/17.300
	CGM1	7857/15717/7868/11.170
	CGM2	7697/15402/7748/10.330
	CGM3	6296/12601/6306/7.575
	CGM4	7754/15513/7847/11.470
NONSCOMP (10000)	TDLS	36/73/37/0.086
	CGM1	32/65/33/0.075
	CGM2	36/73/37/0.081
	CGM3	39/79/40/0.083
	CGM4	32/65/33/0.077

Table 1: Continued.

Name (Dim)	Method	Iter/Nf/Ng/CPU
OSCIPATH (1000)	TDLS	10/19/14/0.005
	CGM1	10/19/14/0.006
	CGM2	10/19/14/0.005
	CGM3	10/19/14/0.005
	CGM4	10/19/14/0.006
OSCIGRAD (10000)	TDLS	78/157/79/0.256
	CGM1	81/163/82/0.271
	CGM2	79/159/80/0.260
	CGM3	97/195/98/0.303
	CGM4	81/163/82/0.276
PENALTY1 (1000)	TDLS	45/110/68/0.025
	CGM1	44/105/66/0.027
	CGM2	49/120/77/0.029
	CGM3	89/190/102/0.042
	CGM4	45/106/67/0.027
POWELLSG (5000)	TDLS	68/138/74/0.087
	CGM1	56/114/61/0.078
	CGM2	122/248/132/0.166
	CGM3	2828/5657/2829/3.180
	CGM4	233/482/254/0.323
POWER (1000)	TDLS	134/269/135/0.048
	CGM1	116/233/117/0.048
	CGM2	134/269/135/0.053
	CGM3	—
	CGM4	116/233/117/0.050
QUARTC (1000)	TDLS	29/59/30/0.012
	CGM1	29/59/30/0.014
	CGM2	29/59/30/0.017
	CGM3	29/59/30/0.012
	CGM4	29/59/30/0.016
SCHMVETT (5000)	TDLS	14/29/15/0.074
	CGM1	14/29/15/0.076
	CGM2	14/29/15/0.074
	CGM3	15/31/16/0.077
	CGM4	14/29/15/0.075
SENSORS (100)	TDLS	26/85/63/0.391
	CGM1	17/44/29/0.206
	CGM2	27/80/58/0.376
	CGM3	27/64/40/0.292
	CGM4	17/44/29/0.207
SINQUAD (100)	TDLS	40/89/52/0.018
	CGM1	31/71/46/0.014
	CGM2	40/89/52/0.016
	CGM3	32/73/47/0.012
	CGM4	31/71/46/0.014
SPARSINE (1000)	TDLS	4344/8689/4345/3.120
	CGM1	4467/8935/4468/3.305
	CGM2	4378/8757/4379/3.205
	CGM3	5455/10911/5456/3.698
	CGM4	4467/8935/4468/3.369

Table 1: Continued.

Name (Dim)	Method	Iter/Nf/Ng/CPU
SPARSQUR (1000)	TDLS	19/39/20/0.011
	CGM1	19/39/20/0.012
	CGM2	19/39/20/0.012
	CGM3	19/39/20/0.011
	CGM4	19/39/20/0.012
SPMSRTLS (499)	TDLS	113/233/122/0.056
	CGM1	114/235/123/0.062
	CGM2	114/235/123/0.060
	CGM3	109/225/118/0.052
	CGM4	114/235/123/0.063
SROSENBR (5000)	TDLS	11/24/15/0.017
	CGM1	12/26/17/0.024
	CGM2	11/24/15/0.016
	CGM3	27/57/33/0.036
	CGM4	12/26/17/0.018
TESTQUAD (3000)	TDLS	1470/2941/1471/1.140
	CGM1	1434/2869/1435/1.369
	CGM2	1472/2945/1473/1.271
	CGM3	1507/3015/1508/1.179
	CGM4	1748/3497/1749/1.803
TOINTGSS (10000)	TDLS	3/7/4/0.017
	CGM1	3/7/4/0.016
	CGM2	3/7/4/0.017
	CGM3	3/7/4/0.016
	CGM4	3/7/4/0.016
TQUARTIC (10000)	TDLS	32/81/56/0.104
	CGM1	32/72/45/0.091
	CGM2	27/71/51/0.092
	CGM3	97/203/114/0.230
	CGM4	26/58/38/0.078
TRIDIA (10000)	TDLS	1115/2231/1116/2.110
	CGM1	1115/2231/1116/2.208
	CGM2	1115/2231/1116/2.139
	CGM3	1116/2233/1117/1.944
	CGM4	1119/2239/1120/2.303
VARDIM (1000)	TDLS	37/75/39/0.016
	CGM1	37/75/39/0.018
	CGM2	37/75/39/0.017
	CGM3	38/78/41/0.016
	CGM4	37/77/41/0.019
VAREIGVL (1000)	TDLS	73/198/125/0.075
	CGM1	78/210/132/0.084
	CGM2	70/190/120/0.075
	CGM3	—
	CGM4	76/204/128/0.083
WOODS (1000)	TDLS	433/897/484/0.215
	CGM1	358/756/414/0.204
	CGM2	225/515/306/0.133
	CGM3	250/529/289/0.126
	CGM4	220/485/291/0.135

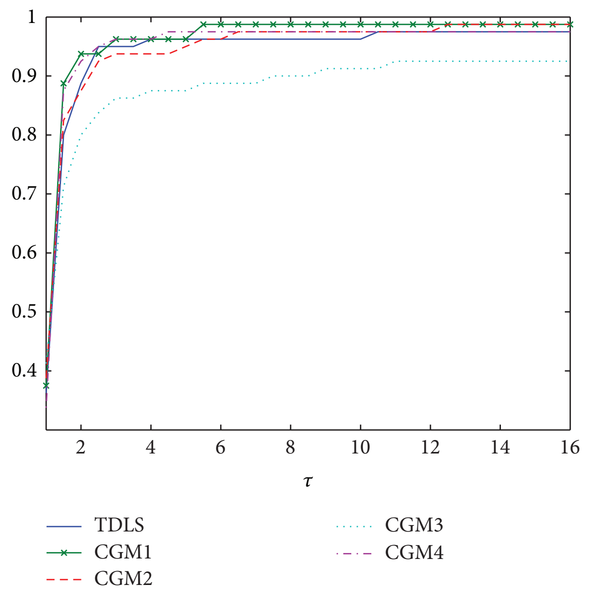

FIGURE 2: Performance profile based on the number of gradient evaluations.

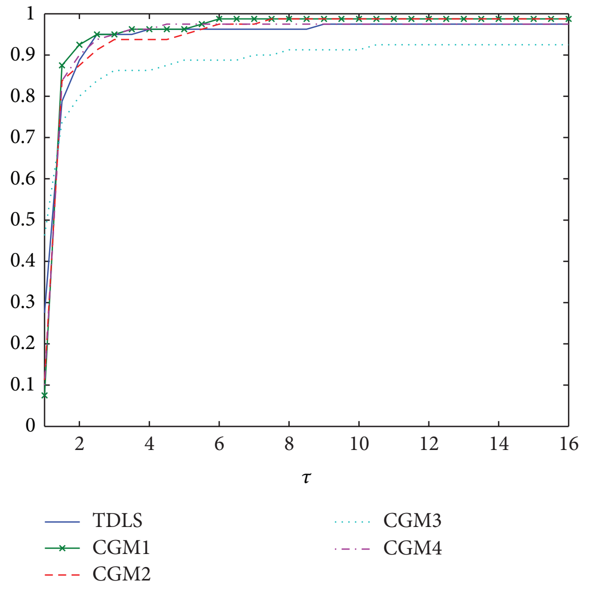

FIGURE 3: Performance profile based on the CPU time.

the weak Wolfe conditions are fulfilled, it is strongly convergent with $\lim_{k\to\infty} g_k = 0$. Then, we gave several specific guaranteed descent conjugate gradient methods and investigated their numerical behaviors using the test problems from the CUTEr library. From the numerical results, we can conclude that the specific methods are efficient to solve unconstrained nonlinear problems.

More recently, a class of conjugate gradient methods [28] was proposed based on different secant conditions. They followed the form of the HZ method and satisfied sufficient descent condition. While not all of the global convergence properties of them were obtained for a general objective function, then our further investigation is to improve these methods from theory analysis and numerical efficiency.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors are very grateful to the associate editor and the referees for their valuable comments. This work is supported by the National Science Foundation of China, no. 61373174.

References

[1] P. Wolfe, "Convergence conditions for ascent methods," *SIAM Review*, vol. 11, no. 2, pp. 226–235, 1969.

[2] P. Wolfe, "Convergence conditions for ascent methods. II: some corrections," *SIAM Review*, vol. 13, no. 2, pp. 185–188, 1971.

[3] R. Fletcher and C. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, pp. 149–154, 1964.

[4] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[5] E. Polak and G. Ribière, "Note sur la convergence de méthodes de directions conjuguées," *Revue Française D'informatique et de Recherche Opérationnelle, Série Rouge*, vol. 3, no. 16, pp. 35–43, 1969.

[6] B. T. Polyak, "The conjugate gradient method in extremal problems," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 4, pp. 94–112, 1969.

[7] Y. H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property," *SIAM Journal on Optimization*, vol. 10, no. 1, pp. 177–182, 2000.

[8] Y. Liu and C. Storey, "Efficient generalized conjugate gradient algorithms, part 1: theory," *Journal of Optimization Theory and Applications*, vol. 69, no. 1, pp. 129–137, 1991.

[9] Y. H. Dai and Q. Ni, "Testing different conjugate gradient methods for large-scale unconstrained optimization," *Journal of Computational Mathematics*, vol. 21, no. 3, pp. 311–320, 2003.

[10] M. J. D. Powell, "Nonconvex minimization calculations and the conjugate gradient method," in *Numerical Analysis*, vol. 1066 of *Lecture Notes in Mathematics*, pp. 122–141, Springer, Berlin, Germany, 1984.

[11] J. C. Gilbert and J. Nocedal, "Global convergence properties of conjugate gradient methods for optimization," *SIAM Journal on Optimization*, vol. 2, no. 1, pp. 21–42, 1992.

[12] L. Zhang, W. Zhou, and D.-H. Li, "A descent modified Polak-Ribière-Polyak conjugate gradient method and its global convergence," *IMA Journal of Numerical Analysis*, vol. 26, no. 4, pp. 629–640, 2006.

[13] Y. F. Hu and C. Storey, "Global convergence result for conjugate gradient methods," *Journal of Optimization Theory and Applications*, vol. 71, no. 2, pp. 399–405, 1991.

[14] Y. H. Dai and Y. Yuan, "An efficient hybrid conjugate gradient method for unconstrained optimization," *Annals of Operations Research*, vol. 103, no. 1–4, pp. 33–47, 2001.

[15] Y.-H. Dai and L.-Z. Liao, "New conjugacy conditions and related nonlinear conjugate gradient methods," *Applied Mathematics & Optimization*, vol. 43, no. 1, pp. 87–101, 2001.

[16] W. W. Hager and H. Zhang, "A new conjugate gradient method with guaranteed descent and an efficient line search," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 170–192, 2006.

[17] L. Zhang, W. Zhou, and D. Li, "Global convergence of a modified Fletcher–Reeves conjugate gradient method with Armijo-type line search," *Numerische Mathematik*, vol. 104, no. 4, pp. 561–572, 2006.

[18] L. Zhang and J. Li, "A new globalization technique for nonlinear conjugate gradient methods for nonconvex minimization," *Applied Mathematics and Computation*, vol. 217, no. 24, pp. 10295–10304, 2011.

[19] W. Nakamura, Y. Narushima, and H. Yabe, "Nonlinear conjugate gradient methods with sufficient descent properties for unconstrained optimization," *Journal of Industrial and Management Optimization*, vol. 9, no. 3, pp. 595–619, 2013.

[20] W. W. Hager and H. Zhang, "A survey of nonlinear conjugate gradient methods," *Pacific Journal of Optimization*, vol. 2, no. 1, pp. 35–58, 2006.

[21] W. W. Hager and H. Zhang, "Algorithm 851: CG_DESCENT, a conjugate gradient method with guaranteed descent," *ACM Transactions on Mathematical Software*, vol. 32, no. 1, pp. 113–137, 2006.

[22] Y. H. Dai, "Nonlinear conjugate gradient methods," in *Wiley Encyclopedia of Operations Research and Management Science*, J. J. Cochran, L. A. Cox Jr., P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, Eds., vol. 8, John Wiley & Sons, Hoboken, NJ, USA, 2011.

[23] G. Zoutendijk, "Nonlinear programming, computational methods," in *Integer and Nonlinear Programming*, J. Abadie, Ed., pp. 37–86, North-Holland, Amsterdam, The Netherlands, 1970.

[24] D.-H. Li and M. Fukushima, "A modified BFGS method and its global convergence in nonconvex minimization," *Journal of Computational and Applied Mathematics*, vol. 129, no. 1-2, pp. 15–35, 2001.

[25] H. Yabe and M. Takano, "Global convergence properties of nonlinear conjugate gradient methods with modified secant condition," *Computational Optimization and Applications*, vol. 28, no. 2, pp. 203–225, 2004.

[26] S. Babaie-Kafaki, R. Ghanbari, and N. Mahdavi-Amiri, "Two new conjugate gradient methods based on modified secant equations," *Journal of Computational and Applied Mathematics*, vol. 234, no. 5, pp. 1374–1386, 2010.

[27] K. Sugiki, Y. Narushima, and H. Yabe, "Globally convergent three-term conjugate gradient methods that use secant conditions and generate descent search directions for unconstrained optimization," *Journal of Optimization Theory and Applications*, vol. 153, no. 3, pp. 733–757, 2012.

[28] Y. Narushima and H. Yabe, "Conjugate gradient methods based on secant conditions that generate descent search directions for unconstrained optimization," *Journal of Computational and Applied Mathematics*, vol. 236, no. 17, pp. 4303–4317.

[29] I. Bongartz, A. R. Conn, N. Gould, and P. L. Toint, "CUTE: constrained and unconstrained testing environment," *ACM Transactions on Mathematical Software*, vol. 21, no. 1, pp. 123–160, 1995.

[30] N. I. M. Gould, D. Orban, and P. L. Toint, "CUTEr and SifDec: a constrained and unconstrained testing environment, revisited," *ACM Transactions on Mathematical Software*, vol. 29, no. 4, pp. 373–394, 2003.

[31] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.