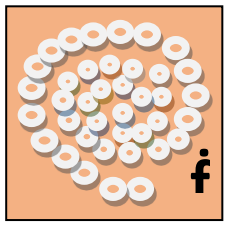

International Journal of Mathematical Analysis and Modelling

(Formerly Journal of the Nigerian Society for Mathematical Biology)

Volume 8 Issue 2, 2025

ISSN (Print): 2682 - 5694
ISSN (Online): 2682 – 5708
www.tnsmb.org

A modified Dai-Liao-type conjugate gradient method for unconstrained optimization

S.A. Ayinde*and J.F. Adelodun†

Abstract

Conjugate gradient (CG) methods have proved over the years to be efficient for solving large-scale unconstrained optimization problems. In this paper, a Dai-Liao (DL)-type CG method is proposed using proper combination of the update parameters of LS-CG and AyO-CG methods. Under some certain assumptions, descent and convergence properties were established. Preliminary results illustrate that the new schemes can compete favorably well with some existing ones, when subjected to numerical test under Dolan and More performance profile tools.

Keywords and phrases: global convergence; unconstrained optimization; strong Wolfe conditions; descent direction; step length
AMS subject classifications: 65K05, 90C06, 90C52, 90C56

1 Introduction

The paper proposes a hybrid conjugate gradient method to solve an optimization problem

$$min_{u\in R^n} f(u), \tag{1.1}$$

where $f : R^n \longrightarrow R$ is continuously differentiable and g_k is the gradient of f at u_k. CG methods are effective and very easy to use for solving (1) by using the following iteration.

$$u_{k+1} = u_k + \alpha_k d_k, \tag{1.2}$$

where $\alpha_k > 0$ representing the step-length can be determined from a line search that uses direction d_k given by

$$d_k = \begin{cases} -g_k, & k = 0, \\ -g_k + \beta_k d_{k-1}, & k \geq 1. \end{cases} \tag{1.3}$$

In any CG method, the formula $\beta_k \in R$ is the major parameter used in naming and characterizing any CG method and is known as an update parameter. The following list of formulas β_k are the first and second generations of CG methods presented and are referred to as the classical methods.

*Corresponding Author; Department of Basic Sciences , Babcock University, Ilishan-Remo, Ogun State, Nigeria. email: ayindes@babcock.edu.ng
†Department of Basic Sciences , Babcock University, Ilishan-Remo, Ogun State, Nigeria. email: adelodunj@babcock.edu.ng

$$\begin{cases} \beta_k^{HS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \\ \beta_k^{FR} = \frac{||g_k||^2}{||g_{k-1}||^2}, \\ \beta_k^{PRP} = \frac{g_k^T y_{k-1}}{||g_{k-1}||^2}, \\ \beta_k^{LS} = \frac{-g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}}, \\ \beta_k^{DY} = \frac{||g_k||^2}{d_{k-1}^T y_{k-1}}, \\ \beta_k^{CD} = \frac{-||g_k+1||^2}{d_k^T g_k}, \end{cases} \tag{1.4}$$

where $y_{k-1} = g_k - g_{k-1}$ and $||.||$ is the euclidean norm.

CG methods are studied due to their efficiency and low memory status for the solution of large-scale unconstrained optimization problems. A number of classical CG methods were introduced and studied in [1-6]. As a result of failures attributed to the classical methods which include jamming, poor convergences rate etc, other notable methods called hybrid CG methods have been introduced. For more on hybrid CG methods, see [7-11]. In 2001, [11] introduced an efficient conjugate gradient method, called DL CG method by introducing a scaling factor in the conjugacy condition. Recently, a number of authors, especially in [7, 12, 14] proposed some more efficient Dai-Liao-type methods by using different techniques. Lately, [12] proposed a Dai-Liao-type method that was more promising than Dai-Liao method. This influenced the consideration of AyO CG method proposed in the work of [12] for modification here. In what follows, despite the fact that LS-method is susceptible to jamming, its performance in general is better that the performance of methods with $||g_k||$ in the numerator of β_k (see [10]). Evidence of this is obvious from the work in [8]. This informed the reason LS CG method is chosen to replace the first term in AyO CG method in this work.
This paper extends the study of Dai-Liao-type conjugate gradient methods with the introduction of a robust hybrid CG method that enhances the better performance of LS CG method and also possesses some Hessian matrices information inherited from DL CG method to solve unconstrained optimizations problems. Analyses of its descent and convergence properties are done alongside numerical experiments to showcase the efficiency and robustness of the proposed method.
Section 2 of this work explains the proposed method and the algorithm for it. In section 3, convergence and descent properties of the method are rigorously considered. Numerical test and discussion of results are presented in section 4. Section 5 gives the conclusion.

2 Proposed β_k

In any CG method, the choice of the step-length α_k in (2) is a condition for global convergence. Hence, the following strong Wolfe conditions

$$f(u_k) - f(u_k + \alpha_k d_k) \geq -\delta \alpha_k d_k^T g_k \tag{2.1}$$

and

$$|g(x_k + \alpha_k d_k)^T d_k| \leq -\sigma g_k^T d_k \tag{2.2}$$

where $0 < \delta \leq \sigma < 1$ are used to estimate α_k.
This paper proposes another CG method by replacing the first term in AyO CG parameter

$$\beta_k^{AyO} = \frac{||g_k||^2}{d_{k-1}^T y_{k-1}} + \frac{t g_k^T s_{k-1}}{d_{k-1}^T g_{k-1}} \tag{2.3}$$

with LS CG parameter

$$\beta_k^{LS} = \frac{-g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}}, \tag{2.4}$$

thereby having

$$\beta_k^{LSAyO} = \frac{-g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}} + \frac{t g_k^T s_{k-1}}{d_{k-1}^T g_{k-1}}. \tag{2.5}$$

The formula β_k^{LSAyO} alongside (2) and (3) shall be referred to as LSAyO CG method.

2.1 Algorithm (new β_k)
Step 1: With the initial point $u_0 \in R^n$ and $\epsilon > 0$, we set $k = 0$ and $d_0 = -g_0$. If $||g_k|| \leq \epsilon$, then stop;
Step 2: Find $\alpha_k > 0$ by using strong Wolfe conditions (5-6) ;
Step 3: Compute β_k^{LSAyO} and generate the sequences $\{u_k\}$, $\{g_k\}$ and $\{d_k\}$;
Step 4: Set $k = k + 1$ and proceed to step 2.

3 Convergence analysis

In this section, descent and convergence properties of the proposed method are considered and established by making use of the following assumption. We shall first discuss the descent property.

Assumption 1
The objective function (1) satisfies the conditions highlighted below.
(a) The set

$$U = \{u | f(u) \leq f(v)\} \tag{3.1}$$

with $v \in \mathbb{R}^n$ is bounded.
(b) f is continuous and differentiable in W and its gradient $g(u) = \nabla f(u)$ alongside the Lipschitz constant L satisfy Lipschitz continuity

$$||g(u) - g(v)|| \leq L||u - v|| \tag{3.2}$$

for any $u, v \in W$ where W is a neighborhood in U.

Furthermore, by Assumption 1 there exists two constants D and c that satisfy the following inequalities

$$||u - v|| \leq D, \tag{3.3}$$

for any $u, v \in W$
and

$$||g(u)|| \leq c, \tag{3.4}$$

for any $u \in W$.

Lemma 2 *Suppose d_k and g_k are determined by the CG method algorithm given in 2.1. Then, d_k satisfies the following condition*

$$g_k^T d_k < 0, \tag{3.5}$$

for each $k \geq 0$.
Proof. We show by induction. It is obvious for the case when $k = 0$ that

$$g_0^T d_0 = -||g_0||. \tag{3.6}$$

Let (14) be valid for $k \geq 1$. By pre-multiplying $d_k = -g_k + \beta_k d_{k-1}$ by g_k and setting $\beta_k = \beta_k^{LSAyO}$, we have

$$g_k^T d_k = -||g_k||^2 + \beta_k^{LSAyO} g_k^T d_{k-1}. \tag{3.7}$$

Hence,

$$g_k^T d_k = -||g_k||^2 + (\frac{-g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}} + \frac{t g_k^T s_{k-1}}{d_{k-1}^T g_{k-1}}) g_k^T d_{k-1}. \tag{3.8}$$

That is,

$$g_k^T d_k = -||g_k||^2 + (\frac{-g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}} - \frac{t g_k^T s_{k-1}}{(-d_{k-1}^T g_{k-1})}) g_k^T d_{k-1}. \tag{3.9}$$

Or

$$g_k^T d_k = -||g_k||^2 + (\beta_k^{LS} - \frac{t g_k^T s_{k-1}}{(-d_{k-1}^T g_{k-1})}) g_k^T d_{k-1}. \tag{3.10}$$

Descent condition requires

$$y_{k-1}^T d_{k-1} > 0. \tag{3.11}$$

Therefore,

$$y_{k-1}^T d_{k-1} = (g_k - g_{k-1})^T d_{k-1} \tag{3.12}$$

$$= g_k^T d_{k-1} - g_{k-1}^T d_{k-1} \tag{3.13}$$

$$\leq |g_k^T d_{k-1}| - g_{k-1}^T d_{k-1}. \tag{3.14}$$

For (20) to hold always, then

$$g_{k-1}^T d_{k-1} < 0. \tag{3.15}$$

See [12] also.
Hence,

$$g_k^T d_k \leq -||g_k||^2 + \beta_k^{LS} g_k^T d_{k-1} \tag{3.16}$$

By [4]

$$g_k^T d_k < 0 \tag{3.17}$$

Theorem 3. *Suppose $f(u)$ is as defined above, the conditions in the assumption 1 hold and also that u_k be determined by the algorithm. Then,*

$$\lim \inf_{k \to \infty} ||g_k|| = 0 \tag{3.18}$$

Proof. This result is proved by contradiction. This implies that

$$||g_k|| \geq c \ \ \forall \ k. \tag{3.19}$$

See [6]. Since $d_{k-1}^T g_{k-1} < 0$, we have

$$\beta_k^{LSAyO} = \frac{-g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}} + \frac{t g_k^T s_{k-1}}{d_{k-1}^T g_{k-1}}. \tag{3.20}$$

$$\beta_k^{LSAyO} \leq \frac{g_k^T y_{k-1}}{(-d_{k-1}^T g_{k-1})} + \frac{t g_k^T s_{k-1}}{(-d_{k-1}^T g_{k-1})}. \tag{3.21}$$

Hence,

$$\beta_k^{LSAyO} \leq \frac{g_k^T y_{k-1} + t g_k^T s_{k-1}}{|-d_{k-1}^T g_{k-1}|} \tag{3.22}$$

$$\beta_k^{LSAyO} \leq \frac{g_k^T(g_k - g_{k-1}) + tg_k^T s_{k-1}}{|-d_{k-1}^T g_{k-1}|} \tag{3.23}$$

$$\beta_k^{LSAyO} \leq \frac{||g_k||||g_k - g_{k-1}|| + t||g_k||||s_{k-1}||}{||d_{k-1}||||g_{k-1}||} \tag{3.24}$$

Using (11), (12) and (13),

$$\beta_k^{LSAyO} \leq \frac{||g_k|||LD + t||g_k||D}{||d_{k-1}^T||c} \tag{3.25}$$

$$= \frac{D(L+t)||g_k||}{||d_{k-1}||c} \tag{3.26}$$

$$||d_k|| \leq ||g_k|| + \frac{D(L+t)||g_k||}{||d_{k-1}||c}||d_{k-1}||. \tag{3.27}$$

$$= (1 + \frac{D(L+t)}{c})||g_k||. \tag{3.28}$$

$$||d_k||^2 \leq \frac{(c^2 + 2cD(L+t) + D^2(L+t)^2)||g_k||^2}{c^2}. \tag{3.29}$$

$$\frac{||d_k||^2}{||g_k||^4} \leq \frac{(\frac{(c^2+2cD(L+t)+D^2(L+t)^2)}{c^2})c^2}{c^4}. \tag{3.30}$$

$$= \frac{||d_k||^2}{||g_k||^4} \leq \frac{c^2 + 2cD(L+t) + D^2(L+t)^2}{c^4}. \tag{3.31}$$

Hence,

$$\frac{||g_k||^4}{||d_k||^2} \geq \frac{c^4}{c^2 + 2cD(L+t) + D^2(L+t)^2}. \tag{3.32}$$

Equation (41) implies

$$\sum_{k=0}^{\infty} \frac{||g_k||^4}{||d_k||^2} \geq \frac{c^4}{c^2 + 2cD(L+t) + D^2(L+t)^2} = \infty. \tag{3.33}$$

This is a contradiction as a result of $||g_k|| \geq c$. Hence, (27) is satisfied and the result follows.

4 Numerical test and discussion of results

A number of test functions are selected from [15] and numerical experiments are carried out to show the effectiveness of the proposed method against existing methods in the areas of central processing unit (CPU) time, number of function evaluation (FE), gradient norm (G) and number of iterations (IT). All algorithms are executed on MATLAB 2015a that runs on windows 10 OS and RAM 3GD. Numerical test are done under the algorithm code using the settings $\delta = 0.0001$ and $\sigma = 0.9$ for all the methods. The system terminates the algorithm if $\|g_k\| \leq 10^{-6}$ or the number of iterations exceeds 4000.

Existing methods considered are LS [4], DL [11] and DY [16] CG-methods. Tables showing these performance profiles are given below. The following abbreviations are adopted on the tables: RMODF COSINE- RDC; RMODF SINE-RDS; Diagonal 4-DL4; Extended Himmelblau-EU; ARGLINB-ARB; Extended Rosenbrock-ERK; Extended Booth-EO; Diagonal 1-DL1; Raydan 1-Rn1; Raydan 2-Rn2; DQDRTIC-DC.

4.1 Numerical test

The four aforementioned numerical profiles are considered to test the strength, robustness and efficiency of the proposed LSAyO CG method against DL, LS and DY CG methods. Records of the results generated are given in Tables 1 and 2. On these Tables, $prob.$, $Dim.$ and F represent optimization problems, dimensions and failure of a method to solve the problem respectively. Dimensions used to generate the results recorded in the Tables are: 2, 100, 1000 and 10000. Graphical representations for the performance profiles are plotted by using Dolan and More [17] tools: Consider set S of p_s methods to be tested, subject to the given set P of n_s test functions. The tools use

$$r_{p,s} = \frac{R_{p,s}}{min\{R_{p,s} : s \in S \; and \; p \in P\}} \tag{4.1}$$

to test different methods given that $R_{p,s}$ can be either IT, FE, GN or CPU time to compare methods S for each problem P.

$$\phi_s(\tau) = \frac{1}{n_r}|p \in P : logr_{p,s}| \leq \tau \tag{4.2}$$

where $\tau \geq 0$ is the probability that $r_{p,s}$ resides in a factor $\tau \geq 1$ in connection to the method s. $\phi_s(\tau)$ gives the probability that a particular method will perform better than other methods for the value $\tau = 1$. Selected method $s \in S$ will fail to solve a problem whenever $r_i = r_{p,s}$ for some value r_i.

Table 1. Numerical result of CPU and function evaluation FE.

s/n	$Prob.$	$Dim.$	DL	LS	$LSAyO$	DY
			FE/CPU	FE/CPU	FE/CPU	FE/CPU
1	$DL4$	2	847/0.595	723/0.31	507/0.243	124/0.189
2		100	450/0.266	819/0.371	553/0.28	139/0.212
3		1000	633/0.518	818/0.451	698/0.353	144/0.238
4		10000	411/0.85	863/1.313	601/0.887	154/0.453
5	EU	2	F/F	1180/0.688	519/0.439	1029/1.108
6		10	F/F	1256/0.748	533/0.452	1112/1.151
7		1000	F/F	1290/0.839	539/0.544	1186/1.359
8		10000	410/1.208	1278/2.687	571/1.577	1210/3.121
9	$Rn1$	2	39/0.188	21568/49.591	17941/47.663	61/0.059
10		10	3/0.011	3/0.01	3/0.008	3/0.009
11		1000	3/0.011	3/0.011	3/0.011	3/0.011
12		10000	3/0.017	3/0.015	3/0.014	3/0.013
13	$Rn2$	2	13/0.064	19/0.096	20/0.096	49/0.203
14		100	15/0.07	19/0.092	20/0.08	55/0.224
15		1000	15/0.073	21/0.105	20/0.092	59/0.253
16		10000	15/0.113	21/0.147	20/0.132	63/0.372
17	$DL1$	2	15/0.094	43/0.262	203/0.203	59/0.349
18		100	15/0.082	43/0.245	200/0.196	59/0.318
19		1000	15/0.092	43/0.253	196/0.228	59/0.316
20		10000	15/0.093	43/0.28	176/0.359	59/0.383
21	DC	2	451/0.451	109/0.126	109/0.125	515/0.458
22		100	1317/1.413	2815/1.438	3139/1.324	1424/1.405
23		1000	1342/1.54	2165/1.086	3207/1.546	2118/1.983
24		10000	1030/2.233	3362/4.648	3552/4.891	2179/3.988
25	RDC	2	244/0.363	420/0.263	377/0.238	45/0.108
26		100	297/0.734	413/0.275	381/0.28	45/0.107
27		1000	922/0.942	457/0.426	463/0.381	271/0.64
28		10000	F/F	1039/2.835	429/0.945	135/0.497
29	RDS	2	15/0.078	119/0.58	23/0.119	41/0.169
30		100	15/0.076	135/0.635	25/0.124	45/0.174
31		1000	15/0.077	145/0.733	25/0.134	49/0.197
32		10000	15/0.108	157/1.127	25/0.186	53/0.324
33	BO	2	151/0.292	406/0.332	260/0.311	131/0.245
34		100	F/F	412/0.362	284/0.356	146/0.365
35		1000	F/F	490/0.469	285/0.401	155/0.386
36		10000	164/0.566	496/1.041	286/0.783	164/0.87
37	ERK	2	F/F	28707/14.834	3584/2.052	15642/5.807
38		100	F/F	18135/10.505	8218/4.21	16701/6.429
39		1000	F/F	4464/3.004	53960/12.148	F/F
40		10000	F/F	83276/141.314	6610/10.972	11057/15.642
41	ARB	2	10/0.03	1220/0.608	1213/0.597	15/0.027
42		100	40/0.014	40/0.016	40/0.014	40/0.015
43		1000	60/0.016	60/0.017	60/0.016	60/0.018
44		10000	80/0.073	80/0.071	80/0.067	80/0.069

Table 2. Numerical result of number of iterations and gradient norm.

s/n	$Prob.$	$Dim.$	DL	LS	$LSAyO$	DY
			IT/G	IT/G	IT/G	IT/G
1	$DL4$	2	$35/1.96E-07$	$25/8.07E-07$	$19/3.30E-07$	$21/8.78E-07$
2		100	$21/8.42E-08$	$29/2.67E-07$	$21/4.07E-07$	$24/7.01E-07$
3		1000	$32/2.80E-07$	$29/8.46E-07$	$23/2.24E-07$	$25/7.46E-07$
4		10000	$26/1.36E-07$	$31/5.79E-07$	$23/7.09E-07$	$27/4.60E-07$
5	EU	2	F/F	$54/7.88E-07$	$36/6.31E-07$	$123/8.44E-07$
6		100	F/F	$58/3.72E-07$	$38/8.87E-07$	$133/9.83E-07$
7		1000	F/F	$60/3.05E-07$	$40/7.84E-07$	$142/4.44E-07$
8		10000	$50/1.98E-07$	$60/9.69E-07$	$43/9.95E-07$	$145/6.67E-07$
9	$Rn1$	2	$19/7.93E-07$	$4000/1.49E-01$	$4000/6.51E-03$	$6/0.00E+00$
10		100	$1/4.74E-120$	$1/4.74E-120$	$1/4.74E-120$	$1/4.74E-120$
11		1000	$1/0.00E+00$	$1/0.00E+00$	$1/0.00E+00$	$1/0.00E+00$
12		10000	$1/0.00E+00$	$1/0.00E+00$	$1/0.00E+00$	$1/0.00E+00$
13	$Rn2$	2	$6/5.27E-07$	$9/4.65E-08$	$9/9.94E-09$	$24/6.52E-07$
14		100	$7/4.16E-10$	$9/3.29E-07$	$9/7.03E-08$	$27/8.80E-07$
15		1000	$7/1.31E-09$	$10/1.92E-08$	$9/2.22E-07$	$29/8.35E-07$
16		10000	$7/4.16E-09$	$10/6.06E-08$	$9/7.03E-07$	$31/7.93E-07$
17	$DL1$	2	$7/4.75E-09$	$21/4.78E-07$	$9/1.16E-10$	$29/9.58E-07$
18		100	$7/4.75E-09$	$21/4.78E-07$	$9/1.16E-10$	$29/9.58E-07$
19		1000	$7/4.75E-09$	$21/4.78E-07$	$9/1.16E-10$	$29/9.58E-07$
20		10000	$7/4.75E-09$	$21/4.78E-07$	$9/1.16E-10$	$29/9.58E-07$
21	DC	2	$45/7.59E-07$	$11/2.40E+08$	$11/2.49E+10$	$50/6.65E-07$
22		100	$135/8.14E-07$	$128/7.17E-07$	$117/6.28E-07$	$143/8.15E-07$
23		1000	$141/9.02E-07$	$85/8.39E-07$	$117/7.17E-07$	$213/9.96E-07$
24		10000	$107/9.90E-07$	$133/6.17E-07$	$134/9.18E-07$	$227/6.72E-07$
25	RDC	2	$33/5.91E-07$	$21/2.47E-07$	$20/5.16E-07$	$12/1.07E-07$
26		100	$68/1.20E-07$	$23/5.10E-07$	$22/9.00E-07$	$12/7.55E-07$
27		1000	$72/4.28E-07$	$31/8.11E-07$	$31/5.45E-07$	$72/9.36E-08$
28		10000	F/F	$113/6.60E-07$	$29/9.57E-07$	$36/5.81E-07$
29	RDS	2	$7/9.08E-10$	$59/8.09E-07$	$11/1.86E-07$	$20/5.28E-07$
30		100	$7/6.42E-09$	$67/9.60E-07$	$12/6.40E-08$	$22/8.70E-07$
31		1000	$7/2.03E-08$	$72/9.95E-07$	$12/2.02E-07$	$24/6.41E-07$
32		10000	$7/6.42E-08$	$78/8.24E-07$	$12/6.40E-07$	$26/4.73E-07$
33	BO	2	$28/4.73E-07$	$28/4.85E-07$	$27/9.29E-07$	$27/8.62E-07$
34		100	F/F	$29/9.45E-07$	$30/4.22E-08$	$30/9.74E-07$
35		1000	F/F	$35/3.95E-07$	$32/1.36E-07$	$32/6.55E-07$
36		10000	$32/9.81E-07$	$36/4.29E-07$	$32/4.31E-07$	$34/9.83E-07$
37	ERK	2	F/F	$1327/9.50E-07$	$181/1.44E+10$	$596/F$
38		100	F/F	$929/7.63E+09$	$344/7.26E-07$	$633/F$
39		1000	F/F	$231/2.68E+10$	$2464/1.90E+14$	F/F
40		10000	F/F	$4000/1.31E+06$	$330/3.45E+10$	$445/1.18E+10$
41	ARB	2	$2/2.98E-15$	$53/8.04E-07$	$53/8.04E-07$	$2/2.98E-15$
42		100	$1/1.18E+11$	$1/1.18E+11$	$1/1.18E+11$	$1/1.18E+11$
43		1000	$1/3.30E+17$	$1/3.30E+17$	$1/3.30E+17$	$1/3.30E+17$
44		10000	$1/9.06E+23$	$1/9.06E+23$	$1/9.06E+23$	$1/9.06E+23$

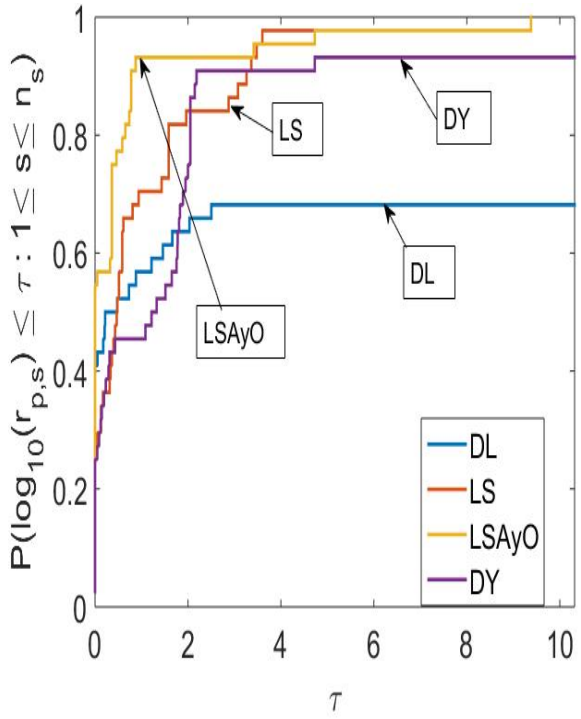

Figure 1: Iterations (IT)

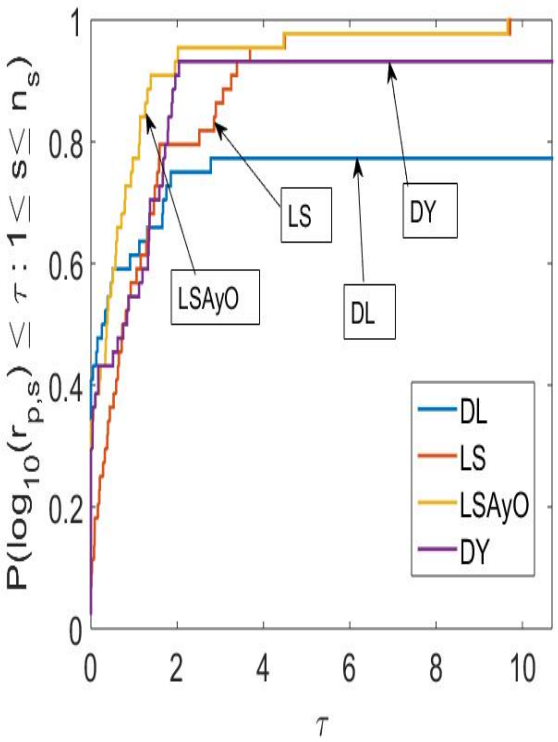

Figure 3: CPU time

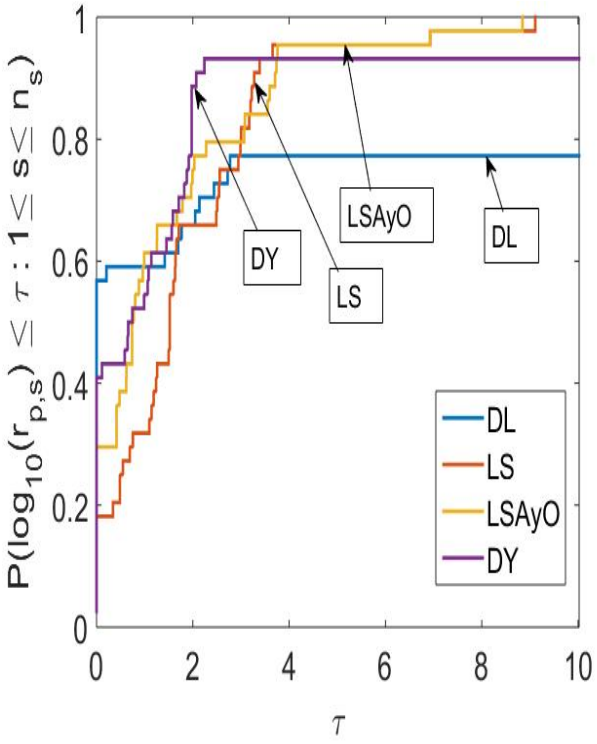

Figure 2: Function Evaluations (FE)

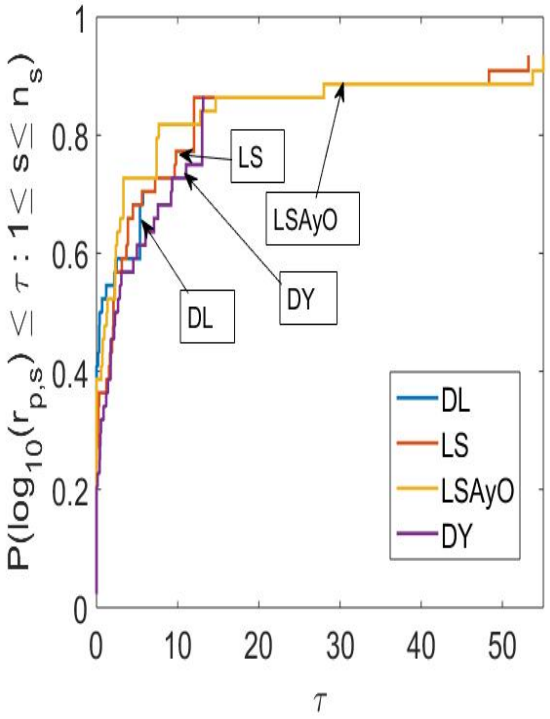

Figure 4: Gradient norm (G)

4.2 Discussion of result

In the Tables 1 and 2 where the four metric profiles were recorded, LSAyO CG and LS CG method solved 100% of the problems while DL and DY CG methods solved 77.3% and 95.5% of the problems respectively. In all, LSAyO CG method is faster than them. On the other hand our method showed weakness where we recorded high gradient norms at some points which may be as a result of too small step-length since there is convergence or ill-conditioning of the test functions.
Graphical Results of performance profiles strength capturing number of iterations, central processing unit time, function evaluation and gradient norms are shown in Figures (1-4). LSAyO method is seen to have outperformed other methods in Figures (1 and 3), where results of number of iterations and central processing unit time were considered. For function evaluation result shown in Figure 2, LSAyO method compete favorably well with other methods for the values of τ from $0-5$ and did better than them for the remaining values of τ. In Figure 4, LSAyO method compete well with other methods in the area of gradient norms.

5 Conclusion

This work modified the Dai-Liao-type CG method presented in [12] by replacing its first term with LS CG method to solve problems of the form (1). Rigorous analyses were done to establish descent and convergence properties of the method proposed. Furthermore, numerical test done on the methods tested were based on strong Wolfe line search scheme. Numerical profiles tested showed that our method performed better than other methods considered in this work. Moreover, the update parameter proposed in this work can be extended or modified for further research by considering its convex combination or variation in the value of t.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referee whose comments improved the original version of this manuscript.

References

[1] Al-Baali, M. (1985). Descent property and global convergence of the Fletcher-Reeves method with inexact line search, IMA Journal Numerical Analysis, 5, 121-124.

[2] Fletcher, R. and Reeves, C.M. (1964). Function minimization by conjugate gradients, Computer Journal, 7, 149-154.

[3] Hestenes, M.R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems, Journal of Research of the National Bureau of Standards, 49, 409-435.

[4] Liu, Y. and Storey, C. (1991). Efficient generalized conjugate gradient algorithms, part 1, Theory, Journal of Optimization Theory and Applications, 69, 129-137.

[5] Polak, E. and Ribiere, G. (1969). Note sur la convergence de directions conjugate, Analysis- Mode 'elisation Math 'ematique et Analyse Num 'erique, 3(R1), 35-43.

[6] Stanimirovic, P.S., Ivanov, B and Mosic, D. (2020). A survey of gradient methods for solving nonlinear optimization, Electronic Research Archive, 28(4), 1573– 1624.

[7] Ayinde, S.A., Agboola, S.O., Adelodun, J.F. and Hassan, S.O. (2023). Two new conjugate gradient methods in unconstrained optimization problems, Annals of Mathematics and Computer Science, 18, 26-39.

[8] Ayinde S.A., Osinuga I.A., Adio A.K., Agboola, S.O., Adelodun, J.F., Uka, U.A. and Awe, O. (2024). A descent conjugate gradient method for optimization problems, IAENG International Journal of Applied Mathematics, 54(9), 1765-1775.

[9] Ayinde S.A., Osinuga I.A., and Adio A.K. (2025). A new descent extension of DY conjugate gradient method based on DL approach, Filomat, 39(15), 5071-5084.

[10] Hager W.W. and Zhang H. (2006). A survey of nonlinear conjugate gradient methods, Pacific Journal of Optimization, 2(1), 35-58.

[11] Dai, Y. and Liao, L. (2001). New conjugacy conditions and related nonlinear conjugate gradient methods, Applied Mathematics and Optimization, 43, 87-101.

[12] Ayinde, S.A., Osinuga, I.A., Adio, A.K. and Adelodun, J.F. (2023). A new Dai-Liao-type conjugate gradient method for unconstrained optimization problems, Engineering Letters, 31(3), 1281-1290.

[13] Yao, S., and Qin, B. (2014). A hybrid of DL and WYL nonlinear conjugate gradient methods, Abstr. Appl. Anal., 279891

[14] Zoutendijk, G. (1970). Nonlinear programming, computational methods, In: J. Abadie (eds.): Integer and Nonlinear Programming, North-Holland, Amsterdam, 37–86.

[15]Andrei N. (2011). Open problems in nonlinear conjugate gradient algorithms for unconstrained optimization, Bulletin of the Malaysian Mathematical Sciences Society, 34(2), 319-330.

[16] Dai, Y. and Yuan, Y. (1999). A nonlinear conjugate gradient method with a strong global convergence property, SIAM Journal of Optimization, 10, 177-182.

[17] Dolan, E.D. and More, J.J. (2002). Benchmarking optimization software with performance profiles, Mathematical Programming, 91, 201-213.