Journal of Computational and Applied Mathematics 371 (2020) 112708

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

An efficient Dai–Liao type conjugate gradient method by reformulating the CG parameter in the search direction equation

Mina Lotfi, S.Mohammad Hosseini *

Department of Applied Mathematics, Faculty of Mathematical Sciences, Tarbiat Modares University, P.O. Box 14115-175, Tehran, Iran

ARTICLE INFO

Article history:
Received 6 March 2019
Received in revised form 30 November 2019

Keywords:
Unconstrained optimization
Conjugate gradient method
Secant equation
Modified BFGS method
Global convergence

ABSTRACT

Based on the modified BFGS method proposed by Li and Fukushima (2001), we present a new value of the parameter t in Dai–Liao Conjugate Gradient Method. Under mild assumptions, we establish the global convergence property of the proposed method. Numerical results on some test problems in the CUTEst library illustrate computational efficiency of the new method.

1. Introduction

We consider the following unconstrained optimization problem

$$\min f(x), \qquad x \in \mathbb{R}^n, \tag{1.1}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuously differentiable and its gradient is denoted by $g(x) = \nabla f(x)$. Conjugate gradient (CG) methods are one of the most popular methods for solving (1.1), some important properties of these methods are their low memory requirement and strong global convergence properties, which made them useful tools in solving large-scale optimization problems. A CG method, starting from an initial point $x_0 \in R^n$, generates a sequence of points x_k, obtained by

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where x_k is the current iteration point, $\alpha_k > 0$ is the stepsize that is usually determined to fulfill the standard Wolfe line search conditions

$$f(x_k + \alpha_k d_k) - f(x_k) \leq \delta \alpha_k g_k^T d_k, \tag{1.3}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k, \tag{1.4}$$

or stronger version of the Wolfe line search conditions, given by (1.3) and

$$|g(x_k + \alpha_k d_k)^T d_k| \leq -\sigma |g_k^T d_k|,$$

* Corresponding author.
E-mail addresses: minalotfi@modares.ac.ir (M. Lotfi), hossei_m@modares.ac.ir (S.M. Hosseini).

https://doi.org/10.1016/j.cam.2019.112708
0377-0427/

where $0 < \delta < \sigma < 1$, $g_k = \nabla f(x_k)$, and the search direction d_{k+1}, for $k \geq 0$, is computed by

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \qquad d_0 = -g_0, \tag{1.5}$$

where the coefficient β_k is a scalar, distinguishing the CG method. The most well-known CG methods include Fletcher–Reeves (FR) method [1], the Dai–Yuan (DY) method [2], the Hestenes–Stiefel (HS) method [3], Liu–Storey (LS) method [4] and the Polak–Ribiére–Polyak (PRP) method [5,6], see also [7] for more details. In conjugate gradient method, the search direction d_k is determined in such a way that the following conjugacy condition holds

$$d_i^T G d_j = 0, \qquad\qquad i \neq j, \tag{1.6}$$

where G is the Hessian of the objective function. On the other hand, according to the mean value theorem, there exists some $\lambda \in (0, 1)$ such that

$$d_{k+1}^T y_k = \alpha_k d_{k+1}^T \nabla^2 f(x_k + \lambda \alpha_k d_k) d_k, \tag{1.7}$$

where $y_k = g_{k+1} - g_k$. Now, by combining (1.6) and (1.7) the following conjugate condition can be deduced

$$d_{k+1}^T y_k = 0.$$

Dai and Liao (DL) [8] with modification of conjugate condition, presented a family of CG methods and established their convergence for convex functions, in the DL method the CG coefficient is computed by

$$\beta_k^{DL} = \frac{g_{k+1}^T y_k}{d_k^T y_k} - t \frac{g_{k+1}^T s_k}{d_k^T y_k}, \tag{1.8}$$

where $t > 0$ and $s_k = x_{k+1} - x_k$. In order to establish the global convergence of DL method for general functions, β_k^{DL} is updated as below

$$\beta_k^{DL+} = max\{\frac{g_{k+1}^T y_k}{d_k^T y_k}, 0\} - t \frac{g_{k+1}^T s_k}{d_k^T y_k}. \tag{1.9}$$

In DL method, numerical performance is very much dependent on the parameter t [9], and the search directions do not necessarily satisfy the descent condition. In the family of related CG methods, Hager and Zhang proposed [10] a conjugate gradient method, named CG-DESCENT, which is a developed form of that of Hager and Zhang [7], with the following CG coefficient

$$\beta_k^{HZ} = \frac{g_{k+1}^T y_k}{d_k^T y_k} - 2 \frac{\|y_k\|^2 g_{k+1}^T d_k}{(d_k^T y_k)^2},$$

where $\|.\|$ stands for Euclidean norm. They showed that the search direction of their method satisfies the sufficient descent condition $g_{k+1}^T d_{k+1}^{HZ} \leq -\frac{7}{8}\|g_{k+1}\|^2$. It is easy to see that β_k^{HZ} is a version of β_k^{DL} with $t = 2\frac{\|y_k\|^2}{d_k^T y_k}$. By seeking the conjugate gradient direction nearest to the direction of the scaled memoryless BFGS method [11], Dai and Kou (DK) [12] proposed the following CG coefficient

$$\beta_k^{DK} = \frac{g_{k+1}^T y_k}{d_k^T y_k} - \left(\tau_k + \frac{\|y_k\|^2}{s_k^T y_k} - \frac{s_k^T y_k}{\|s_k\|^2}\right) \frac{g_{k+1}^T s_k}{d_k^T y_k},$$

where τ_k is corresponding to the scaling parameter in the self-scaling memoryless BFGS method. In [12], it was shown that the DK method, as a member of Dai–Liao family with $t = \tau_k + \frac{\|y_k\|^2}{s_k^T y_k} - \frac{s_k^T y_k}{\|s_k\|^2}$, outperforms many existing conjugate gradient methods. In recent years, much efforts has been made to find the proper choice for the nonnegative parameter t in (1.8), see [13–17]. Based on a singular value study on the DL method, Babaie-Kafaki and Ghanbari [18] proposed the following adaptive choices for t

$$t_1 = \frac{s_k^T y_k}{\|s_k\|^2} + \frac{\|y_k\|}{\|s_k\|}, \qquad\qquad t_2 = \frac{\|y_k\|}{\|s_k\|}.$$

In [18], it was shown that the DL method with $t = t_1$ or $t = t_2$ outperforms the methods proposed by Hager and Zhang [10], and Dai and Kou [12]. Recently, by clustering the eigenvalues of the matrix which determines the search direction of the DL method, Andrei [19] suggested the following value for t

$$t_3 = \frac{s_k^T y_k}{\|s_k\|^2},$$

which becomes a variant of the DL method that is more efficient and more robust than some other variants of the same method based on minimizing the condition number of the matrix associated to the search direction. In this paper, based on the above discussions, motivated by the strong theoretical properties and computational efficiency of the modified BFGS method proposed by Li and Fukushima [20], and also taking benefit of properties of the gradient and Hessian of the objective functions, we propose a new value for the parameter t in (1.8). The remainder of this work is organized as follows. In Section 2, we present our new method for the selection of the parameter t. In Section 3, we prove the global convergence of our method. In Section 4, we tested our method on some unconstrained optimization problems from CUTEst collection and compared with those of some variants of DL methods. Numerical results show that the proposed modified method is practically efficient.

2. New version of CG method

As we know, in a small neighborhood of the current point x_{k+1}, close enough to a local minimizer, the nonlinear objective function behaves as a quadratic one. Therefore, if the point x_{k+1} is close enough to a local minimizer, then a good direction to follow is the Newton direction $-\bigtriangledown^2 f(x_{k+1})^{-1}g_{k+1}$, that is,

$$d_{k+1} = -\bigtriangledown^2 f(x_{k+1})^{-1}g_{k+1}.$$

Hence, equivalently, from (1.5) and the formula for β_k in (1.8), namely β_k^{DL}, it is reasonable to compute the parameter t in such a manner that the following equation is satisfied:

$$-\bigtriangledown^2 f(x_{k+1})^{-1}g_{k+1} = -g_{k+1} + \frac{g_{k+1}^T y_k}{d_k^T y_k} d_k - t\frac{g_{k+1}^T s_k}{d_k^T y_k} d_k.$$

After some algebra, we then obtain

$$t = \frac{s_k^T g_{k+1} - s_k^T \bigtriangledown^2 f(x_{k+1}) g_{k+1} + \frac{g_{k+1}^T y_k}{s_k^T y_k} s_k^T \bigtriangledown^2 f(x_{k+1}) s_k}{\frac{g_{k+1}^T s_k}{s_k^T y_k} s_k^T \bigtriangledown^2 f(x_{k+1}) s_k}. \tag{2.1}$$

Obviously, in order to avoid the exact computation of the Hessian matrix $\bigtriangledown^2 f(x_{k+1})$, we use a Quasi-Newton method. In the Quasi-Newton(QN) methods [21,22], the search direction d_{k+1} is computed by solving system of linear equations, $B_{k+1}d_{k+1} = -g_{k+1}$, where the matrix B_{k+1} is updated from an approximation matrix B_k of the Hessian $\bigtriangledown^2 f(x_k)$, such that the matrix B_{k+1} satisfies some version of the Quasi-Newton equation, or secant equation, say,

$$B_{k+1}s_k = y_k.$$

Theorem 2.1. *If f is sufficiently smooth and $\|s_k\|$ is sufficiently small, then the following estimate holds:*

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

where,

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

Proof. Using Taylor's expansion, we have

$$f(x_k) = f(x_{k+1}) - g_{k+1}^T s_k + \frac{1}{2}s_k^T \bigtriangledown^2 f(x_{k+1})s_k - \frac{1}{6}s_k^T(T_{k+1}s_k)s_k + O(\|s_k\|^4),$$

and

$$g_k^T s_k = g_{k+1}^T s_k - s_k^T \bigtriangledown^2 f(x_{k+1})s_k + \frac{1}{2}s_k^T(T_{k+1}s_k)s_k + O(\|s_k\|^4),$$

which completes the proof. □

Among several kinds of Quasi-Newton methods [22], the so-called BFGS method is regarded as the most effective one in which the Hessian is updated by

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}.$$

In spite of promising numerical results and global convergence of BFGS method for convex problems, it may fail to converge for nonconvex problems. So, in an effort to overcome its convergence failure for more general objective function, Li and Fukushima [20] proposed a modified version of BFGS method and established its global convergence and superlinear convergence rate even without convexity assumption on the objective function. In their method the matrix B_{k+1} is obtained from the following modified secant equation

$$B_{k+1}s_k = z_k, \qquad z_k = y_k + h_k\|g_k\|^r s_k, \tag{2.2}$$

Table 1
The test problems and their dimensions.

No	Name	Dim	No	Name	Dim	No	Name	Dim
1	ARGLINA	200	2	ARGLINA	100	3	BOX	100
4	BDEXP	1000	5	BDEXP	5000	6	BRYBND	1000
7	BRYBND	5000	8	CHAINWOO	100	9	CHAINWOO	1000
10	CHNROSNB	50	11	COSINE	100	12	COSINE	1000
13	DIXMAANA	3000	14	DIXMAANA	9000	15	DIXMAANB	3000
16	DIXMAANB	9000	17	DIXMAANC	3000	18	DIXMAANC	9000
19	DIXMAAND	3000	20	DIXMAAND	9000	21	DIXMAANE	3000
22	DIXMAANE	9000	23	DIXMAANF	3000	24	DIXMAANF	9000
25	DIXMAANG	3000	26	DIXMAANG	9000	27	DIXMAANH	3000
28	DIXMAANH	9000	29	DIXMAANI	3000	30	DIXMAANI	9000
31	DIXMAANJ	3000	32	DIXMAANJ	9000	33	DIXMAANK	3000
34	DIXMAANK	9000	35	DIXMAANL	3000	36	DIXMAANL	9000
37	DIXMAANM	3000	38	DIXMAANM	9000	39	DIXMAANN	3000
40	DIXMAANN	9000	41	DIXMAANO	3000	42	DIXMAANO	9000
43	DIXMAANP	3000	44	DIXMAANP	9000	45	DIXON3DQ	100
46	DIXON3DQ	1000	47	EG2	1000	48	EIGENALS	110
49	EIGENBLS	110	50	FLETCHCR	100	51	FLETCHCR	1000
52	FMINSRF2	5625	53	FMINSRF2	10000	54	FMINSURF	5625
55	FMINSURF	10000	56	GENROSE	100	57	GENROSE	500
58	GENHUMPS	1000	59	LIARWHD	5000	60	LIARWHD	10000
61	LMINSURF	5625	62	LMINSURF	10000	63	MANCINO	50
64	MANCINO	100	65	MOREBV	1000	66	MOREBV	5000
67	MSQRTALS	1024	68	MSQRTBLS	1024	69	NLMSURF	10000
70	NLMSURF	5625	71	NONSCOMP	5000	72	NONDQUAR	1000
73	NONDQUAR	5000	74	PENALTY1	100	75	POWELLSG	5000
76	POWELLSG	10000	77	SPMSRTLS	1000	78	SPMSRTLS	4999
79	SROSENBR	1000	80	SROSENBR	5000	81	TESTQUAD	1000
82	TESTQUAD	5000	83	TOINTGSS	5000	84	TOINTGSS	10000
85	TRIDIA	1000	86	TRIDIA	5000	87	VAREIGVL	100
88	VAREIGVL	500	89	WOODS	4000	90	WOODS	10000

where h_k is given by

$$h_k = C + max\left\{-\frac{s_k^T y_k}{\|s_k\|^2}, 0\right\}\|g_k\|^{-r},$$

where C, r are two positive constants. A remarkable feature of this method is that: if the line search procedure guarantees $s_k^T y_k \geq 0$, for all $k \geq 0$, then, we have

$$s_k^T z_k = s_k^T y_k + C\|g_k\|^r\|s_k\|^2 + max\left\{-\frac{s_k^T y_k}{\|s_k\|^2}, 0\right\}\|s_k\|^2 > C\|g_k\|^{-r}\|s_k\|^2 > 0$$

which ensuring the matrix B_{k+1} to inherit the positive definiteness of B_k.

So, taking advantage of the strong theoretical and numerical properties of the modified BFGS method, we use secant equation (2.2) to simplify (2.1) and propose the following formula for parameter t

$$t_4 = \frac{(1 - h_k\|g_k\|^r)s_k^T g_{k+1} + \frac{g_{k+1}^T y_k}{y_k^T s_k} h_k\|g_k\|^r\|s_k\|^2}{g_{k+1}^T s_k + \frac{g_{k+1}^T s_k}{s_k^T y_k} h_k\|g_k\|^r\|s_k\|^2}. \tag{2.3}$$

Then to ensure the new search direction satisfies the descent condition, and the proposed value of parameter t, denoted by t_4, is nonnegative similar to that of [23] we adjust it as follows:

$$t_4^* = max\left\{t_4, \vartheta\frac{\|y_k\|^2}{s_k^T y_k}\right\}.$$

3. Convergence analysis

Analysis of convergence is covered in Theorem 3.1 but we first need to recall some preliminary assumptions and results.

Assumption 3.1. The level set $\mathcal{L} = \{x \in \mathbb{R}^n | f(x) \leq f(x_0)\}$ is bounded, where x_0 is an initial point.

Table 2
The numerical results.

Problem	Dim	$DL1$	$DL2$	$DL3$	MDL	$HZ+$	$DK+$
		$n_i/n_f/n_g$	$n_i/n_f/n_g$	$n_i/n_f/n_g$	$n_i/n_f/n_g$	$n_i/n_f/n_g$	$n_i/n_f/n_g$
ARGLINA	100	2/3/2	2/3/2	2/3/2	2/3/2	2/3/2	2/3/2
ARGLINA	200	2/3/2	2/3/2	2/3/2	2/3/2	2/3/2	2/3/2
BOX	100	88/124/100	69/100/77	141/247/191	29/41/32	12/22/15	94/179/141
BDEXP	1000	3/4/3	3/4/3	3/4/3	3/4/3	3/4/3	3/4/3
BDEXP	5000	3/4/3	3/4/3	3/4/3	3/4/3	3/4/3	3/4/3
BRYBND	1000	102/152/115	147/199/159	150/201/166	170/228/186	165/219/179	41/79/52
BRYBND	5000	67/117/81	93/144/110	41/94/62	40/90/56	85/144/109	128/185/146
CHAINWOO	100	500/592/539	489/593/534	599/822/725	403/479/434	618/722/662	509/586/544
CHAINWOO	1000	438/531/476	462/572/512	593/834/739	374/450/403	694/803/743	473/552/506
CHNROSNB	50	1074/1162/ 1110	872/967/917	1545/1742/1653	579/646/602	770/855/804	837/890/848
COSINE	100	12/25/15	11/24/13	12/25/14	11/24/13	11/23/13	12/24/14
COSINE	1000	13/27/15	12/2514	12/25/14	21/37/23	13/27/15	12/25/14
DIXMAANA	3000	17/20/21	17/28/20	17/28/21	17/34/24	17/28/20	16/35/25
DIXMAANA	9000	17/29/21	17/28/20	17/28/21	17/34/24	17/28/20	16/35/25
DIXMAANB	3000	16/35/24	16/38/26	16/38/26	18/32/22	18/52/36	18/31/22
DIXMAANB	9000	19/37/27	16/40/28	17/42/29	16/30/20	17/42/29	18/32/23
DIXMAANC	3000	18/51/38	20/43/31	20/43/31	20/40/29	18/46/31	17/39/28
DIXMAANC	9000	18/52/37	17/43/30	17/43/30	17/38/26	19/48/34	18/41/30
DIXMAAND	3000	16/47/30	16/47/31	16/47/31	13/37/23	17/48/32	13/37/23
DIXMAAND	9000	15/49/31	18/49/33	17/47/32	14/38/24	15/49/31	12/35/22
DIXMAANE	3000	367/405/373	437/474/444	570/652/604	402/435/407	558/592/564	432/468/438
DIXMAANE	9000	874/928/893	617/668/634	777/873/825	774/808/777	1018/1055/1024	666/702/668
DIXMAANF	3000	347/388/356	364/415/380	490/556/517	318/361/327	562/609/575	400/439/408
DIXMAANF	9000	568/623/588	706/767/727	751/887/819	455/497/467	829/868/838	609/651620
DIXMAANG	3000	417/472/437	279/341/301	576/666/616	327/386/347	478/539/499	410/464/431
DIXMAANG	9000	644/712/671	569/647/605	982/1126/1055	483/546/507	868/936/895	671/722/688
DIXMAANH	3000	307/360/327	343/405/367	637/737/683	275/329/295	517/576/536	394/447/414
DIXMAANH	9000	705/780/738	494/566/521	715/834/776	523/579/541	742/799/762	549/603/566
DIXMAANI	3000	7512/7999/7796	6854/7343/7145	4080/4711/4478	6434/6469/6443	7849/7907/7870	5097/5130/5100
DIXMAANI	9000	7294/7750/7569	5592/6018/5843	4817/5636/5328	6568/6603/6572	7935/7967/7942	7140/7174/7146
DIXMAANJ	3000	398/457/425	349/402/368	587/680/635	470/518/488	927/982/950	667/718/685
DIXMAANJ	9000	412/467/434	347/409/371	754/839/794	519/570/538	536/587/556	508/556/525
DIXMAANK	3000	357/414/377	246/305/269	341/433/387	445/497/463	98/144/113	511/560/529
DIXMAANK	9000	370/427/389	288/437/308	437/520/473	442/495/461	143/190/160	405/453/423
DIXMAANL	3000	255/315/280	295/365/322	460/546/500	303/363/327	414/485/443	540/599/563
DIXMAANL	9000	352/414/379	254/315/276	476/565/515	299/355/320	534/600/561	495/553/516
DIXMAANM	3000	7983/8508/8299	7725/8267/8052	4092/4821/4538	4963/4998/4967	7067/7107/7075	5490/5526/5496
DIXMAANM	9000	8743/9328/9102	8326/8938/8699	3702/4318/4083	6521/6557/6528	8304/8347/8314	7930/7966/7933
DIXMAANN	3000	1175/1266/1214	1089/1178/1127	1472/1701/1608	1524/1563/1535	2485/2586/2535	1614/1715/1685
DIXMAANN	9000	1846/1977/1908	2106/2438/2300	1684/1815/ 1752	1428/1472/1442	2312/2430/ 2372	1708/1752/1723
DIXMAANO	3000	1073/1144/1101	983/1085/1034	946/1116/1040	1543/1582/1553	1815/1916/1826	1439/1484/1448
DIXMAANO	9000	1378/1472/1422	1543/1650/1596	1424/1612/1532	1192/1232/1200	1784/1830/1797	1597/1640/1608
DIXMAANP	3000	1035/1127/1076	868/952/903	922/1102/1022	1296/1346/1312	1649/1710/1673	1320/1370/1337
DIXMAANP	9000	1237/1345/1291	1042/1137/1088	1150/1355/1270	1088/1138/1102	1676/1734/1676	1541/1592/1665
DIXON3DQ	100	1491/1584/1538	1628/1711/1666	1580/1829/1726	767/807/779	1677/1725/1692	1164/1197/1170
DIXON3DQ	1000	–/–/–	–/–/–/	–/–/–	7853/7933/7886	–/–/–	–/–/–
EG2	1000	6/19/6	6/19/6	6/19/6	6/19/6	6/19/6	6/19/6
EIGENALS	110	4018/4482/4298	3876/4344/4155	2429/2995/2795	1967/2101/2033	1881/2163/2045	2475/2659/2573
EIGENBLS	110	513/561/527	560/604/571	908/996/944	494/541/504	853/910/873	598/639/608
FLETCHCR	100	1070/1122/1089	1163/1226/1188	2403/2672/2560	849/898/868	1938/2009/1970	1620/1669/1639
FLETCHCR	1000	7906/8002/7954	8095/8224/8163	–/–/–	5785/5841/5809	–/–/–	8952/8994/8915
FMINSRF2	5625	396/425/403	405/436/417	680/737/705	626/659/634	856/893/867	554/585/562
FMINSRF2	10000	544/580/554	498/534/512	549/609/577	542/575/552	1073/1170/1085	631/658/639
FMINSURF	5625	1087/1131/1100	847/897/868	1097/1225/1170	830/872/844	1259/1298/1372	925/956/945
FMINSURF	10000	1306/1384/1344	1147/1202/1172	1245/1406/1343	850/885/863	1734/1762/1742	1109/1138/1117
GENROSE	100	490/573/524	488/584/529	728/852/789	495/591/539	685/792/736	610/684/639
GENROSE	500	1882/2044/1967	1949/2122/2040	2389/2670/2552	1927/2062/1995	3330/3537/3445	2293/2414/2350
GENHUMPS	1000	4069/4266/4172	3767/3945/3855	5249/5644/5488	4218/4494/4350	6525/6797/6676	4990/5186/5091
LIARWHD	5000	348/1563/1385	272/1256/1119	332/1506/1342	127/245/197	151/339/274	146/391/331
LIARWHD	10000	352/1727/1547	266/1381/1246/	310/1514/1360	100/201/168	236/621/509	207/1028/929
LMINSURF	5625	412/444/419	453/484/464	565/623/592	467/502/475	803/835/812	554/583/559
LMINSURF	10000	571/602/580	475/511/491	683/754/719	532/568/545	1071/1100/1077	671/698/677
MANCINO	50	12/23/12	11/21/11	11/21/11	11/21/11	11/21/11	11/21/11
MANCINO	100	12/23/12	12/23/12	12/23/12	12/23/12	12/23/12	12/23/12

(*continued on next page*)

Table 2 (*continued*).

Problem	Dim	*DL1* $n_i/n_f/n_g$	*DL2* $n_i/n_f/n_g$	*DL3* $n_i/n_f/n_g$	*MDL* $n_i/n_f/n_g$	*HZ+* $n_i/n_f/n_g$	*DK+* $n_i/n_f/n_g$
MOREBV	1000	229/246/232	267/288/274	283/325/302	93/104/95	621/636/624	369/383/371
MOREBV	5000	134/147/137	131/1025/945	132/147/135	93/104/95	246/279/268	134/144/135
MSQRTALS	1024	–/–/–	–/–/–	7786/9003/8541	7212/7322/7261	–/–/–	7894/7947/7909
MSQRTBLS	1024	8479/9011/8793	7544/8033/7829	5697/6615/6273	3460/3547/3497	6258/6333/6288	5149/5198/5162
NLMSURF	1024	259/295/269	252/289/265	246/282/255	305/343/315	442/477/451	343/380/352
NLMSURF	5625	496/533/509	450/487/462	704/763/729	572/612/584	1144/1188/1159	705/741/716
NONSCOMP	5000	66/144/97	70/143/101	85/178/28	58/128/91	65/134/92	76/149/108
NONDQUAR	1000	5974/8129/7367	7657/10719/9655	4114/6782/5937	9914/10393/10208	7192/8258/7886	–/–/–
NONDQUAR	5000	6294/8406/7674	8495/11570/10482	4151/6580/5790	5967/6239/6134	9734/10850/10461	–/–/–
PENALTY1	100	178/419/342	175/400/324	169/395/321	142/315/257	198/439/367	210/470/390
POWELLSG	5000	570/1049/896	450/919/771	478/916/784	233/320/277	311/415/367	473/674/596
POWELLSG	10000	430/783/667	278/538/447	321/634/537	260/384/326	309/419/371	441/603/538
SPMSRTLS	1000	180/215/187	171/203/175	226/266/236	176/214/180	316/353/321	178/215/188
SPMSRTLS	4999	268/307/276	257/302/269	582/657/608	270/309/280	522/578/542	336/375/345
SROSENBR	1000	161/529/448	135/467/400	187/702/607	112/376/322	35/65/48	101/375/325
SROSENBR	5000	161/529/448	135/467/400	187/702/607	70/216/181	91/195/152	101/375/325
TESTQUAD	1000	–/–/–	–/–/–	–/–/–	9258/9654/9478	8822/9491/9219	–/–/–
TESTQUAD	5000	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–	–/–/–
TOINTGSS	5000	5/9/5	5/9/5	5/9/5	6/11/6	6/11/6	5/9/5
TOINTGSS	10000	4/7/4	5/9/5	5/9/5	5/9/5	5/9/5	5/9/5
TRIDIA	5000	5659/6064/5892	5792/6196/6016	5171/5980/5668	2513/2597/2541	4865/4914/4872	3262/3310/3264
TRIDIA	10000	–/–/–	–/–/–	7443/8615/8169	3666/3759/3695	7147/7207/7161	5440/5495/5444
VAREIGVL	100	34/63/34	29/54/29	29/54/29	28/52/28	31/57/31	29/54/29
VAREIGVL	500	35/61/35	33/58/33	39/70/39	32/57/32	35/60/35	32/56/32
WOODS	4000	457/629/553	514/715/640	657/1046/906	476/633/670	454/614/549	635/791/727
WOODS	10000	360/525/457	489/694/615	614/1024/879	505/675/608	403/546/489	431/579/519

Assumption 3.2. In some neighborhood $\mathcal{N}$ of $\mathcal{L}, f$ is continuously differentiable and its gradient is Lipschitz continuous, namely, there exists a positive constant $L > 0$ such that

$$\|g(x) - g(y)\| \leq L\|x - y\|, \qquad \forall x, y \in \mathcal{N}. \tag{3.1}$$

Remark 3.1. Assumption 3.1 implies that there exists a positive constant γ such that

$$\|g_k\| \leq \gamma, \quad \forall x \in \mathcal{L}. \tag{3.2}$$

Remark 3.2. In order to establish the convergence property of new method, we assume the new proposed parameter is bounded. For this purpose, we set $t_4^* \leftarrow min\{t_4^*, M\}$ where M is a large positive constant.

Lemma 3.1. *Suppose that Assumptions 3.1 and 3.2 hold. Consider any CG method of the form (1.2)* and *(1.5), where d_k is a descent direction and the stepsize α_k satisfies the standard Wolfe line search conditions. then*

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty. \tag{3.3}$$

Proof. From (1.4) and (3.1) we have

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

which implies

$$\alpha_k \geq \frac{\sigma - 1}{L} \frac{(g_k^T d_k)}{\|d_k\|^2}. \tag{3.4}$$

On the other hand, it follows from (1.3) and (3.4) that

$$f(x_k) - f(x_k + \alpha_k d_k) \geq -\delta\alpha_k g_k^T d_k \geq \delta \frac{1-\sigma}{L} \frac{(g_k^T d_k)^2}{\|d_k\|^2},$$

Then, summing up both sides of the above relation, and using Assumption 3.1, we obtain (3.3) which completes the proof. □

A sufficient condition for the convergence of conjugate gradient methods with strong Wolfe line search is expressed in the following lemma, proved by [24].

Table 3
The numerical results.

Problem	Dim	*JC+* $n_i/n_f/n_g$	*NHS2* $n_i/n_f/n_g$	*IFD* $n_i/n_f/n_g$
ARGLINA	100	2/3/2	2/3/2	2/3/2
ARGLINA	200	2/3/2	2/3/2	2/3/2
BDEXP	1000	3/4/3	3/4/3	3/4/3
BDEXP	5000	3/4/3	3/4/3	3/4/3
BRYBND	1000	37/74/49	95/141/106	119/284/249
BRYBND	5000	131/192/152	244/305/262	46/150/99
CHNROSNB	50	612/693/643	2508/3526/3244	733/1448/1421
COSINE	100	11/24/13	14/27/16	34/72/38
DIXMAANA	3000	16/32/23	17/36/26	9/36/22
DIXMAANA	9000	16/32/23	17/36/26	9/36/22
DIXMAANB	3000	16/35/24	19/49/37	7/51/33
DIXMAANB	9000	18/36/26	26/62/50	7/50/32
DIXMAANC	3000	17/39/28	27/54/42	7/61/39
DIXMAANC	9000	18/41/30	16/42/29	7/61/39
DIXMAAND	3000	13/37/23	13/37/23	9/68/43
DIXMAAND	9000	13/37/23	13/37/23	9/68/43
DIXMAANE	3000	454/496/461	2782/4343/3926	361/642/629
DIXMAANE	9000	997/1063/1022	6070/9089/8273	616/1141/1128
DIXMAANF	3000	433/477/443	2648/4104/3720	332/591/571
DIXMAANF	9000	1237/1321/1272	5900/9040/8172	588/1091/1071
DIXMAANG	3000	387/450/408	1821/2961/2665	327/595/573
DIXMAANG	9000	804/862/825	4644/6942/6313	566/1068/1046
DIXMAANH	3000	416/473/434	2837/43347/3936	326/608/579
DIXMAANH	9000	824/880/842	5414/8075/7348	361/1071/1044
DIXMAANI	3000	5515/5879/5726	–/–/–	–/–/–
DIXMAANI	9000	3687/3780/3727	–/–/–	–/–/–
DIXMAANJ	3000	605/673/634	2499/3967/3582	5803/11216/11188
DIXMAANJ	9000	543/598/567	2396/3556/3236	1018/1915/1895
DIXMAANK	3000	439/389/358	1996/3048/2758	4592/8905/8880
DIXMAANK	9000	412/464/428	1916/2909/2647	817/1535/1512
DIXMAANL	3000	406/465/430	1472/2254/2032	4331/8365/8336
DIXMAANL	9000	307/366/329	1590/2498/2244	750/1400/1374
DIXMAANM	3000	6378/6751/6597	–/–/–	–/–/–
DIXMAANM	9000	4895/5100/5097	–/–/–	–/–/–
DIXMAANN	3000	1457/1544/1500	–/–/–	–/–/–
DIXMAANN	9000	1596/1712/1654	–/–/–	4469/8628/8603
DIXMAANO	3000	1145/1221/1177	–/–/–	–/–/–
DIXMAANO	9000	2231/2370/2299	–/–/–	3943/7536/7512
DIXMAANP	3000	1160/1243/1197	8093/12530/11355	–/–/–
DIXMAANP	9000	866/931/884	–/–/–	2615/4993/4967
DIXON3DQ	100	1029/1104/1059	5916/7889/7271	950/1773/1766
DIXON3DQ	1000	8102/8742/8488	–/–/–	–/–/–
EG2	1000	6/19/6	6/19/6	3/18/5
EIGENALS	110	1877/2170/2047	–/–/–	1185/2614/2565
EIGENBLS	110	668/729/692	1724/2347/2168	443/912/892
FLETCHCR	100	1698/1822/1760	5489/7562/7014	100/2116/2103
FLETCHCR	1000	–/–/–	–/–/–	7043/12477/12467
FMINSRF2	5625	445/1822/1760	1773/2023/1919	441/739/736
FMINSRF2	10000	660/698/671	24489/2948/2780	529/902/901
FMINSURF	5625	1059/1106/1079	4446/5396/5057	782/1388/1384
FMINSURF	10000	1174/1243/1208	5210/6396/5984	1024/1831/1827
GENROSE	100	558/683/624	815/1109/1004	770/1417/1407
GENROSE	500	2332/2538/2450	4086/5396/5009	1816/3270/3258
GENHUMPS	1000	4981/5224/5120	7359/10221/9407	5055/10045/9819
LIARWHD	5000	174/764/679	263/1386/1244	714/1734/1600
LIARWHD	10000	169/843/754	277/1515/1368	248/805/675
LMINSURF	5625	449/480/460	1519/1753/1665	423/710/707
LMINSURF	10000	546/586/559	2527/2919/2769	545/926/925
MANCINO	50	11/21/11	11/21/11	11/21/11
MANCINO	100	12/23/12	12/23/12	12/23/12
MOREBV	1000	272/295/278	645/904/826	234/393/392
MOREBV	5000	104/116/105	151/190/171	109/171/169
MSQRTALS	1024	9874/10719/10386	–/–/–	–/–/–
MSQRTBLS	1024	6971/7554/7318	–/–/–	6556/12667/12653
NLMSURF	1024	251/286/264	793/863/823	267/455/453

(*continued on next page*)

Table 3 (*continued*).

Problem	Dim	*JC+* $n_i/n_f/n_g$	*NHS2* $n_i/n_f/n_g$	*IFD* $n_i/n_f/n_g$
NLMSURF	5625	493/529/503	2239/2465/2366	408/773/769
NONSCOMP	5000	7069/144/106	66/138/108	75/194/173
NONDQUAR	1000	7029/8263/7816	–/–/–	–/–/–
NONDQUAR	5000	5559/6441/6111	–/–/–	–/–/–
PENALTY1	100	187/429/354	270/898/766	32/232/167
POWELLSG	5000	227/336/288	2361/5214/4625	907/2075/2001
POWELLSG	10000	193/286/243	1654/3891/3421	1257//2793/2714
SPMSRTLS	1000	189/226/194	380/463/416/	195/340/334
SPMSRTLS	4999	313/565/500	856/1113/1015	302/515/510
SROSENBR	1000	115/378/320	177/764/670	23/104/81
SROSENBR	5000	115/378/320	161/691/605	23/104/81
TESTQUAD	1000	6815/7649/7312	8745/13238/12216	6974/14336/14308
TESTQUAD	5000	–/–/–	–/–/–	–/–/–
TOINTGSS	5000	6/11/6	6/11/6	6/11/6
TOINTGSS	10000	5/9/5	6/11/6	5/9/5
TRIDIA	5000	4843/5262/5074	–/–/–	4106/7822/7812
TRIDIA	10000	8430/9139/8846	–/–/–	6798/12990/12974
VAREIGVL	100	28/52/28	29/54/29	28/62/33
VAREIGVL	500	37/64/37	33/57/33	28/66/39
WOODS	4000	394/565/500	1306/2585/2293	309/783/694
WOODS	10000	371/503/449	1165/2344/2068	333/777/707

Lemma 3.2. *Suppose that Assumptions 3.1 and 3.2 hold. Consider any CG method of the form (1.2)* and *(1.5), where d_k is a descent direction and the stepsize α_k satisfies the strong Wolfe line search conditions. If*

$$\sum_{k=0}^{\infty} \frac{1}{\|d_k\|^2} = \infty,$$

then

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

Lemma 3.3. *Consider the proposed CG method, including (1.2), (1.5), and (1.8), with $t = t_4^*$. If the line search procedure guarantees $d_k^T y_k \geq 0$, for all $k \geq 0$, then, we have*

$$g_{k+1}^T d_{k+1} \leq -c\|g_{k+1}\|^2, \tag{3.5}$$

where $c = (1 - \frac{1}{4\nu})$.

Proof. It follows from $d_k^T y_k \geq 0$ that

$$\begin{aligned} g_{k+1}^T d_{k+1} &= g_{k+1}^T\Big(-g_{k+1} + \frac{g_{k+1}^T y_k}{d_k^T y_k} d_k - t_4^* \frac{g_{k+1}^T s_k}{d_k^T y_k} d_k\Big), \\ &= -\|g_{k+1}\|^2 + \frac{(g_{k+1}^T y_k)(g_{k+1}^T s_k)}{s_k^T y_k} - t_4^* \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k}, \\ &\leq -\|g_{k+1}\|^2 + \frac{(g_{k+1}^T y_k)(g_{k+1}^T s_k)}{s_k^T y_k} d_k - \vartheta \frac{\|y_k\|^2}{s_k^T y_k} \frac{(g_{k+1}^T s_k)^2}{s_k^T y_k}, \\ &= g_{k+1}^T d_{k+1}^{\vartheta}. \end{aligned} \tag{3.6}$$

Then, from (3.6), and Theorem 1 of [25] which shows $g_{k+1}^T d_{k+1}^{\vartheta} \leq (\frac{1}{4\vartheta} - 1)\|g_{k+1}\|^2$, we obtain (3.5) which completes the proof. □

Theorem 3.1. *Suppose that Assumptions 3.1 and 3.2 hold. Consider the proposed CG method, including (1.2), (1.5), and (1.8), with $t = t_4^*$, and the stepsize α_k obtained by the strong Wolfe line search conditions. If f be a strongly convex function on the level set $\mathcal{L}$, then, we have*

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

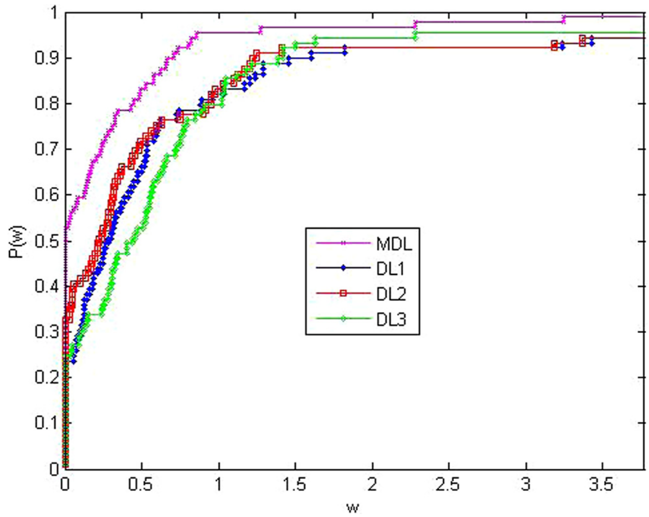

Fig. 1. Performance profile of MDL, DL1, DL2 and DL3 methods in terms of number of iterations.

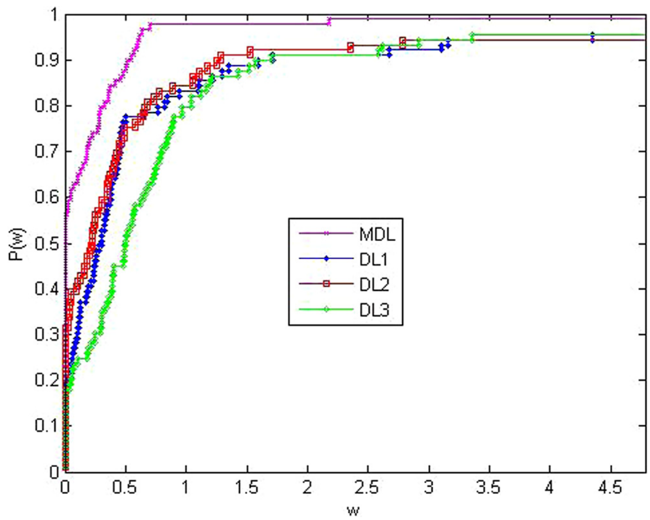

Fig. 2. Performance profile of MDL, DL1, DL2 and DL3 methods in terms of number of function evaluations.

Proof. For the strongly convex function f, we know that there exists a positive constant ν such that

$$s_k^T y_k \geq \nu \|s_k\|^2. \tag{3.7}$$

Then from (3.7), (3.1) and (3.2), we obtain

$$\begin{aligned}
\|d_{k+1}\| &= \|-g_{k+1} + \beta_k d_k\| \\
&\leq \|g_{k+1}\| + \left| \frac{g_{k+1}^T y_k}{d_k^T y_k} - t\frac{g_{k+1}^T s_k}{d_k^T y_k} \right| \|d_k\| \\
&\leq \|g_{k+1}\| + \left(\frac{\|g_{k+1}\| \|y_k\|}{s_k^T y_k} + t\frac{\|g_{k+1}\| \|s_k\|}{s_k^T y_k} \right) \|s_k\| \\
&\leq \|g_{k+1}\| + \left(\frac{\|g_{k+1}\|(\|y_k\| + t\|s_k\|)}{s_k^T y_k} \right) \|s_k\| \\
&\leq \|g_{k+1}\| + \left(\frac{\|g_{k+1}\|((L+t)\|s_k\|)}{s_k^T y_k} \right) \|s_k\|
\end{aligned}$$

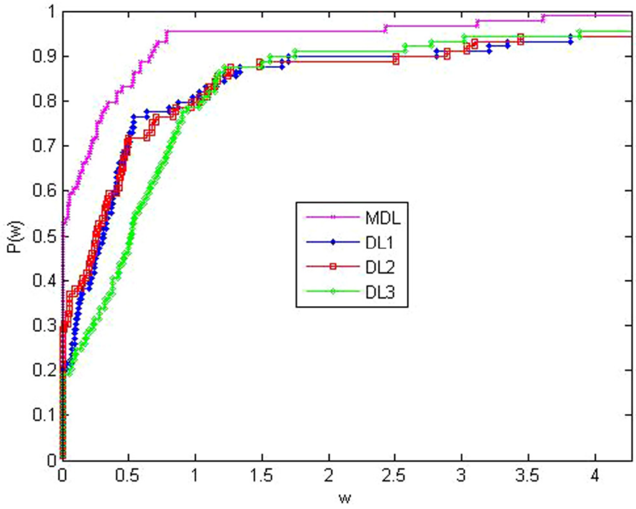

Fig. 3. Performance profile of MDL, DL1, DL2 and DL3 methods in terms of number of gradient evaluations.

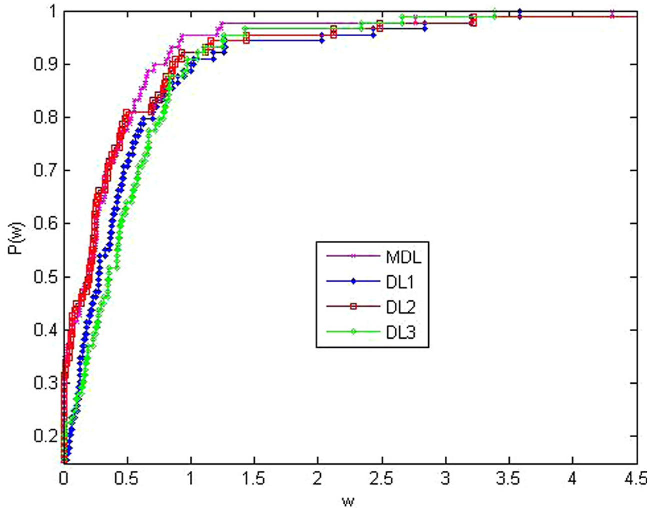

Fig. 4. Performance profile of MDL, DL1, DL2 and DL3 methods in terms of CPU time.

$$
\begin{aligned}
&\leq \|g_{k+1}\| + \left(\frac{\|g_{k+1}\|((L+t)\|s_k\|)}{\nu\|s_k\|^2}\right)\|s_k\| \\
&\leq (1+\frac{(L+M)}{\nu})\gamma,
\end{aligned} \tag{3.8}
$$

where the second inequality is obtained from the Cauchy–Schwarz inequality, and the last inequality is obtained from Remark 3.2. It follows from (3.8) that

$$\sum_{k=0}^{\infty}\frac{1}{\|d_k\|^2} \geq ((1+\frac{(L+M)}{\nu})\gamma)^{-2}\sum_{k=0}^{\infty}1 = \infty,$$

So, from Lemma 3.2, we conclude

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

which completes the proof. □

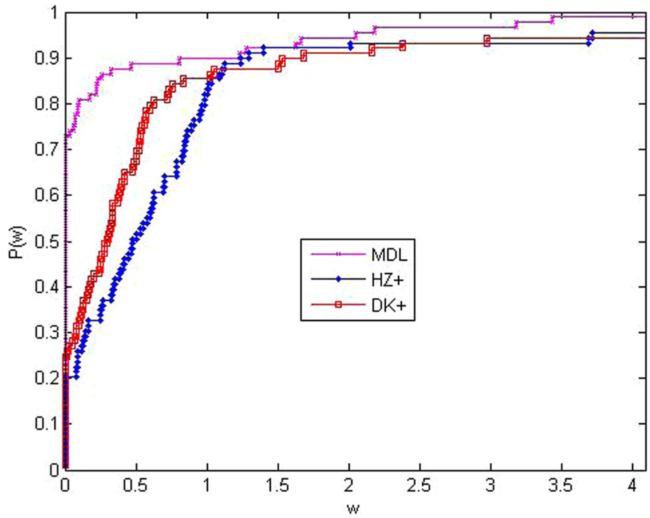

Fig. 5. Performance profile of MDL, HZ and DK methods in terms of number of iterations.

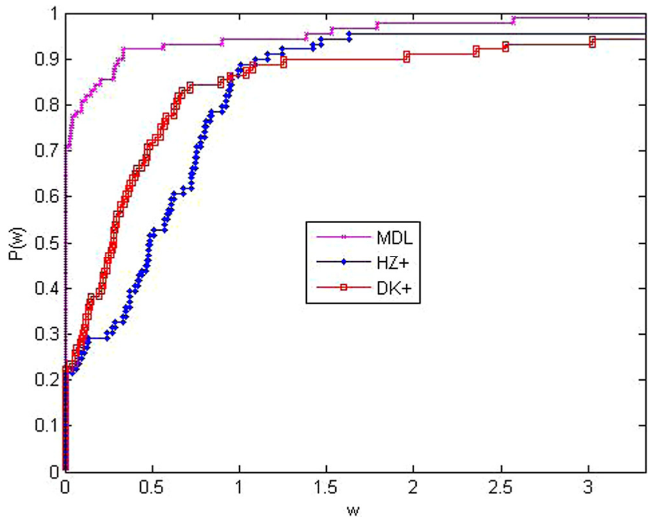

Fig. 6. Performance profile of MDL, HZ and DK methods in terms of number of function evaluations.

In order to prove the global convergence for general function, we use the truncation technique in [26] and update β_k as below

$$\beta_k^{DL+} = max\{\frac{g_{k+1}^T y_k}{d_k^T y_k}, 0\} - t_4^* \frac{g_{k+1}^T s_k}{d_k^T y_k}.$$

The proof is similar to that of Theorem 3.6 in [10].

4. Computational efficiency

In this section, we first compare the computational results obtained from the implementation of our proposed method (based on (1.2), (1.5), and (1.9) with $t = t_4^*$) denoted by "MDL", and three variants of the $DL+$ method (1.9) with $t = t_1, t = t_2$, and $t = t_3$ denoted respectively by "DL1", "DL2" and "DL3". Then we compare the computational results of our method with the CG–DESCENT method version 5.3 [10] denoted by "HZ+", and the method proposed by Dai and Kou [12] denoted by "DK+". We also compare our results with those of the following CG methods:

–"JC+": Algorithm JC proposed in [27] with $\theta_k = \theta_k^{JC+}$.

–"IDF": Algorithm proposed in [28] with $\beta_k = \beta_k^{IFD}$.

–"NHS2": Algorithm 4.2 proposed in [29].

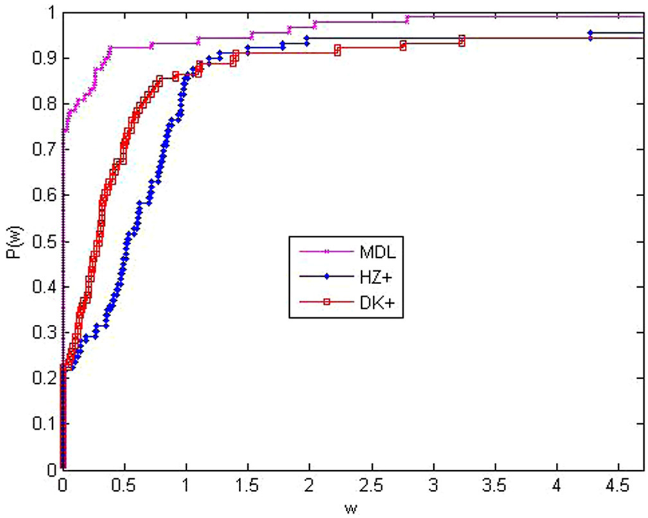

Fig. 7. Performance profile of MDL, HZ and DK methods in terms of number of gradient evaluations.

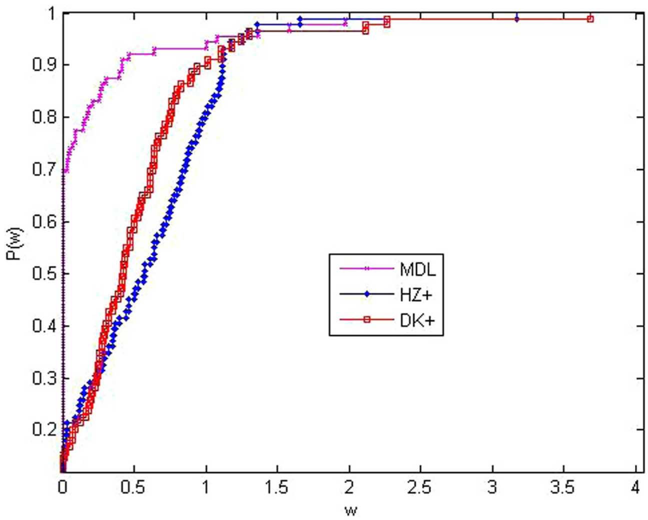

Fig. 8. Performance profile of MDL, HZ and DK methods in terms of CPU time.

All the methods have been coded in MATLAB and ran on a PC (CPU 2.5 GHZ, RAM 3.8 GB) with Linux operating system. The test problems have been taken from the CUTEst library [30]. The dimensions of the problems range from 50 to 10,000. Table 1 lists those problems with their dimensions. We used the strong Wolfe line search conditions and computed the initial guess of the stepsize by the scheme proposed in [22]. The iterations were stopped when the number of iterations exceeded 10,000 or $\|g_k\| \leq \varepsilon$, and the other used parameters are

$$\varepsilon = 10^{-6}, \qquad \delta = 0.01, \qquad \sigma = 0.9, \qquad \alpha_{0,0} = 1, \qquad \alpha_{k+1,0} = \alpha_k \frac{g_k^T d_k}{g_{k+1}^T d_{k+1}}.$$

Also, we consider $\nu = 0.26$ for the MDL method. We utilized the performance profile of Dolan and Moré [31] (in log_2 scale) to compare numerical results of the methods. Given a set of problems $\mathcal{P}$ and a set of solvers S, we define $i_{p,s}$ as the number of iterations required to solve problem p by solver s. The performance ratio is given by

$$r_{p,s} = \frac{i_{p,s}}{\{min \quad r_{p,s} | s \in S\}}.$$

Then, the performance profile is defined by

$$P_s(w) = \frac{\{size \quad p \in \mathcal{P} | \quad i_{p,s} \leq w\}}{size \quad \mathcal{P}}.$$

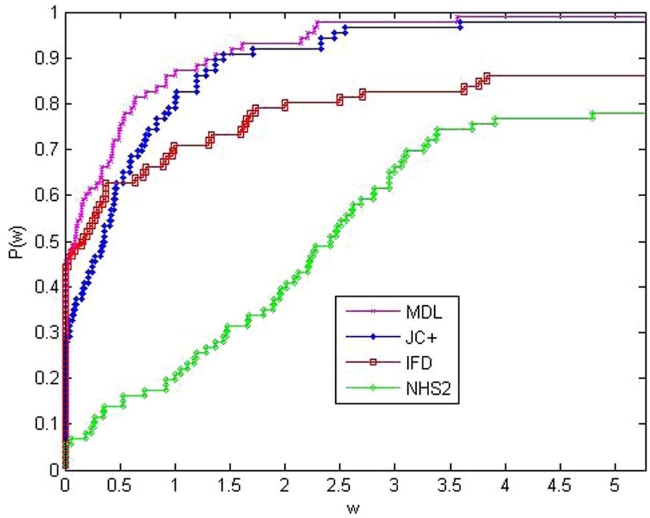

Fig. 9. Performance profile of MDL, JC+, IFD and NHS2 methods in terms of number of iterations.

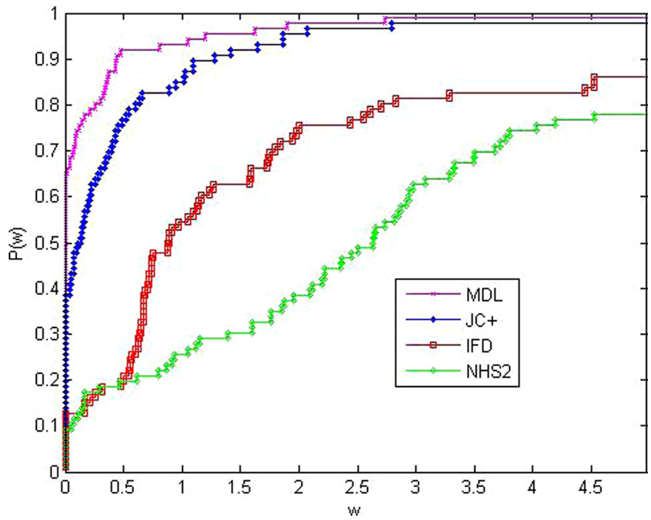

Fig. 10. Performance profile of MDL, JC+, IFD and NHS2 methods in terms of number of function evaluations.

Detailed numerical results are listed in Tables 2 and 3, which include the number of iterations n_i, the number of function evaluations n_f, and the number of gradient evaluations n_g, respectively. In Tables 2 and 3, '-' indicates the convergence failure of the used method for that problem, within 10,000 iterations.

The performance comparisons of MDL method with those of DL1, DL2, DL3 are reported in Fig. 1 in terms of number of iterations n_i, in Fig. 2 in terms of number of function evaluations n_f, in Fig. 3 in terms of number of gradient evaluations n_g, and in Fig. 4 in terms of CPU time. As these figures illustrate, it can be seen that the method MDL performs better than the three other methods, with respect to the number of iterations, the number of function evaluations, the number of gradient evaluations, and the CPU time.

In Fig. 5, we present the performance profile of the methods MDL, HZ+, and DK+, based on the number of iterations. This figure illustrates that MDL outperforms the other two methods. From Fig. 6, it is concluded that MDL is more efficient than others, with respect to the number of function evaluations. It is seen that MDL method, in comparison with the other mentioned methods, solves overall about 66% of the test problems with the least number of function evaluations. Fig. 7 shows that MDL method performs slightly superior to HZ+, and DK+ methods with respect to the number of gradient evaluations. Numerical results show that MDL method solves about 68% of the test problems with relatively least number gradient evaluations. Fig. 8 illustrates a better performance of MDL method with respect to the CPU time. We found that MDL solves about 70% of the test problems with the least CPU time.

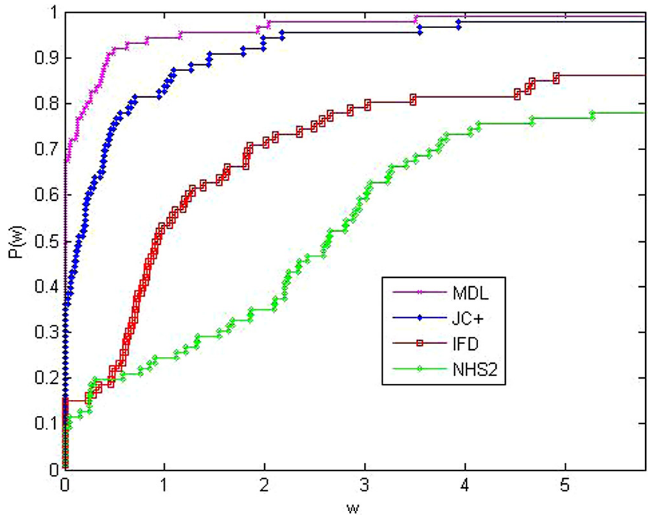

Fig. 11. Performance profile of MDL, JC+, IFD and NHS2 methods in terms of number of gradient evaluations.

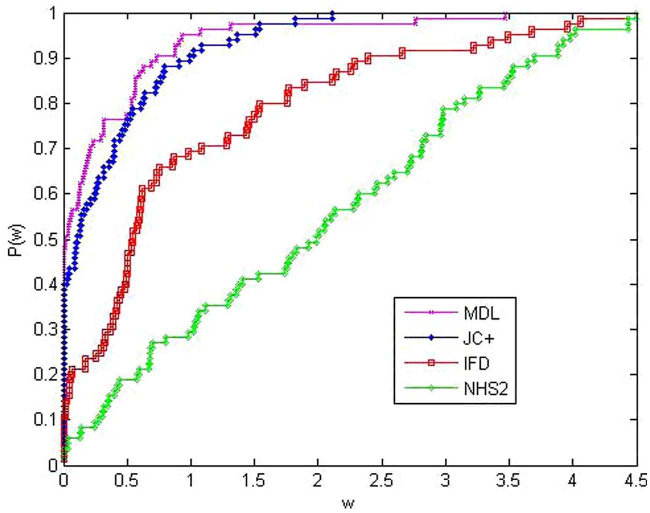

Fig. 12. Performance profile of MDL, JC+, IFD and NHS2 methods in terms of CPU time.

In Fig. 9, we compare the performance profile of MDL, JC+, IDF, and NHS2 methods based on number of iterations. In Fig. 10, we compare MDL, JC+, IDF, and NHS2 methods based on number of function evaluations. We observed that MDL method solves about 53% of the test problems with the least number of function evaluations. From Fig. 11, we see that MDL method is more efficient than others, with respect to the number of gradient evaluations, since it solves about 55% of the test problems with the least number of gradient evaluations. Fig. 12 shows the performance profile of MDL, JC+, IDF, and NHS2 methods based on the CPU time. The MDL method solves about 48% of the test problems with the least CPU time.

5. Conclusions

In this paper, we presented a new parameter for Dai–Liao family of CG methods. The new parameter contains not only the gradient information but also some Hessian matrix information. Moreover, it utilizes a modified BFGS update in order to achieve higher order accuracy in approximating the second order curvature of the objective function. Under some mild assumptions, we established the global convergence property of our new CG method. Numerical comparisons on a large class of well known test problems indicated that the new method is more efficient and robust in application.

Acknowledgments

The authors like to thank the principal editor Prof. André A. Keller and the two other reviewers for their comments and suggestions that have improved the paper very much.

References

[1] R. Fletcher, C.M. Reeves, Function minimization by conjugate gradients, Comput. J. 7 (2) (1964) 149–154.
[2] Y.H. Dai, Y. Yuan, A nonlinear conjugate gradient method with a strong global convergence property, SIAM. J. Optim. 10 (1) (1999) 177–182.
[3] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 49 (1952) 409–436.
[4] Y. Liu, C. Storey, Efficient generalized conjugate gradient algorithms, J. Optim. Theory Appl. 69 (1) (1991) 129–137.
[5] E. Polak, G. Ribiére, Note sur la convergence de directions conjugées, Rev. Fr. Inform. Rech. Oper. 3 (16) (1969) 35–43.
[6] B.T. Polyak, The conjugate gradient method in extremal problems, USSR. Comput. Math. Math. Phys. 9 (4) (1969) 94–112.
[7] W.W. Hager, H. Zhang, A survey of nonlinear conjugate gradient methods, Pac. J. Optim. 2 (1) (2006) 35–58.
[8] Y.H. Dai, L.Z. Liao, New conjugacy conditions and related nonlinear conjugate gradient methods, Appl. Math. Optim. 43 (1) (2001) 87–101.
[9] N. Andrei, Open problems in nonlinear conjugate gradient algorithms for uncon- strained optimization, Bull. Malays. Math. Sci. Soc. 34 (2) (2011).
[10] W.W. Hager, H. Zhang, A new conjugate gradient method with guaranteed descent and an efficient line search, SIAM. J. optim. 16 (1) (2005) 170–192.
[11] W. Sun, Y.X. Yuan, Optimization theory and methods, in: Nonlinear Programming, Springer, New York, 2006.
[12] Y.-H. Dai, C.X. Kou, A nonlinear conjugate gradient algorithm with an optimal property and an improved wolfe line search, SIAM. J. Optim. 23 (1) (2013) 296–320.
[13] S. Babaie-Kafaki, R. Ghanbari, A descent family of Dai–Liao conjugate gradient methods, Optim. Methods Softw. 29 (3) (2014) 583–591.
[14] S. Babaie-Kafaki, R. Ghanbari, Two optimal Dai–Liao conjugate gradient methods, Optim. 64 (11) (2015) 2277–2287.
[15] S. Babaie-Kafaki, R. Ghanbari, A class of adaptive Dai–Liao conjugate gradient methods based on the scaled memoryless BFGS update, 4OR 15 (1) (2017) 85–92.
[16] M. Fatemi, A new efficient conjugate gradient method for unconstrained optimiza- tion, J. Comput. Appl. Math. 300 (2016) 207–216.
[17] M. Fatemi, An optimal parameter for Dai–Liao family of conjugate gradient methods, J. Optim. Theory Appl. 169 (2) (2016) 587–605.
[18] S. Babaie-Kafaki, R. Ghanbari, The Dai–Liao nonlinear conjugate gradient method with optimal parameter choices, European J. Oper. Res. 234 (3) (2014) 625–630.
[19] N. Andrei, A Dai–Liao conjugate gradient algorithm with clustering of eigenvalues, Numer. Algorithms 77 (4) (2018) 1273–1282.
[20] D.H. Li, M. Fukushima, A modified BFGS method and its global convergence in nonconvex minimization, J. Comput. Appl. Math. 129 (1) (2001) 15–35.
[21] C.X. Kou, Y.H. Dai, A modified self-scaling memoryless Broyden–Fletcher–Goldfarb–Shanno method for unconstrained optimization, J. Optim. Theory Appl. 165 (1) (2015) 209–224.
[22] J. Nocedal, S. Wright, Numerical Optimization, second ed., springer, New York, 2006.
[23] Z. Aminifard, S. Babaie-Kafaki, An optimal parameter choice for the Dai–Liao family of conjugate gradient methods by avoiding a direction of the maximum magnification by the search direction matrix, 4OR (2018) 1–14.
[24] Y. Dai, J. Han, G. Liu, D. Sun, H. Yin, Y.-X. Yuan, Convergence properties of nonlinear conjugate gradient methods, SIAM. J. Optim. 10 (2) (2000) 345–358.
[25] S. Babaie-Kafaki, On the sufficient descent condition of the Hager–Zhang conjugate gradient methods, 4OR 12 (3) (2014) 285–292.
[26] J.C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, SIAM J. Optim. 2 (1) (1992) 21–42.
[27] J. Jian, Q. Chen, X. Jiang, Y. Zeng, J. Yin, A new spectral conjugate gradient method for large-scale unconstrained optimization, Optim. Methods Softw. 32 (3) (2017) 503–515.
[28] X. Jiang, J. Jian, Improved Fletcher–Reeves and Dai–Yuan conjugate gradient methods with the strong wolfe line search, J. Comput. Appl. Math. 348 (1) (2019) 525–534.
[29] X.L. Dong, D.R. Han, R. Ghanbari, X.L. Li, Z.F. Dai, Some new three-term Hestenes–Stiefel conjugate gradient methods with affine combination, Optim. 66 (5) (2017) 759–776.
[30] N.I. Gould, D. Orban, P.L. Toint, Cutest: a constrained and unconstrained testing environment with safe threads for mathematical optimization, Comput. Optim. Appl. 60 (3) (2015) 545–557.
[31] E.D. Dolan, J. Moré, Benchmarking optimization software with performance profiles, Math. Program. 91 (2) (2002) 201–213.