Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 147025, 14 pages
http://dx.doi.org/10.1155/2013/147025

Research Article

A Globally Convergent Hybrid Conjugate Gradient Method and Its Numerical Behaviors

Yuan-Yuan Huang,[1] San-Yang Liu,[1] Xue-Wu Du,[2] and Xiao-Liang Dong[1]

[1] *Department of Mathematics, Xidian University, Xi'an 710071, China*
[2] *College of Mathematics Science, Chong Qing Normal University, Chong Qing 40047, China*

Correspondence should be addressed to Yuan-Yuan Huang; yyuanhuang@126.com and San-Yang Liu; liusanyang@126.com

Received 21 December 2012; Revised 12 March 2013; Accepted 27 March 2013

Academic Editor: Martin Weiser

We consider a hybrid Dai-Yuan conjugate gradient method. We confirm that its numerical performance can be improved provided that this method uses a practical steplength rule developed by Dong, and the associated convergence is analyzed as well.

1. Introduction

Consider the following problem of finding an $x \in R^n$ such that

$$g(x) = 0, \tag{1}$$

where R^n is the n-dimensional Euclidean space and $g : R^n \to R^n$ is continuous. Throughout this paper, this problem corresponds to optimality condition of a certain problem of minimizing $f : R^n \to R$ which may be not easy to calculate or cannot be expressed as elementary functions. When the dimension n is large, conjugate gradient methods can be efficient to solve problem (1). For any given starting point $x_0 \in R^n$, a sequence $\{x_k\}$ is generated by the following recursive relation:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$

with

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

where α_k is a steplength, d_k is a descent direction, g_k stands for $g(x_k)$, and β_k is a parameter. Different choices of β_k result in different nonlinear conjugate gradient methods. The Dai-Yuan (DY) formula [1] and the Hestenes-Stiefel (HS) formula [2] are two famous ones, and are given by

$$\beta_k^{\mathrm{DY}} = \frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}}, \qquad \beta_k^{\mathrm{HS}} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \tag{4}$$

respectively, where $\|\cdot\|$ means the 2-norm and $y_{k-1} = g_k - g_{k-1}$. Other well-known formulae for β_k such as the Fletcher-Reeves formula [3], the Polak-Ribière-Polyak formula [4, 5], and the Hager-Zhang formula [6], please refer to [7, 8] for further survey.

In [1], the steplength α_k is obtained by the following weak Wolfe line search:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k; \tag{5}$$

$$\sigma g_k^T d_k \leq g(x_k + \alpha_k d_k)^T d_k, \tag{6}$$

where $0 < \delta < \sigma < 1$. With the same line search, two hybrid versions related to the DY method and the HS method

were proposed in [9], which generate the parameter β_k by

$$\beta_k^{\mathrm{DYHS}} = \max\left\{-\frac{1-\sigma}{1+\sigma}\beta_k^{\mathrm{DY}}, \min\left\{\beta_k^{\mathrm{DY}}, \beta_k^{\mathrm{HS}}\right\}\right\},$$
$$\beta_k^{\mathrm{DYHS+}} = \max\left\{0, \min\left\{\beta_k^{\mathrm{DY}}, \beta_k^{\mathrm{HS}}\right\}\right\}, \tag{7}$$

respectively. And initial numerical results in [10] suggested that the two hybrid conjugate gradient methods (abbreviated as DYHS and DYHS+, resp.) are efficient, especially the DYHS+ method performed better.

The line search plays an important role in the efficiency of conjugate gradient methods. Hager and Zhang [6] showed that the first condition (5) of the weak Wolfe line search limits the accuracy of a conjugate gradient method to the order of the square root of the machine precision (see also [11, 12]); thus, in order to get higher precision, they proposed approximate Wolfe conditions [6, 7],

$$\sigma g_k^T d_k \le g(x_k + \alpha_k d_k)^T d_k \le (2\delta - 1)\, g_k^T d_k, \tag{8}$$

where $0 < \delta < 1/2$ and $\delta < \sigma < 1$, which are usually used combining with the weak Wolfe line search. However, there is no theory to guarantee convergence in [6–8]. By following a referee's suggestion, we adapt the approximate Wolfe conditions to a Dai-Yuan hybrid conjugate gradient method and investigate its numerical performances.

More recently, Dong designed a practical Armijo-type steplength rule [13] only using gradient, please see [14] for a more conceptual version, the steplength α_k is chosen by the following steps: choose $0 < t < 1$, compute some appropriate initial steplength $\rho_k > 0$, determine a real number μ_k (see (12)) and find α_k to be the largest $\alpha \in \{\rho_k t^j \mid j = 0, 1, 2, \ldots\}$ such that

$$g(x_k + \alpha d_k)^T d_k - \frac{1}{2}\mu_k \alpha \|d_k\|^2 \le \sigma g_k^T d_k, \quad k \le K;$$
$$g(x_k + \alpha d_k)^T d_k + \frac{1}{2}\max\{-\mu_k, 0\}\,\alpha\|d_k\|^2 \le \sigma g_k^T d_k, \quad k > K, \tag{9}$$

where K is a nonnegative integer and $0 < \sigma < 1$.

The main differences between the practical steplength rule and the weak Wolfe conditions are that the former does not require function evaluations, it is in high accuracy and has broader application scope. The feature of high accuracy is supported by the corresponding theory analysis in [6, 11]. Numerical results reported in [13] also imply that the line search (9) is efficient and highly accurate. So, it is meaningful to imbed the line search into the hybrid conjugate gradient method with parameter $\beta_k^{\mathrm{DYHS+}}$ and to check its efficiency.

This paper is to solve problem (1), which corresponds to optimality condition of a certain problem of minimizing $f : R^n \to R$. If the original function f cannot be expressed as elementary functions, the weak Wolfe conditions cannot be applied directly, while the practical steplength rule (9) and the approximate Wolfe conditions (8) can be used to solve this kind of nonlinear unconstrained minimization problems. So, in order to investigate the numerical performances of the two modified methods with steplength rules (9) and (8) and to confirm their broader application scope, two classes of test problems are selected. One class is composed of unconstrained nonlinear optimization problems from the CUTEr library, and the other class is composed of some boundary value problems.

The rest of this paper is organized as follows. In Section 2, we give some basic definitions and properties used in this paper. In Section 3, we describe two modified versions of the Dai-Yuan hybrid conjugate gradient method with the line search (9) and the approximate Wolfe conditions (8) and illustrate that if g is Lipschitz continuous, the former version is convergent in the sense that $\liminf_{k\to+\infty}\|g_k\| = 0$. In Section 4, we test the modified hybrid conjugate gradient methods over two classes of standard test problems and compare them with the DYHS method and the DYHS+ method. Finally, some conclusions are given in Section 5.

2. Preliminaries

In this section, we give some basic definitions and related properties which will be used in the following discussions.

Assumption 1. Assume $g : R^n \to R^n$ is L-Lipschitz continuous in $X \subset R^n$, that is, there exists a constant $L > 0$ such that

$$\|g(x) - g(y)\| \le L\|x - y\|, \quad \forall x, y \in X. \tag{10}$$

And its original function $f : R^n \to R$ is bounded below in the level set $\mathscr{L} = \{x \in R^n : f(x) \le f(x_0)\}$.

Definition 2. A mapping $g : R^n \to R^n$ is said to be μ-monotone on the set $X \subset R^n$ if there exists $\mu \in R$ such that

$$\langle g(x) - g(y), x - y\rangle \ge \mu\|x - y\|^2, \quad \forall x, y \in X. \tag{11}$$

By using the definition above, we know that if the gradient $g : R^n \to R^n$ is L-Lipschitz continuous, then for any given $x_k, d_k \in R^n$, the gradient g must be μ_k-monotone along the ray $\{x_k + \alpha d_k : \alpha \ge 0\}$ for some $\mu_k \in [-L, L]$. Then, how to evaluate such μ_k? In [13], it is suggested to evaluate using the following approximation formula:

$$\mu_k \approx \frac{(g(x_k + \alpha_{k-1}d_k) - g_k)^T d_k}{\alpha_{k-1}\|d_k\|^2}. \tag{12}$$

Lemma 3. *For any given $x, d \in R^n$ and for all $\alpha \ge 0$, if $f : R^n \to R$ is continuously differentiable and the given gradient $g : R^n \to R^n$ is μ-monotone along the ray $\{x + \alpha d : \alpha \ge 0\}$, then the following inequality holds:*

$$f(x + \alpha d) \le f(x) + \alpha g(x + \alpha d)^T d - \frac{1}{2}\mu\alpha^2\|d\|^2. \tag{13}$$

Proof. Please see [13] for a detailed proof. □

3. Algorithm and Its Convergence

In this section, we first formally describe the hybrid conjugate gradient method. Then, we give its two modified versions and

illustrate that the modified version with steplength rule (9) is convergent.

Algorithm 4. Step 0. Choose $\epsilon > 0$. Set $d_0 = -g_0$, $\alpha_{-1} = 1$ and $k := 0$.

Step 1. If $\|g_k\|_\infty \le \epsilon$, then stop, otherwise find α_k such that certain steplength rule holds.

Step 2. Compute the new iterate by (2) and the new search direction by

$$d_{k+1} = -g_{k+1} + \beta_{k+1}^{\mathrm{DYHS+}} d_k. \tag{14}$$

Set $k := k + 1$ and go to Step 1.

Denote

$$\rho_k = \max\left\{10^{-9}, \frac{1}{\max\{10^{-9}, |\mu_k|\}} \min\left\{10^{9}, \frac{-g_k^T d_k}{\|g_k\|^2}\right\} \frac{\|g_k\|^2}{\|d_k\|^2}\right\}. \tag{15}$$

(i) We abbreviate Algorithm 4 as MDYHS+, if the steplength rule in Step 1 is finding α_k to be the largest $\alpha \in \{\rho_k t^j \mid j = 0, 1, 2, \ldots\}$ $(0 < t < 1)$ such that

$$g(x_k + \alpha d_k)^T d_k + \frac{1}{2} \max\{-\mu_k, 0\} \alpha \|d_k\|^2 \le \sigma g_k^T d_k \tag{16}$$

holds, where μ_k is defined in (12) and $0 < \sigma < 1$.

(ii) And we abbreviate Algorithm 4 as MDYHS + 1, if α_k in Step 1 is located to satisfy the approximate Wolfe conditions (8). It should be noticed that on the face of it, the approximate Wolfe conditions only use gradient information to locate the steplength, while they require function evaluations in practical implementations in [6–8], please refer to [8, pages 120–125] for details. So, the algorithm in [6–8] to generate the steplength satisfying the approximate Wolfe conditions (8) is not applicable. Here, we determine the steplength α_k of the MDYHS+1 method following the inexact line search strategies of [15, Algorithm 2.6]. Detailed steps are described in Algorithm 6. The initial value of α_k is taken to be ρ_k.

Remark 5. The choice of ρ_k comes from [13]. Since it is important to use current information about the algorithm and the problem to make an initial guess of α_k^*, the author of [13] uses the relation $g(x_k + \alpha_k^* d_k)^T d_k = 0$ and

$$\frac{\left(g\left(x_k + \alpha_k^* d_k\right) - g_k\right)^T d_k}{\alpha_k^*} \approx \frac{\left(g\left(x_k + \alpha_{k-1} d_k\right) - g_k\right)^T d_k}{\alpha_{k-1}} \tag{17}$$

to give an approximation to the optimal steplength through

$$\alpha_k^* \approx \frac{-g_k^T d_k}{|\mu_k| \|g_k\|^2} \frac{\|g_k\|^2}{\|d_k\|^2}. \tag{18}$$

Now, we describe the line search algorithm in [15], which is very close to one suggested in [16].

Algorithm 6. Step 0. Set $u = 0$ and $v = +\infty$. Choose $\alpha > 0$. Set $j := 0$.

Step 1. If α does not satisfy

$$g(x_k + \alpha d_k)^T d_k \le (2\delta - 1) g_k^T d_k, \tag{19}$$

then set $j := j + 1$, and go to Step 2. If α does not satisfy

$$\sigma g_k^T d_k \le g(x_k + \alpha d_k)^T d_k, \tag{20}$$

then set $j := j + 1$, and go to Step 3. Otherwise, set $\alpha_k := \alpha$, and return.

Step 2. Set $v = \alpha, \alpha = (u + v)/2$. Then go to Step 1.

Step 3. Set $u = \alpha, \alpha = 2u$. Then go to Step 1.

Next, we analyze the convergence properties of the MDYHS+ method.

Lemma 7. *Consider the previous MDYHS+ method. If $g_k \ne 0$ for all $k \ge 0$, then the steplength α_k is well-defined, namely, α_k can be found to satisfy* (16) *after a finite number of trials. The search direction d_k satisfies the sufficient descent condition*

$$g_k^T d_k \le -\|g_k\|^2 < 0. \tag{21}$$

Furthermore, $g_{k+1}^T d_k < 0$.

Proof. We prove the desired results by induction. When $k = 0$, by using $d_0 = -g_0$, we have

$$g_0^T d_0 = -\|g_0\|^2 < 0. \tag{22}$$

We now show that the steplength α_0 will be determined within a finite number of trials. Otherwise, for any $j = 0, 1, 2, \ldots$, the following inequality holds

$$g\left(x_0 + \rho_0 t^j d_0\right)^T d_0 + \frac{1}{2} \max\{-\mu_0, 0\} \rho_0 t^j \|d_0\|^2 > \sigma g_0^T d_0, \tag{23}$$

where $0 < \sigma < 1$. Since g is continuous, taking the limits with respect to j on the both sides of (23) yields

$$g_0^T d_0 \ge \sigma g_0^T d_0, \tag{24}$$

this is a contradiction. Then, α_0 is well-defined and

$$g_1^T d_0 = g(x_0 + \alpha_0 d_0)^T d_0 \le \sigma g_0^T d_0 < 0. \tag{25}$$

Assume that (21) holds for $k - 1$ and $g_k^T d_{k-1} < 0$. A similar discussion to the $k = 0$ case yields α_k is well-defined. Multiplying g_k^T by

$$d_k = -g_k + \beta_k^{\mathrm{DYHS+}} d_{k-1}, \tag{26}$$

we have that

$$g_k^T d_k = -\|g_k\|^2 + \beta_k^{\mathrm{DYHS+}} g_k^T d_{k-1}, \tag{27}$$

it follows from $g_k^T d_{k-1} < 0$ and $\beta_k^{\mathrm{DYHS+}} \geq 0$ that

$$g_k^T d_k \leq -\|g_k\|^2 < 0. \tag{28}$$

Then,

$$\begin{aligned} g_{k+1}^T d_k &\leq g_{k+1}^T d_k + \frac{1}{2}\max\{-\mu_k, 0\}\,\alpha_k \|d_k\|^2 \\ &\leq \sigma g_k^T d_k < 0. \end{aligned} \tag{29}$$

□

Theorem 8. *Consider the previos MDYHS+ method, and assume that g is μ_k-monotone in the interval $\{x_k + \alpha d_k : 0 \leq \alpha \leq \rho_k\}$, where μ_k is the very μ_k defined in* (12)*. If Assumption 1 holds, then either*

$$\liminf_{k \to +\infty} \|g_k\| = 0, \tag{30}$$

or

$$\sum_{k \geq 0} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} < +\infty. \tag{31}$$

Proof. Since g is μ_k-monotone, by using Lemma 3, we obtain

$$\begin{aligned} &f(x_k + \alpha_k d_k) \\ &\quad \leq f(x_k) + \alpha_k g(x_k + \alpha_k d_k)^T d_k - \frac{1}{2}\mu_k \alpha_k^2 \|d_k\|^2. \end{aligned} \tag{32}$$

Furthermore,

$$\begin{aligned} &f(x_k + \alpha_k d_k) \\ &\quad \leq f(x_k) + \alpha_k g(x_k + \alpha_k d_k)^T d_k + \frac{1}{2}\max\{-\mu_k, 0\}\,\alpha_k^2 \|d_k\|^2. \end{aligned} \tag{33}$$

Combining (33) with (16) yields

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \sigma \alpha_k g_k^T d_k, \tag{34}$$

which, together with (21), implies

$$\sum_{i=0}^{k} \alpha_i \|g_i\|^2 \leq \sigma^{-1}\left(f(x_0) - f(x_{k+1})\right). \tag{35}$$

Next, we follow [13] to consider two possible cases.

Case 1 ($\sum_{k=0}^{+\infty} \alpha_k = +\infty$). Consider Assumption 1, we have

$$\begin{aligned} \min_{0 \leq i \leq k} \|g_i\|^2 &\leq \sigma^{-1}\left(f(x_0) - f(x_{k+1})\right)\left(\sum_{i=0}^{k} \alpha_i\right)^{-1} \\ &\longrightarrow 0, \quad \text{as } k \longrightarrow +\infty. \end{aligned} \tag{36}$$

Thus, (30) holds.

Case 2 ($\sum_{k=0}^{+\infty} \alpha_k < +\infty$). From this case, we have $\alpha_k \to 0$. Since α_k is the maximal number in $\{\rho_k t^j : j = 0, 1, 2, \ldots\}$ such that (16) holds, then there must exist a natural number $N > 0$, when $k \geq N$, there exists $j_k > 1$ such that $\alpha_k = \rho_k t^{j_k}$ and $t^{-1}\alpha_k$ violates (16), namely,

$$\begin{aligned} &g\left(x_k + t^{-1}\alpha_k d_k\right)^T d_k + \frac{1}{2}\max\{-\mu_k, 0\}\, t^{-1}\alpha_k \|d_k\|^2 \\ &\quad > \sigma g_k^T d_k, \end{aligned} \tag{37}$$

then

$$\begin{aligned} &\frac{\left(g\left(x_k + t^{-1}\alpha_k d_k\right) - g_k\right)^T d_k}{t^{-1}\|d_k\|^2} + \frac{1}{2}\frac{\max\{-\mu_k, 0\}\, t^{-1}\alpha_k \|d_k\|^2}{t^{-1}\|d_k\|^2} \\ &\quad \geq \frac{(\sigma - 1)\, t g_k^T d_k}{\|d_k\|^2}. \end{aligned} \tag{38}$$

Combining it with the Lipschitz continuity of g and the fact that $|\mu_k| \leq L$ yields

$$\alpha_k \geq \frac{-2(1-\sigma)\, t g_k^T d_k}{3L\|d_k\|^2}, \tag{39}$$

which, together with (34), implies

$$f(x_k) - f(x_{k+1}) \geq \gamma \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2}, \tag{40}$$

with $\gamma = 2\sigma(1-\sigma)t/(3L) > 0$. Using (40) recursively, we obtain

$$\gamma \sum_{i=N}^{k} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} \leq f(x_N) - f(x_{k+1}). \tag{41}$$

Combining it with Assumption 1 yields

$$\sum_{k \geq N} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} < +\infty, \tag{42}$$

namely,

$$\sum_{k \geq 0} \frac{\left(g_k^T d_k\right)^2}{\|d_k\|^2} < +\infty. \tag{43}$$

□

Theorem 9. *The MDYHS+ method is convergent in the sense that* $\liminf_{k \to \infty} \|g_k\| = 0$.

Proof. The proof of Theorem 9 is much like the convergence analysis of the DY method [1, Theorem 3.3], so the corresponding details are omitted here. □

Table 1: List of test problems from the CUTEr library.

Prob.	Name	n
1	ARWHEAD	1000
2	BDQRTIC	500, 1000
3	BIGGSB1	1000, 5000
4	BOX	2000
5	BROYDN7D	1000
6	BROWNAL	200
7	BRYBND	5000
8	CHAINWOO	100
9	COSINE	150
10	CRAGGLVY	1000
11	CURLY10	1000
12	CURLY20	1000
13	CURLY30	1000
14	DIXMAANA	3000
15	DIXMAANB	3000
16	DIXMAANC	1500
17	DIXMAAND	3000
18	DIXMAANE	3000, 6000
19	DIXMAANF	3000, 9000
20	DIXMAANG	3000, 9000
21	DIXMAANH	3000
22	DIXMAANI	300
23	DIXMAANJ	300
24	DIXMAANK	300
25	DIXMAANL	1500
26	DIXON3DQ	1000
27	DQRTIC	5000
28	EIGENALS	420
29	EIGENBLS	110
30	EIGENCLS	132, 462
31	ENGVAL1	1000
32	EXTROSNB	1000
33	FLETCBV2	1000
34	FLETCHCR	500
35	FMINSRF2	1024
36	FMINSURF	1024
37	FREUROTH	1000
38	GENHUMPS	5000
39	GENROSE	500
40	HILBERTA	200
41	HILBERTB	300
42	JIMACK	3549
43	LIARWHD	5000
44	MANCINO	150
45	MODBEALE	200
46	MSQRTALS	529
47	MOREBV	500
48	NCB20	1010
49	NCB20B	500, 2000
50	NONCVXU2	1000
51	NONCVXUN	100
52	NONDIA	10000
53	NONDQUAR	100
54	NONSCOMP	10000
55	PENALTY1	500
56	PENALTY2	100
57	POWELLSG	5000
58	POWER	100
59	QUARTC	1000, 10000
60	SCHMVETT	1000, 5000
61	SENSORS	100
62	SPARSQUR	5000, 10000
63	SROSENBR	10000
64	TESTQUAD	3000, 5000
65	TOINTGSS	1000
66	TQUARTIC	1000
67	TRIDIA	10000
68	VAREIGVL	500
69	WOODS	4000

4. Numerical Experiments

In this section, we did some numerical experiments to test the performances of the MDYHS+ method and the MDYHS + 1 method. One purpose of this section is to compare them with the DYHS method and the DYHS+ method. The other purpose is to confirm their broader application scope by solving boundary value problems. So, two classes of test problems were selected here. One class was drawn from the CUTEr library [17, 18], and the other class came from [19]. More information was described in the following subsections.

For the MDYHS+ method, we set $\sigma = 10^{-4}$ and $t = 0.5$. For the MDYHS+1 method, we followed [8] to choose $\sigma = 0.9$ and $\delta = 0.1$. And for the hybrid conjugate gradient methods DYHS and DYHS+, the values of δ and σ in (5) and (6) were taken to be 0.01 and 0.1, respectively. The initial value of α_k was taken to be $1/\|g_0\|$ for the first iteration and $\alpha_{k-1} g_{k-1}^T d_{k-1}/(g_k^T d_k)$ for $k \geq 1$ (see [10]). For all the methods, the largest trial times of choosing steplength at each iteration was taken to be $j_{\max} = 30$, and the stopping criterion used was $\|g_k\|_\infty \leq \epsilon$. In order to understand the numerical performance of each method deeply, we did numerical experiments with $\epsilon = 10^{-i}, i = 3, 6, 9, 12$. Our computations were carried out using Matlab R2012a on a desktop computer with an Intel(R) Xeon(R) 2.40 GHZ CPU and 6.00 GB of RAM. The operating system is Linux: Ubuntu 8.04.

4.1. Tested by Some Problems in the CUTEr Library. In this subsection, we implemented four different hybrid conjugate gradient methods and compared their numerical performances. Because the DYHS method and the DYHS+ method

Table 2: Numerical results for test problems from the CUTEr library.

ϵ	Algorithm	#1 (1000)	#2 (500)	#2 (1000)	#3 (1000)	#3 (5000)	#4 (2000)
10^{-3}	MDYHS+	21/36/0.015	134/221/0.086	102/166/0.09	533/1066/0.34	738/1476/1.48	31/62/0.05
	MDYHS+1	6/19/0.005	99/109/0.056	75/98/0.06	500/500/0.23	1999/1999/2.99	9/14/0.015
	DYHS+	3/7/0.003	89/238/0.080	77/209/0.11	255/412/0.14	255/412/0.44	8/73/0.05
	DYHS	4/9/0.004	108/274/0.093	187/477/0.23	255/412/0.14	255/412/0.42	15/93/0.06
10^{-6}	MDYHS+	41/76/0.031	177/307/0.13	163/288/0.15	738/2214/0.49	2738/5476/5.54	82/104/0.09
	MDYHS+1	8/21/0.007	172/182/0.09	106/129/0.09	500/500/0.25	2500/2500/3.76	11/16/0.016
	DYHS+	7/68/0.029	851/20742/5.97	427/8252/3.37	3884/6188/2.29	13726/22027/22.90	—
	DYHS	5/11/0.005	—	—	4048/6461/2.42	13759/22098/22.98	24/166/0.12
10^{-9}	MDYHS+	61/11/0.051	228/409/0.15	214/390/0.20	1299/2598/0.82	3923/7846/8.05	68/204/0.13
	MDYHS+1	8/21/0.007	251/261/0.145	139/162/0.11	500/500/0.25	2501/2501/3.73	12/17/0.018
	DYHS+	7/68/0.028	1223/30509/9.06	900/20509/8.28	8831/14347/5.34	44013/71661/74.32	—
	DYHS	—	—	—	7754/12601/4.07	42394/69153/71.49	—
10^{-12}	MDYHS+	80/153/0.068	307/549/0.20	318/581/0.28	1579/3157/1.00	5817/11634/11.85	99/192/0.17
	MDYHS+1	12/54/0.016	323/3330.17	248/271/0.19	502/502/0.22	2504/2504/3.81	33/38/0.043
	DYHS+	—	3020/77907/22.34	1618/36542/14.75	13400/21926/7.40	—	—
	DYHS	—	—	—	12338/20212/6.60	—	—
ϵ	Algorithm	#5 (1000)	#6 (200)	#7 (5000)	#8 (100)	#9 (150)	#10 (1000)
10^{-3}	MDYHS+	367/643/0.81	27/53/0.03	35/57/0.15	248/453/0.12	9/14/0.004	66/93/0.08
	MDYHS+1	310/316/0.50	3/14/0.006	37/70/0.16	230/247/0.08	6/14/0.004	70/91/0.08
	DYHS+	322/545/0.66	2/4/0.007	57/152/0.32	289/669/0.15	8/35/0.009	61/155/0.11
	DYHS	321/553/0.67	2/9/0.007	55/148/0.30	307/701/0.17	8/35/0.009	58/154/0.11
10^{-6}	MDYHS+	392/693/0.83	54/105/0.06	55/97/0.23	347/651/0.16	18/32/0.008	104/169/0.14
	MDYHS+1	334/340/0.54	8/19/0.010	45/78/0.19	398/415/0.14	9/17/0.005	109/130/0.12
	DYHS+	352/868/0.96	14/42/0.03	92/223/0.48	438/996/0.23	11/45/0.010	200/3163/2.11
	DYHS	339/595/0.72	14/42/0.03	96/232/0.50	456/1028/0.25	11/45/0.010	—
10^{-9}	MDYHS+	414/737/0.88	842/1355/0.88	75/137/0.33	471/899/0.22	28/52/0.012	142/245/0.19
	MDYHS+1	357/363/0.58	9/20/0.011	54/87/0.22	561/578/0.22	13/21/0.006	150/171/0.16
	DYHS+	442/3072/3.01	20/59/0.04	135/309/0.68	—	31/93/0.022	573/13215/8.23
	DYHS	—	20/59/0.04	137/322/0.70	—	155/390/0.083	—
10^{-12}	MDYHS+	436/778/0.93	—	95/177/0.42	612/1128/0.32	36/67/0.016	187/322/0.26
	MDYHS+1	379/385/0.61	—	62/95/0.25	724/741/0.26	16/24/0.007	193/214/0.21
	DYHS+	—	—	167/376/0.82	—	—	—
	DYHS	—	—	171/398/0.88	—	—	—
ϵ	Algorithm	#11 (1000)	#12 (1000)	#13 (1000)	#14 (3000)	#15 (3000)	#16 (1500)
10^{-3}	MDYHS+	3764/7333/2.91	5422/10414/5.14	6065/11565/6.74	15/26/0.03	10/14/0.02	13/20/0.011
	MDYHS+1	4199/4222/2.42	4950/4979/3.54	5796/5826/4.68	6/14/0.015	7/18/0.020	8/23/0.009
	DYHS+	7418/14260/5.42	8775/17315/8.59	13564/27163/15.10	4/17/0.015	4/18/0.017	4/22/0.009
	DYHS	8798/16878/6.07	9565/18913/8.71	11328/22680/11.97	4/17/0.016	4/18/0.016	4/24/0.009
10^{-6}	MDYHS+	6334/12473/4.99	9030/17630/8.73	1004/19523/11.08	24/44/0.05	20/34/0.039	24/42/0.022
	MDYHS+1	6768/6791/3.88	9230/9259/6.83	10415/10445/8.34	7/15/0.020	8/19/0.020	9/24/0.010
	DYHS+	—	—	—	5/20/0.02	5/21/0.021	5/26/0.010
	DYHS	—	—	—	6/22/0.02	5/21/0.020	6/29/0.011
10^{-9}	MDYHS+	9104/18013/7.15	12496/24559/12.32	12979/25367/14.46	34/64/0.07	30/54/0.061	34/62/0.031
	MDYHS+1	9806/9829/6.02	12937/12966/8.85	14193/14223/11.29	8/16/0.022	9/20/0.021	10/25/0.011
	DYHS+	—	—	—	38/807/0.70	6/24/0.022	8/35/0.014
	DYHS	—	—	—	—	7/27/0.025	8/35/0.014
10^{-12}	MDYHS+	—	—	—	44/84/0.09	41/76/0.087	45/84/0.041
	MDYHS+1	13775/13798/8.59	—	—	10/18/0.024	10/21/0.0222	12/27/0.013
	DYHS+	—	—	—	109/2633/2.32	9/30/0.037	141/533/0.212
	DYHS	—	—	—	—	8/29/0.034	22/71/0.03

Table 3: Numerical results for test problems from the CUTEr library.

ϵ	Algorithm	#17 (3000)	#18 (3000)	#18 (6000)	#19 (3000)	#19 (9000)	#20 (3000)
10^{-3}	MDYHS+	15/23/0.022	81/155/0.17	90/173/0.29	31/42/0.05	31/42/0.11	27/36/0.045
	MDYHS+1	9/28/0.027	70/77/0.12	80/87/0.21	66/76/0.12	44/54/0.16	61/76/0.101
	DYHS+	5/27/0.021	61/90/0.15	61/146/0.22	22/70/0.06	41/108/0.21	38/102/0.096
	DYHS	5/27/0.021	60/86/0.12	60/140/0.20	30/88/0.07	35/93/0.18	29/82/0.072
10^{-6}	MDYHS+	26/45/0.054	235/463/0.51	317/627/1.08	278/513/0.57	453/845/2.09	214/391/0.46
	MDYHS+1	11/30/0.029	229/236/0.38	310/317/0.78	176/186/0.28	279/289/0.93	177/192/0.28
	DYHS+	7/33/0.03	315/400/0.62	373/759/1.19	287/608/0.60	473/958/2.03	252/527/0.52
	DYHS	8/37/0.031	321/420/0.66	434/884/1.28	275/564/0.52	487/979/2.15	243/503/0.50
10^{-9}	MDYHS+	36/65/0.071	322/637/0.72	443/879/1.50	457/871/0.99	740/1419/3.43	381/725/0.81
	MDYHS+1	13/32/0.033	317/324/0.51	432/439/1.09	332/342/0.52	508/518/1.70	320/335/0.53
	DYHS+	9/40/0.031	591/1166/1.21	991/1993/3.13	647/1334/1.37	1004/2179/4.47	458/919/0.95
	DYHS	10/43/0.036	564/1209/1.14	797/1593/2.34	527/1200/1.07	982/2089/4.06	529/1133/1.12
10^{-12}	MDYHS+	46/85/0.10	461/915/1.00	609/1211/2.10	614/1185/1.33	984/1907/4.57	517/997/1.08
	MDYHS+1	15/34/0.035	379/386/0.63	529/536/1.34	462/472/0.78	779/789/2.60	465/480/0.79
	DYHS+	54/949/0.71	—	—	—	—	—
	DYHS	285/698/0.61	—	—	—	—	—
ϵ	Algorithm	#20 (9000)	#21 (3000)	#22 (300)	#23 (300)	#24 (300)	#25 (1500)
10^{-3}	MDYHS+	26/34/0.52	23/29/0.04	89/170/0.047	29/39/0.012	29/39/0.012	27/37/0.02
	MDYHS+1	38/53/0.15	59/78/0.12	102/109/0.042	37/47/0.016	40/54/0.021	42/61/0.03
	DYHS+	17/38/0.11	21/72/0.06	101/231/0.057	36/60/0.023	26/78/0.018	30/92/0.04
	DYHS	30/89/0.16	27/83/0.07	98/222/0.050	28/79/0.017	26/78/0.016	27/85/0.035
10^{-6}	MDYHS+	395/735/1.78	261/484/0.55	525/1042/0.30	706/1358/0.37	785/1486/0.40	2363/4633/2.32
	MDYHS+1	274/289/0.97	177/196/0.29	485/492/0.21	400/410/0.16	430/444/0.17	1425/1444/1.03
	DYHS+	334/674/1.40	283/611/0.59	1267/2465/0.63	1000/1856/0.49	989/1893/0.49	273/568/0.27
	DYHS	347/723/1.43	259/545/0.48	1385/2676/0.65	1050/2041/0.51	1290/2479/0.58	289/602/0.26
10^{-9}	MDYHS+	688/1321/3.19	436/834/0.93	685/1362/0.38	1281/2508/0.69	1236/2388/0.65	3799/7505/3.72
	MDYHS+1	475/490/1.63	316/336/0.53	578/585/0.23	890/900/0.40	929/943/0.37	3539/3558/2.58
	DYHS+	1131/2375/4.87	500/1039/1.02	2892/6062/1.58	2920/5644/1.53	2425/4934/1.32	10163/19925/9.19
	DYHS	1027/2087/4.14	553/1123/1.07	3361/6720/1.74	2681/5228/1.40	3045/6257/1.55	12152/24343/10.54
10^{-12}	MDYHS+	899/1743/4.17	591/1144/1.27	995/1982/0.55	1822/3590/1.03	1753/3422/0.97	5812/11531/5.71
	MDYHS+1	767/782/2.59	462/481/0.75	940/947/0.37	1299/1309/0.53	1356/1370/0.56	5972/5991/4.30
	DYHS+	—	—	—	—	—	—
	DYHS	—	—	—	—	—	—
ϵ	Algorithm	#26 (1000)	#27 (5000)	#28 (420)	#29 (110)	#30 (132)	#30 (462)
10^{-3}	MDYHS+	1040/2080/0.63	24/24/0.036	1594/2930/2.94	256/372/0.14	416/699/0.24	2213/3889/4.16
	MDYHS+1	1000/1000/0.46	24/24/0.036	3014/3034/3.89	280/286/0.12	318/333/0.15	1076/1097/1.51
	DYHS+	390/720/0.22	38/282/0.22	1264/3001/2.39	229/479/0.14	413/906/0.27	1355/2925/2.53
	DYHS	390/720/0.21	39/271/0.20	1485/3475/2.62	236/520/0.13	404/899/0.25	1415/2980/2.55
10^{-6}	MDYHS+	1289/2578/0.79	29/29/0.039	2364/4441/4.26	416/646/0.22	641/1149/0.41	3234/5894/6.22
	MDYHS+1	1000/1000/0.48	29/29/0.039	5946/5966/7.64	1862/1895/0.83	554/569/0.26	2008/2029/2.80
	DYHS+	8951/17009/5.20	49/358/0.24	3407/8062/6.28	394/835/0.25	800/1756/0.53	3353/7227/6.35
	DYHS	6985/13160/3.81	48/330/0.22	3446/8045/6.30	447/961/0.26	781/1714/0.49	3499/7433/6.74
10^{-9}	MDYHS+	2072/4144/1.32	33/33/0.049	2873/5459/5.18	3470/5723/1.90	868/1603/0.57	4053/7532/7.97
	MDYHS+1	1000/1000/0.50	33/33/0.051	8848/8868/11.38	2660/2693/1.19	794/809/0.38	2929/2950/4.10
	DYHS+	20160/38721/11.86	57/413/0.28	6420/15036/11.67	2837/5939/1.66	1089/2365/0.79	5248/11260/10.04
	DYHS	18349/35090/9.94	53/375/0.25	6067/14100/11.01	2857/5951/1.72	1165/2556/0.81	5641/12002/10.26
10^{-12}	MDYHS+	2931/5862/1.83	38/38/0.050	7543/12766/13.09	3681/6135/2.00	1276/2317/0.83	5300/9593/10.18
	MDYHS+1	1003/1003/0.46	38/38/0.052	11287/11307/14.57	3370/3403/1.50	1039/1054/0.52	3858/3879/5.41
	DYHS+	29056/55909/17.24	66/468/0.33	8118/19039/15.35	3651/7576/1.97	1461/3187/0.96	7351/15793/13.82
	DYHS	27355/52663/14.61	67/459/0.32	—	3666/7598/2.14	1506/3316/0.98	7403/15795/14.09

Table 4: Numerical results for test problems from the CUTEr library.

ϵ	Algorithm	#31 (1000)	#32 (1000)	#33 (1000)	#34 (500)	#35 (1024)	#36 (1024)
10^{-3}	MDYHS+	18/36/0.013	393/685/0.26	0/0/0	3062/5668/1.87	130/213/0.11	160/261/0.15
	MDYHS+1	15/22/0.010	893/950/0.49	0/0/0	2200/2218/1.04	91/91/0.07	118/118/0.09
	DYHS+	13/55/0.020	352/945/0.29	0/0/0	2931/5770/1.76	102/214/0.10	113/232/0.12
	DYHS	13/55/0.020	361/978/0.30	0/0/0	—	100/210/0.09	113/232/0.12
10^{-6}	MDYHS+	38/66/0.03	7842/14738/10.49	1001/2002/1.29	3139/5822/1.94	286/524/0.26	385/688/0.35
	MDYHS+1	23/30/0.015	—	942/942/0.83	2378/2396/1.14	264/264/0.18	228/228/0.16
	DYHS+	25/160/0.054	4248/11389/3.86	1074/1906/1.20	2978/5905/1.82	616/1290/0.58	287/605/0.27
	DYHS	—	—	907/1631/0.95	—	538/1151/0.50	295/620/0.28
10^{-9}	MDYHS+	58/106/0.047	—	3431/6862/4.30	3215/5974/1.99	434/820/0.43	490/898/0.46
	MDYHS+1	31/38/0.021	—	1498/1498/1.32	2543/2561/1.21	340/340/0.24	361/361/0.25
	DYHS+	76/1618/0.52	—	4634/19386/9.92	3009/6001/1.85	1087/2464/1.07	466/999/0.44
	DYHS	—	—	—	—	1293/2828/1.18	509/1055/0.46
10^{-12}	MDYHS+	70/127/0.057	—	5003/10006/6.30	3384/6265/2.07	498/948/0.48	577/1072/0.53
	MDYHS+1	40/47/0.026	—	2181/2181/1.92	2725/2743/1.31	515/1030/0.35	471/471/0.33
	DYHS+	181/4886/1.56	—	—	3072/6171/1.91	—	—
	DYHS	—	—	—	—	—	—
ϵ	Algorithm	#37 (1000)	#38 (5000)	#39 (500)	#40 (200)	#41 (300)	#42 (3549)
10^{-3}	MDYHS+	56/97/0.05	5789/9969/32.28	1751/3049/1.05	47/94/0.23	16/32/0.16	359/653/81.32
	MDYHS+1	34/59/0.030	8454/8547/38.40	1093/1141/0.51	6/6/0.026	4/4/0.03	336/337/46.12
	DYHS+	28/141/0.06	7487/13997/44.57	1332/2982/0.99	6/19/0.054	4/13/0.06	443/861/84.48
	DYHS	51/202/0.09	4817/12146/34.72	1287/2923/0.82	6/19/0.053	4/13/0.06	492/952/96.01
10^{-6}	MDYHS+	89/163/0.09	5793/9973/32.60	1770/3087/1.08	188/376/0.88	26/52/0.26	5884/11703/1203.4
	MDYHS+1	45/70/0.04	8458/8551/38.59	1113/1161/0.53	12/12/0.038	5/5/0.03	7022/7023/964.08
	DYHS+	—	7493/14019/44.41	1342/3015/0.93	12/39/0.079	5/16/0.07	8060/15523/1508.6
	DYHS	—	4822/12166/34.68	1297/2957/0.80	12/39/0.079	5/16/0.07	8684/16652/1662.2
10^{-9}	MDYHS+	141/263/0.14	5800/9987/32.43	1790/3127/1.09	527/1054/2.51	36/72/0.36	24741/49417/5064.2
	MDYHS+1	52/77/0.05	8458/8551/38.59	1124/1172/0.54	34/34/0.11	6/6/0.04	25795/25796/3494.0
	DYHS+	—	7493/14019/44.48	—	1037/2829/5.45	6/19/0.08	—
	DYHS	—	4824/12173/34.70	—	2256/6052/11.57	6/19/0.08	—
10^{-12}	MDYHS+	182/331/0.18	5810/10007/32.52	1873/3267/1.16	1057/2114/4.99	46/92/0.48	43624/87169/8874.9
	MDYHS+1	64/89/0.05	8459/8552/38.66	1135/1183/0.55	72/72/0.22	7/7/0.05	44983/44984/6104.9
	DYHS+	—	7494/14022/44.62	—	5969/16319/32.19	7/23/0.09	—
	DYHS	—	4824/12173/34.68	—	—	7/23/0.09	—
ϵ	Algorithm	#43 (5000)	#44 (150)	#45 (200)	#46 (529)	#47 (500)	#48 (1010)
10^{-3}	MDYHS+	110/207/0.35	38/75/1.05	210/374/0.12	217/386/0.44	7/14/0.004	508/860/1.74
	MDYHS+1	586/602/1.37	8/8/0.15	169/226/0.08	298/310/0.46	9/9/0.004	466/486/1.22
	DYHS+	19/66/0.08	8/26/0.32	87/272/0.08	311/658/0.66	10/24/0.007	361/1091/1.69
	DYHS	219/599/0.78	8/26/0.31	177/533/0.14	295/628/0.62	10/24/0.007	481/1370/2.15
10^{-6}	MDYHS+	127/241/0.41	44/86/1.10	310/569/0.18	4916/9716/10.94	468/936/0.28	1331/2492/4.76
	MDYHS+1	859/875/1.99	10/10/0.17	261/318/0.12	2367/2379/3.64	516/516/0.22	556/576/1.50
	DYHS+	21/72/0.10	11/36/0.41	156/476/0.13	7103/13549/13.28	495/924/0.27	—
	DYHS	246/679/0.91	11/36/0.41	326/977/0.25	7129/13668/13.33	350/673/0.21	—
10^{-9}	MDYHS+	136/257/0.43	—	406/761/0.25	6880/13644/15.12	21368/42736/12.82	1453/2736/5.31
	MDYHS+1	1147/1163/2.79	—	362/419/0.17	3956/3968/6.02	18659/18659/8.30	646/666/1.67
	DYHS+	22/75/0.10	—	189/579/0.16	13395/25751/25.48	—	—
	DYHS	262/730/0.96	—	498/1497/0.42	16291/31452/32.49	—	—
10^{-12}	MDYHS+	141/263/0.45	—	593/1030/0.34	8837/17436/19.87	46983/93966/29.49	1773/3332/6.32
	MDYHS+1	1294/1310/3.06	—	491/548/0.23	5994/6006/9.58	42158/42158/19.46	859/879/2.23
	DYHS+	—	—	330/999/0.28	21089/40486/39.78	—	—
	DYHS	268/749/0.10	—	644/1930/0.53	26127/50417/49.36	—	—

Table 5: Numerical results for test problems from the CUTEr library.

ϵ	Algorithm	#49 (500)	#49 (2000)	#50 (1000)	#51 (100)	#52 (10000)	#53 (100)
10^{-3}	MDYHS+	79/134/0.14	74/129/0.48	1877/3565/1.93	237/394/0.10	23/44/0.10	94/129/0.03
	MDYHS+1	68/71/0.11	73/78/0.36	884/884/0.65	106/106/0.04	3/9/0.02	156/169/0.06
	DYHS+	58/151/0.13	60/167/0.48	3154/6174/3.13	276/614/0.13	2/6/0.012	103/291/0.06
	DYHS	65/173/0.14	48/143/0.41	1334/2566/1.31	246/545/0.12	2/6/0.012	94/256/0.05
10^{-6}	MDYHS+	3234/6363/6.58	3110/6127/21.04	2844/5499/2.98	347/614/0.16	44/85/0.20	2168/3506/0.87
	MDYHS+1	3169/3172/4.46	5088/5093/24.20	1887/1887/1.43	208/208/0.07	9/27/0.05	11836/11849/4.06
	DYHS+	—	3885/35556/80.37	9989/19969/10.11	526/1135/0.26	6/19/0.042	1242/3710/0.71
	DYHS	—	—	9642/18822/9.61	462/1012/0.25	10/32/0.062	1410/4146/0.82
10^{-9}	MDYHS+	4046/7987/8.24	4207/8321/28.92	3751/7313/4.08	462/844/0.21	51/97/0.23	27768/50730/12.06
	MDYHS+1	5957/5960/8.40	10111/10116/48.22	2824/2824/2.14	282/282/0.10	21/48/0.10	—
	DYHS+	—	—	—	—	8/25/0.05	5250/15889/3.02
	DYHS	—	—	—	—	11/35/0.07	20679/62630/12.10
10^{-12}	MDYHS+	5708/11097/11.51	5876/11485/40.39	5034/9605/5.23	546/1010/0.26	—	—
	MDYHS+1	8594/8597/12.12	15076/15081/71.84	3598/3598/2.78	357/357/0.13	—	—
	DYHS+	—	—	—	—	—	12363/37959/7.59
	DYHS	—	—	—	—	—	—
ϵ	Algorithm	#54 (10000)	#55 (500)	#56 (100)	#57 (5000)	#58 (100)	#59 (1000)
10^{-3}	MDYHS+	32/52/0.11	22/22/0.009	40/52/0.026	90/152/0.23	39/45/0.021	21/21/0.010
	MDYHS+1	24/33/0.08	22/22/0.009	42/49/0.021	205/217/0.31	41/46/0.025	21/25/0.009
	DYHS+	20/75/0.12	2/60/0.012	39/165/0.062	54/168/0.21	34/165/0.051	29/220/0.051
	DYHS	22/78/0.13	3/59/0.012	37/165/0.041	54/168/0.21	30/153/0.042	24/161/0.048
10^{-6}	MDYHS+	52/92/0.20	62/97/0.029	109/165/0.059	631/1197/1.37	59/65/0.022	26/26/0.011
	MDYHS+1	36/45/0.11	62/100/0.030	80/87/0.037	3455/3467/5.14	60/65/0.020	26/30/0.011
	DYHS+	32/110/0.18	18/175/0.039	115/1226/0.31	179/548/0.47	47/218/0.042	37/296/0.064
	DYHS	34/111/0.18	30/426/0.092	—	239/707/0.56	46/217/0.034	32/222/0.047
10^{-9}	MDYHS+	72/132/0.28	87/170/0.048	156/259/0.095	1479/2854/2.92	79/85/0.027	30/30/0.013
	MDYHS+1	48/57/0.15	185/412/0.079	120/127/0.055	—	79/84/0.028	30/34/0.013
	DYHS+	44/150/0.25	22/187/0.042	—	210/655/0.56	56/272/0.046	43/327/0.071
	DYHS	47/153/0.26	38/451/0.099	—	405/1192/0.95	60/263/0.041	38/245/0.055
10^{-12}	MDYHS+	83/150/0.33	107/210/0.069	209/363/0.13	2541/4941/5.05	99/105/0.033	35/35/0.015
	MDYHS+1	61/70/0.19	315/357/0.13	160/167/0.07	—	99/104/0.031	35/39/0.015
	DYHS+	55/187/0.30	23/191/0.042	—	227/720/0.62	70/334/0.053	46/345/0.074
	DYHS	58/188/0.31	49/556/0.121	—	571/1672/1.34	78/325/0.054	45/303/0.064
ϵ	Algorithm	#59 (10000)	#60 (1000)	#60 (5000)	#61 (100)	#62 (5000)	#62 (10000)
10^{-3}	MDYHS+	25/25/0.060	34/59/0.059	31/60/0.27	41/77/0.53	14/14/0.051	15/15/0.106
	MDYHS+1	25/25/0.060	22/23/0.030	23/27/0.15	50/64/0.52	14/26/0.071	15/26/0.141
	DYHS+	56/317/0.34	28/70/0.061	27/73/0.29	21/73/0.45	25/139/0.24	39/222/0.716
	DYHS	51/343/0.35	28/71/0.062	26/68/0.27	21/73/0.45	28/161/0.27	33/201/0.627
10^{-6}	MDYHS+	29/29/0.065	58/107/0.10	55/108/0.49	53/101/0.65	19/19/0.068	20/20/0.139
	MDYHS+1	29/29/0.065	40/41/0.06	40/44/0.27	54/68/0.55	19/31/0.084	19/30/0.168
	DYHS+	68/377/0.42	56/144/0.13	67/162/0.64	26/90/0.55	36/203/0.346	53/311/1.005
	DYHS	62/406/0.41	52/180/0.14	51/133/0.51	26/90/0.55	34/199/0.341	41/242/0.758
10^{-9}	MDYHS+	34/34/0.070	80/151/0.15	77/152/0.68	63/119/0.83	24/24/0.085	24/24/0.175
	MDYHS+1	34/34/0.070	57/58/0.07	58/62/0.36	58/72/0.61	23/35/0.103	24/35/0.210
	DYHS+	77/441/0.46	—	—	87/647/3.52	45/252/0.43	67/380/1.212
	DYHS	70/462/0.46	7087/16410/14.10	15757/35456/138.93	—	46/278/0.448	52/304/0.962
10^{-12}	MDYHS+	39/39/0.09	104/197/0.18	106/202/0.92	—	28/28/0.103	29/29/0.203
	MDYHS+1	39/39/0.09	74/75/0.10	74/78/0.46	—	28/40/0.12	29/40/0.241
	DYHS+	83/477/0.52	—	—	—	56/321/0.536	80/462/1.479
	DYHS	82/535/0.53	—	—	—	51/310/0.506	61/356/1.123

Table 6: Numerical results for test problems from the CUTEr library.

ϵ	Algorithm	#63 (10000)	#64 (3000)	#64 (5000)	#65 (1000)	#66 (1000)	#67 (10000)	#68 (500)	#69 (4000)
10^{-3}	MDYHS+	22/40/0.070	1468/2936/2.026	1700/3400/2.85	3/5/0.003	6143/12287/4.78	876/1752/2.74	24/38/0.022	32/53/0.063
	MDYHS+1	10/29/0.049	798/798/0.815	955/955/1.27	3/3/0.003	20/92/0.022	910/910/2.07	19/20/0.009	169/283/0.35
	DYHS+	7/25/0.036	3973/8053/5.04	5584/10957/8.96	3/9/0.005	89/310/0.096	2651/5026/7.46	20/72/0.037	47/150/0.16
	DYHS	17/55/0.077	4628/9417/5.25	4289/8500/6.05	3/9/0.005	10/74/0.022	2741/5183/6.96	21/75/0.044	42/138/0.12
10^{-6}	MDYHS+	43/82/0.156	2228/4456/3.04	2653/5306/4.54	13/25/0.015	12974/25949/9.39	1093/2186/3.44	44/78/0.032	309/530/0.68
	MDYHS+1	17/36/0.061	1220/1220/1.31	1567/1567/2.05	6/6/0.005	24/96/0.024	1114/1114/2.53	29/30/0.017	177/291/0.36
	DYHS+	10/34/0.052	7159/14493/8.55	8996/17733/14.42	6/19/0.010	177/579/0.176	4414/8363/12.53	32/112/0.040	51/163/0.15
	DYHS	44/140/0.191	7589/15343/9.41	7553/14971/10.78	6/19/0.010	28/130/0.038	4749/9120/12.37	33/115/0.036	68/217/0.21
10^{-9}	MDYHS+	63/122/0.212	3135/6270/4.33	3693/7386/6.45	23/45/0.027	18572/37138/13.56	1356/2712/4.21	64/118/0.047	355/622/0.80
	MDYHS+1	26/45/0.083	1682/1682/1.73	2057/2057/2.64	9/9/0.007	27/99/0.025	1287/1287/2.88	39/40/0.023	187/301/0.39
	DYHS+	14/47/0.067	10438/21030/13.69	13050/25776/20.78	9/29/0.015	222/723/0.237	6796/13011/19.32	44/152/0.052	56/179/0.17
	DYHS	82/255/0.352	9905/20054/11.25	11373/22610/16.98	9/29/0.015	43/176/0.052	6714/12900/17.28	45/155/0.053	110/336/0.31
10^{-12}	MDYHS+	69/132/0.237	4054/8108/5.61	4642/9284/7.80	33/65/0.039	19491/38762/14.058	1840/3677/5.64	84/158/0.075	407/706/0.88
	MDYHS+1	34/53/0.102	2077/2077/2.35	2636/2636/3.38	12/12/0.010	27/99/0.025	1439/1439/3.25	48/49/0.027	195/309/0.38
	DYHS+	17/57/0.078	12221/24617/15.98	17027/33541/27.03	463/1103/0.595	233/759/0.234	8793/16906/24.85	56/192/0.064	61/195/0.19
	DYHS	111/345/0.466	12135/24540/13.86	15923/31589/22.33	133/324/0.166	57/221/0.064	8612/16567/22.24	57/195/0.062	129/394/0.37

need the information of the original function of (1), we selected a collection of test problems from the CUTEr library and listed them in Table 1. The first column "Prob." denotes the problem number, and the columns "Name" and "n" denote the name and the dimension of the problem, respectively. Since we were interested in large-scale problems, we only considered problems with size at least 100. The largest dimension was set to 10,000. Moreover, we accessed CUTEr functions from within Matlab R2012a by using Matlab interface.

Our numerical results were reported in Tables 2, 3, 4, 5, and 6 in the form of $k/Innerk/CPU$, where k, $Innerk$, and CPU stand for the number of iterations, the total trial times of the line search and the CPU time elapsed, respectively. For the DYHS+ and the DYHS, we let N_f and N_g be the number of function evaluations and the number of gradient evaluations, respectively, and set $Innerk = N_f/3 + N_g$ by automatic differentiation (see [10, 20] for details). Moreover, "—" means the method's failure to achieve a prescribed accuracy when the number of iterations exceeded 50,000, and the test problems are represented in the form of #Pro.(n).

The performances of the four methods, relative to CPU time, were evaluated using the profiles of Dolan and Morè [21]. That is, for the four methods, we plotted the fraction P of problems for which each of the methods was within a factor τ of the best time. Figures 1, 2, and 3 showed the performance profiles referring to CPU time, the number of iterations and the total trial times of the line search, respectively. These figures revealed that the MDYHS+ method and the MDYHS + 1 method performed better than the DYHS method and the DYHS+ method. The performance profiles also showed that the MDYHS+ method and the MDYHS + 1 method were comparable and solved almost all of the test problems up to $\epsilon = 10^{-9}$. Yet, the latter has no convergence.

4.2. Tested by Some Boundary Value Problems. In this section, we implemented the MDYHS+ method and the MDYHS + 1 method to solve some boundary value problems. See [22, Chapter 1] for the background of the boundary value problems.

In order to confirm the efficiency of the MDYHS+ method and the MDYHS + 1 method to solve this class of problems, We drew a set of 11 boundary value problems from [19] and listed them in Table 7, where the test problems were expressed by #Pro.(n) (#Pro. denotes the problem number in [19] and n denotes the dimension), and the test results were listed in the form of $k/Innerk/CPU$.

From Table 7, we can see that both of the MDYHS+ method and the MDYHS + 1 method are efficient in solving boundary value problems. The MDYHS + 1 method seems a little better but has no convergence.

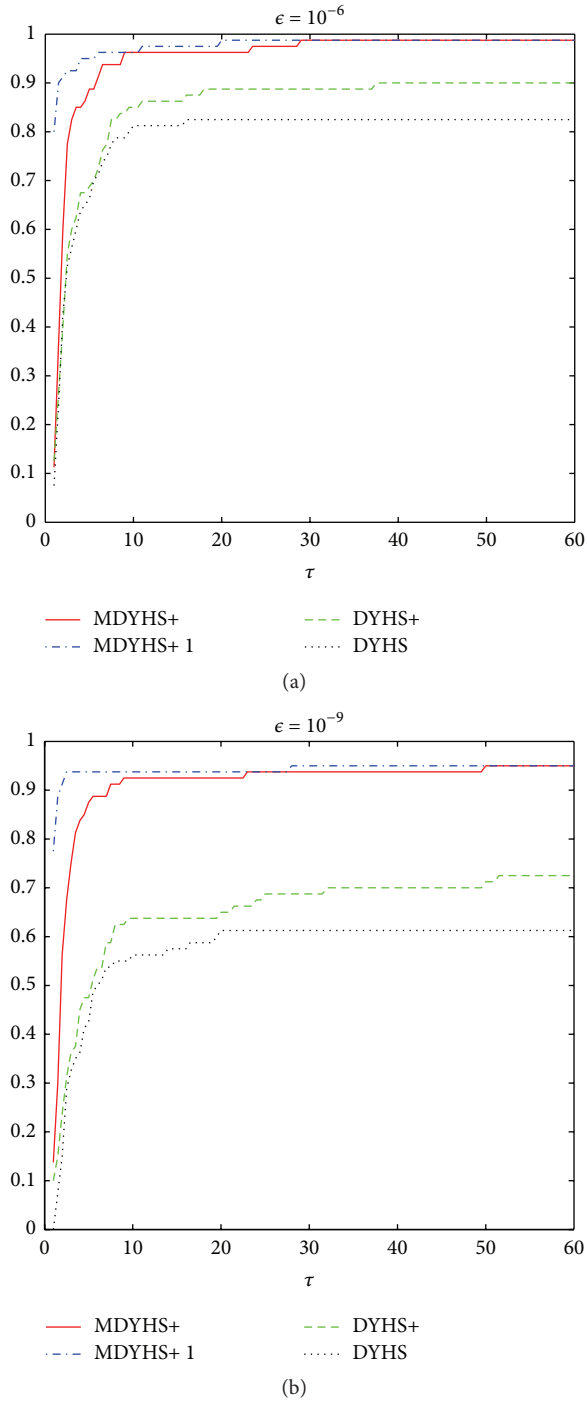


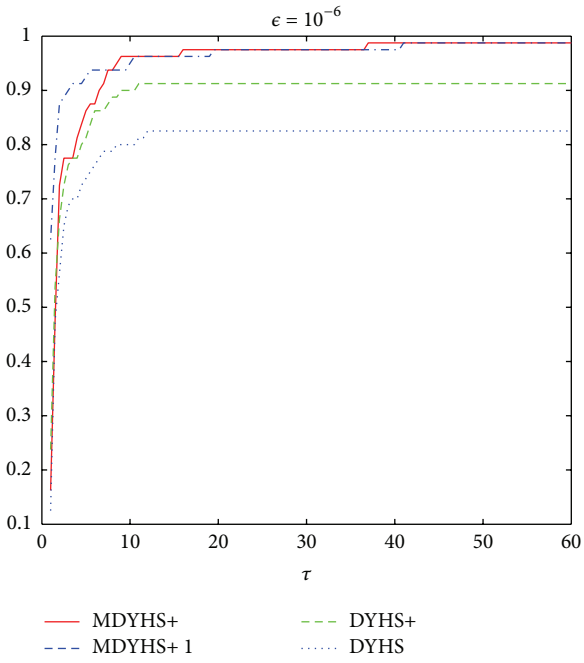
FIGURE 1: Performance profiles based on CPU time.

5. Conclusions

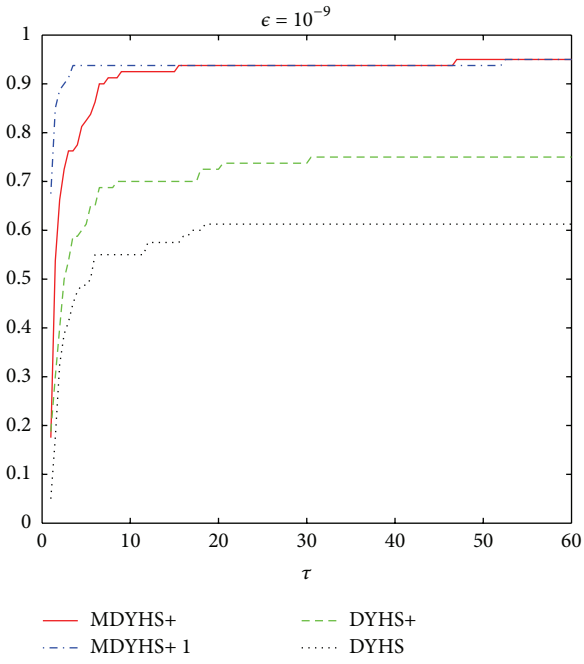
This paper has studied two modified versions of a Dai-Yuan hybrid conjugate gradient method with two different line searches only using gradient information and has proven that with the line search (9), it is convergent in the sense of $\liminf_{k\to\infty}\|g_k\| = 0$. Then, we investigated the numerical behaviors of the two modified versions over two classes of standard test problems. From the numerical results, we can conclude that the two modified hybrid conjugate gradient methods are more efficient (especially in high precision) in solving large-scale nonlinear unconstrained minimization problems and have broader application scope. For example,

Table 7: Numerical results for some boundary value problems.

ϵ	Algorithm	#2 (10000)	#8 (10000)	#12 (10000)	#13 (5000)	#14 (10000)	#24 (1000)	#25 (10000)	#26 (1000)	#27 (10000)	#29 (1000)	#31 (1000)
10^{-3}	MDYHS+	2/3/0.004	9/13/0.07	9/19/0.02	369/738/0.66	8/10/0.06	5/9/0.38	16/27/0.07	367/734/0.16	180/360/0.58	1007/2014/0.72	2/2/0.003
	MDYHS+1	2/4/0.004	7/16/0.08	3/6/0.01	1043/1043/1.30	7/7/0.04	4/8/0.32	12/15/0.04	983/983/0.30	500/500/1.05	1000/1000/0.50	1/2/0.006
10^{-6}	MDYHS+	11/21/0.02	26/47/0.21	19/39/0.04	3012/6024/5.06	19/31/0.14	15/29/1.16	25/45/0.08	1052/2104/0.46	10013/20026/28.03	1181/2362/0.83	9/9/0.004
	MDYHS+1	3/6/0.005	15/32/0.10	4/8/0.01	5000/5000/5.61	11/11/0.06	6/12/0.42	21/24/0.05	1000/1000/0.31	10000/10000/19.14	1000/1000/0.49	8/9/0.007
10^{-9}	MDYHS+	21/41/0.03	46/87/0.40	29/59/0.05	7267/14537/11.92	29/52/0.21	25/49/1.95	35/65/0.12	1653/3306/0.74	19831/39662/54.61	1995/3990/1.41	2413/2413/1.06
	MDYHS+1	3/6/0.005	23/48/0.14	5/10/0.01	5002/5002/5.70	16/16/0.08	9/18/0.68	31/34/0.08	1000/1000/0.31	20003/20003/38.82	1000/1000/0.49	2413/2414/1.06
10^{-12}	MDYHS+	31/61/0.05	56/105/0.45	39/78/0.06	9972/19944/16.08	39/72/0.30	35/69/2.66	45/85/0.16	2236/4472/0.98	30046/60092/92.66	2344/4680/1.65	—
	MDYHS+1	4/8/0.006	31/64/0.19	5/10/0.01	5002/5002/5.93	21/21/0.12	12/24/0.83	42/45/0.09	1000/1000/0.31	29989/29989/58.47	1958/1985/0.96	—

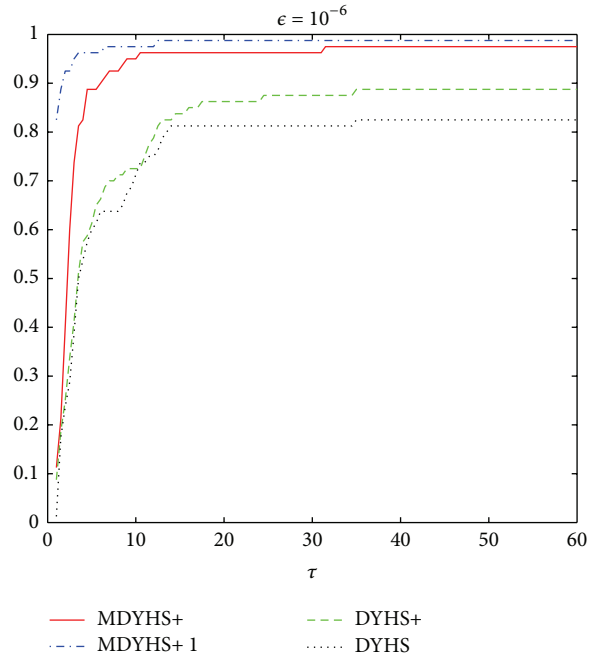


(a)

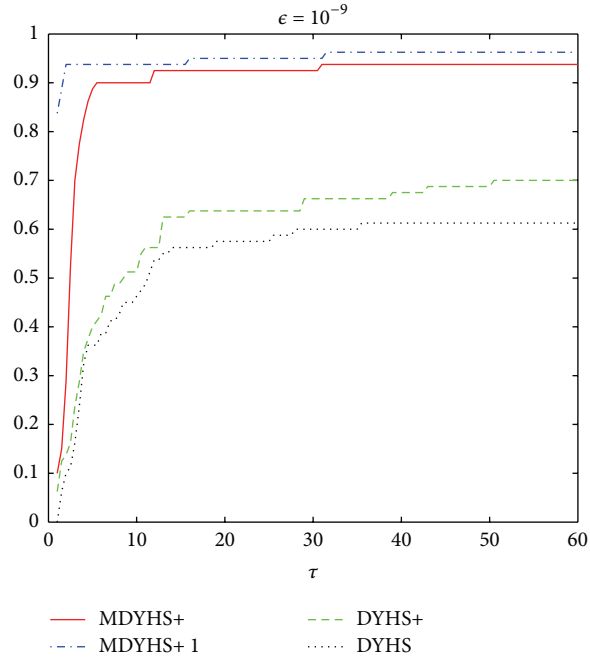


(b)

Figure 2: Performance profiles based on the number of iterations.



(a)



(b)

Figure 3: Performance profiles based on the number of inner iterations.

they can be used to solve some boundary value problems, where functions are not explicit.

Acknowledgments

The authors are very grateful to the associate editor and the referees for their valuable suggestions. Meanwhile, the first author is also very grateful to Yunda Dong for suggesting her to write this paper and to add Section 4.2 to the revised version. This work was supported by National Science Foundation of China, no. 60974082.

References

[1] 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, 1999.

[2] 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, pp. 409–436, 1952.

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

[4] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *Revue francaise d'informatique et derecherche opérationnelle, série rouge*, vol. 3, no. 16, pp. 35–43, 1969.

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

[6] 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, 2005.

[7] 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.

[8] W. W. Hager and H. Zhang, "Algorithm 851: *CG – DESCENT*, a conjugate gradient method with guaranteed descent," *Association for Computing Machinery*, vol. 32, no. 1, pp. 113–137, 2006.

[9] Y. H. Dai and Y. Yuan, "An efficient hybrid conjugate gradient method for unconstrained optimization," *Annals of Operations Research*, vol. 103, pp. 33–47, 2001.

[10] 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.

[11] R. Pytlak, *Conjugate Gradient Algorithms in Nonconvex Optimization*, vol. 89 of *Nonconvex Optimization and its Applications*, Springer, Berlin, Germany, 2009.

[12] Y.-H. Dai and C.-X. Kou, "A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 296–320, 2013.

[13] Y. Dong, "A practical PR+ conjugate gradient method only using gradient," *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 2041–2052, 2012.

[14] Y. Dong, "New step lengths in conjugate gradient methods," *Computers & Mathematics with Applications*, vol. 60, no. 3, pp. 563–571, 2010.

[15] A. S. Lewis and M. L. Overton, "Nonsmooth optimization via quasi-Newton methods," *Mathematical Programming A*, 2012.

[16] C. Lemarechal, "A view of line search," *Optimization and Optimal Control*, vol. 30, pp. 59–78, 1981, Lecture Notes on Control and Information Sciences.

[17] 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.

[18] 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, pp. 373–394, 2003.

[19] E. Spedicato and Z. Huang, "Numerical experience with Newton-like methods for nonlinear algebraic systems," *Computing*, vol. 58, no. 1, pp. 69–89, 1997.

[20] A. Griewank, "On automatic differentiation," in *Mathematical Programming: Recent Developments and Applications*, M. Iri and K. Tanabe, Eds., pp. 83–108, Kluwer Academic, 1989.

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

[22] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, USA, 1970.