


A descent Dai-Liao conjugate gradient method for nonlinear equations

Auwal Bala Abubakar^{1,2} · Poom Kumam^{1,3} 

Received: 7 February 2018 / Accepted: 2 May 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract In this work, we propose an algorithm for solving system of nonlinear equations. The idea is a combination of the descent **Dai-Liao** method by Babaie-Kafaki and Gambari (Optim. Meth. Soft. **29**(3), 583–591, 2014) and the **hyperplane projection method**. Using the monotonicity and Lipschitz continuity assumptions, we prove that the proposed method is globally convergent. Examples of numerical experiment show that the method is promising and efficient compared to the method proposed by Sun et al. (Journal of Inequalities and Applications **236**, 1–8, 2017).

Keywords Non-linear equations · Monotone equations · Derivative-free method · Projection method

Mathematics Subject Classification (2010) 65K05 · 90C06 · 90C52 · 90C56

✉ Poom Kumam
poom.kumam@mail.kmutt.ac.th

Auwal Bala Abubakar
ababubakar.mth@buk.edu.ng

- ¹ KMUTTFixed Point Research Laboratory, Department of Mathematics, Room SCL 802 Fixed Point Laboratory, Science Laboratory Building, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thrung Khru, Bangkok 10140, Thailand
- ² Department of Mathematical Sciences, Faculty of Physical Sciences, Bayero University Kano, Kano, Nigeria
- ³ KMUTT-Fixed Point Theory and Applications Research Group, Theoretical and Computational Science Center (TaCs), Science Laboratory Building, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thrung Khru, Bangkok 10140, Thailand

1 Introduction

Consider the nonlinear equation

$$F(x) = 0, \quad (1)$$

where $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is continuous and monotone. F is monotone means

$$(F(x) - F(y))^T (x - y) \geq 0, \quad \forall x, y \in \mathbb{R}^n. \quad (2)$$

Monotone nonlinear equations arise in many applications, such as the subproblem in the generalized proximal algorithms with Bregman distances [35]. Some monotone variational inequality problems can also be converted into nonlinear monotone equations [13, 22]. Recently, monotone equations were used in signal and image recovery problems [16]. Iterative schemes for solving monotone equations have received high attention over the years.

Several methods have been proposed for system of nonlinear equations. For example, Newton's method and quasi-Newton methods (see in [2, 6, 8, 11, 12, 15, 17–19, 24, 26, 27, 34, 38, 39]) are attractive because of their locally fast convergence rates. Moreover, they are not suitable for solving monotone equations because they require computation of Jacobian matrix or an approximate of it, and solving linear system of equations in every iteration.

Spectral gradient method [4] is another class of method which is quite easy to implement and very efficient for large-scale unconstrained optimization problems. In [23], La Cruz and Raydan extended the spectral gradient method to solve large-scale system of nonlinear equations. They proposed the spectral algorithm called *Spectral algorithm for nonlinear equations* (SANE) in [23]. The global convergence is guaranteed by means of a variation of the nonmonotone strategy of Grippo et al. [21]. In [20], La Cruz et al. developed a fully derivative-free spectral algorithm for nonlinear equations called (DF-SANE) and the numerical examples given indicate that (DF-SANE) algorithm works well for a class of system of nonlinear equations. Zhang and Zhou [42] developed a spectral gradient algorithm by combining the spectral gradient method in [4] with the projection technique in [36]. Recently, Mohammad and Abubakar [1] proposed a combine form of spectral gradient method and the projection method by using the convex combination of two different spectral coefficients.

Conjugate gradient methods (CG) are also efficient for large-scale unconstrained optimization problems due to their simplicity and low storage [41]. For example, In 2001, Dai and Liao [10] introduces two conjugate gradient methods based on the modified conjugacy condition proposed by Perry [28]. This method was later modified by Kafaki and Gambari [3] in order to achieve a descent conjugate gradient method. Generally, conjugate gradient methods performs better than the spectral gradient method in practice. However, conjugate gradient methods for nonlinear equations are relatively rare in literature because most of them are for unconstrained optimization problems. In [7], Cheng combines the well-known Polak-Ribière-Polyak (PRP) conjugate gradient method [29, 33] and the hyperplane projection method for nonlinear monotone equations. Also, Papp and Rapajić [30] introduced some new Fletcher-Reeves (FR) [32] type directions in the frame of algorithm which

is a combination of conjugate gradient approach and hyperplane projection technique for large-scale nonlinear monotone equations. Furthermore, Yuan and Zhang [40] proposed three-term conjugate gradient method where the direction possesses the sufficient descent direction together with the projection technique for large-scale nonlinear equations. Recently in [37], Feng et al. introduce a family of conjugate gradient method combined with the projection technique for large-scale nonlinear equations.

Motivated by the works in [7, 30], we propose a descent conjugate gradient method for solving nonlinear equations. This method is an extension of the conjugate gradient proposed in [3] combined with the hyperplane projection method. The global convergence of the proposed method was proved under some assumptions. Moreover, numerical results presented indicate the efficiency of the method compared to that in [37].

The remaining part of the paper is organized as follows. Section 2 gives the proposed method and its algorithm. Section 3 provides the global convergence and Section 4 reports the numerical result.

2 Proposed method and its algorithm

In this section, we propose an extension of the descent Dai-Liao-type method in [3] to solve (1). The method is a combination of the descent Dai-Liao method for unconstrained optimization problems and the hyperplane projection technique.

Let x_0 be an initial point. An iterative scheme for solving (1) has the general form

$$x_{k+1} = x_k + s_k, \quad k = 0, 1, 2, \dots \quad (3)$$

where $s_k = \alpha_k d_k$, α_k is the step length obtained via some suitable line search and d_k is the search direction. For monotone equations, the process needs to be accelerated using the monotonicity condition of F . By monotonicity of F and letting $z_k = x_k + s_k$, the hyperplane $H_k = \{x \in \mathbb{R}^n | F(z_k)^T (x - z_k) = 0\}$ separates strictly x_k from the solution set of (1). Using this fact, Solodov and Svaiter [36] suggest that the next iterate x_{k+1} to be the projection of x_k onto the hyperplane H_k . So, x_{k+1} can be computed as

$$x_{k+1} = x_k - \frac{F(z_k)^T (x_k - z_k) F(z_k)}{\|F(z_k)\|^2}. \quad (4)$$

This projection method will be used to propose a descent Dai-Liao conjugate gradient method for solving (1).

Throughout the paper, we will assume F satisfy the following assumptions.

Assumption 1

(i) F is uniformly monotone, i.e.,

$$(F(x) - F(y))^T (x - y) \geq c \|x - y\|^2, \quad c \text{ is a positive real number.} \quad (5)$$

(ii) F is Lipschitz continuous, that is there exist a positive constant L such that

$$\|F(x) - F(y)\| \leq L\|x - y\|, \quad (6)$$

for all $x, y \in \mathbb{R}^n$.

Assumption (ii) implies there is a positive constant ω such that

$$\|F(x_k)\| \leq \omega. \quad (7)$$

In this paper, we propose the direction d_k to be

$$d_k = \begin{cases} -F(x_k), & \text{if } k = 0, \\ -F(x_k) + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \quad (8)$$

where β_k is the descent Dai-Liao conjugate gradient parameter given as

$$\beta_k = \frac{F(x_k)^T y_{k-1}}{y_{k-1}^T d_{k-1}} - t_k \frac{F(x_k)^T s_{k-1}}{y_{k-1}^T d_{k-1}}, \quad t_k = p \frac{\|y_{k-1}\|^2}{s_{k-1}^T y_{k-1}} - q \frac{s_{k-1}^T y_{k-1}}{\|s_{k-1}\|^2},$$

$$p \geq \frac{1}{4}, \quad q \leq 0, \quad y_k = F(x_{k+1}) - F(x_k), \quad s_k = z_k - x_k = \alpha_k d_k.$$

Notice that the CG parameter in (8) is a generalization of the two well known CG parameters for solving nonlinear equations. If $p = 2, q = 0$, then β_k will reduce to that of Hager and Zhang [14]. Also if $p = 1, q = 0$, β_k reduce to that of Dai and Kou [9].

Next, we state the algorithm of the proposed method called Dai-Liao projection method (DLPM).

Algorithm 1 Descent Dai-Liao projection method (DLPM)

Step 0. Given $x_0 \in \mathbb{R}^n, r, \sigma \in (0, 1)$, stopping tolerance $\epsilon > 0$, Set $k = 0$.

Step 1. Compute $F(x_k)$. If $\|F(x_k)\| \leq \epsilon$ stop, else go to **Step 2**.

Step 2. Compute d_k by (8). Stop if $d_k = 0$.

Step 3. Determine $\alpha_k = r^{m_k}$ with m_k being the smallest nonnegative integer m such that

$$-F(x_k + r^m d_k)^T d_k \geq \sigma r^m \|F(x_k + r^m d_k)\| \|d_k\|^2. \quad (9)$$

Step 4. Compute $z_k = x_k + \alpha_k d_k$. If $\|F(z_k)\| = 0$ stop, else go to **Step 5**.

Step 5. Compute x_{k+1} using (4).

Step 6. Let $k = k + 1$ and go to **Step 1**.

The next algorithm is a simplified version of the algorithm proposed by Feng et al. in [37].

Algorithm 2 A family of conjugate gradient method (FCG)

Step 0. Given an arbitrary initial point $x_0 \in \mathbb{R}^n$, parameters $0 < r < 1$, $\sigma > 0$, $t > 0$, $\rho > 0$, $\epsilon > 0$, and set $k := 0$.

Step 1. If $\|F(x_k)\| \leq \epsilon$, stop, otherwise go to **Step 2**.

Step 2. Compute

$$d_k = \begin{cases} -F(x_k), & \text{if } k = 0, \\ -(1 + \beta_k \frac{F(x_k)^T d_{k-1}}{\|F(x_k)\|^2}) F(x_k) + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases}$$

where β_k is such that

$$|\beta_k| = t \frac{\|F(x_k)\|}{\|d_{k-1}\|},$$

$\forall k \geq 1$ and $t > 0$.

Step 3. Find the trial point $y_k = x_k + \alpha_k d_k$, where $\alpha_k = \rho r^{m_k}$ and m_k is the smallest nonnegative integer m such that

$$-F(x_k + \rho r^m d_k)^T d_k \geq \sigma \rho r^m \|d_k\|.$$

Step 4. Compute x_{k+1} using (4).

Step 5. Let $k = k + 1$ and go to **Step 1**.

3 Global convergence

To prove the global convergence of Algorithm 1, the following preliminaries are needed.

Lemma 3.1 [3] *Let d_k be defined by (8), then*

$$F(x_k)^T d_k = -\lambda_k \|F(x_k)\|^2, \quad \lambda_k > 0, \quad \forall k \in \mathbb{N}. \quad (10)$$

Lemma 3.2 *If \bar{x} satisfy $F(\bar{x}) = 0$ and $\{x_k\}$ is generated by Algorithm 1. Then*

$$\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2 - \|x_{k+1} - x_k\|^2. \quad (11)$$

In particular $\{x_k\}$ is bounded and

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\| < \infty. \quad (12)$$

Lemma 3.3 *Let $\{x_k\}$ be generated by Algorithm 1, then*

$$\lim_{k \rightarrow \infty} \alpha_k \|d_k\| = 0. \quad (13)$$

Proof Lemma 3.2 implies that the sequence $\{\|x_k - x^*\|\}$ is non-increasing and convergent, therefore bounded. Also, $\{x_k\}$ is bounded and

$$\lim_{k \rightarrow \infty} \|x_{k+1} - x_k\| = 0. \quad (14)$$

From (10) and line search (8)

$$\begin{aligned} \|x_{k+1} - x_k\| &= \frac{|F(z_k)^T (x_k - z_k)|}{\|F(z_k)\|^2} \|F(z_k)\| \\ &= \frac{|\alpha_k F(z_k)^T d_k|}{\|F(z_k)\|} \\ &\geq \frac{\sigma \alpha_k^2 \|F(z_k)\| \|d_k\|^2}{\|F(z_k)\|} \\ &= \sigma \alpha_k^2 \|d_k\|^2 \geq 0. \end{aligned} \quad (15)$$

By (14) and (15), it follows that

$$\lim_{k \rightarrow \infty} \alpha_k \|d_k\| = 0. \quad (16)$$

□

Lemma 3.4 Suppose F is Lipschitz continuous and the sequence $\{x_k\}$ be generated by Algorithm 1 is bounded. Then there exist $M > 0$ such that $\|d_k\| \leq M, \forall k$.

Proof Using (5), (6) and (8), we get

$$|t_k| \leq p \frac{L^2 \|s_{k-1}\|^2}{c \|s_{k-1}\|^2} + |q| \frac{L \|s_{k-1}\|}{\|s_{k-1}\|^2} = p \frac{L^2}{c} + L|q|. \quad (17)$$

Also from (6), (7), (8) and (17), we obtain

$$\begin{aligned} \|d_k\| &= \|-F_k + \beta_k d_{k-1}\| \\ &= \|-F_k + \frac{F_k^T y_{k-1}}{s_{k-1}^T y_{k-1}} s_{k-1} - t_k \frac{F_k^T s_{k-1}}{s_{k-1}^T y_{k-1}} s_{k-1}\| \\ &\leq \|F_k\| + \frac{\|F_k\| \|y_{k-1}\|}{s_{k-1}^T y_{k-1}} \|s_{k-1}\| + |t_k| \frac{\|F_k\| \|y_{k-1}\|}{s_{k-1}^T y_{k-1}} \|s_{k-1}\| \\ &\leq \omega + \frac{L\omega \|s_{k-1}\|}{c \|s_{k-1}\|^2} \|s_{k-1}\| + (p \frac{L^2}{c} + L|q|) \frac{\omega \|s_{k-1}\|}{c \|s_{k-1}\|^2} \|s_{k-1}\| \\ &= \omega + \frac{L\omega}{c} + (p \frac{L^2}{c} + L|q|) \frac{\omega}{c}. \end{aligned} \quad (18)$$

Setting $M := \omega + \frac{L\omega}{c} + (p \frac{L^2}{c} + L|q|) \frac{\omega}{c}$, we establish the result. □

The following theorem establishes the global convergence of Algorithm 1.

Theorem 3.5 Let $\{x_k\}$ and $\{z_k\}$ be sequences generated by Algorithm 1. Then

$$\liminf_{k \rightarrow \infty} \|F(x_k)\| = 0. \quad (19)$$

Proof The proof will be divided into two cases;

Case I If $\liminf_{k \rightarrow \infty} \|d_k\| = 0$, we have $\liminf_{k \rightarrow \infty} \|F(x_k)\| = 0$. By continuity of F , the sequence $\{x_k\}$ has some accumulation point \tilde{x} such that $F(\tilde{x}) = 0$. Since $\{\|x_k - \tilde{x}\|\}$ converges and \tilde{x} is an accumulation point of $\{x_k\}$, it follows that $\{x_k\}$ converges to \tilde{x} .

Case II If $\liminf_{k \rightarrow \infty} \|d_k\| > 0$, we have $\liminf_{k \rightarrow \infty} \|F(x_k)\| > 0$. By (13), it holds that $\lim_{k \rightarrow \infty} \alpha_k = 0$.

Using the line search (9), $-F(x_k + \beta^{m_{k-1}} d_k)^T d_k < \sigma \beta^{m_{k-1}} \|F(x_k + \beta^{m_{k-1}} d_k)\| \|d_k\|^2$ and the boundedness of $\{x_k\}$, $\{d_k\}$, we can choose a subsequence such that allowing k to go to infinity in the above inequality results

$$F(\tilde{x})^T \tilde{d} > 0. \quad (20)$$

On the other hand, allowing k to approach ∞ in (9), implies

$$F(\tilde{x})^T \tilde{d} \leq 0. \quad (21)$$

Equations (20) and (21) indicate a contradiction. Therefore, $\liminf_{k \rightarrow \infty} \|F(x_k)\| > 0$ does not hold and the proof is complete. \square

Corollary 3.6 Let $\{x_k\}$ and $\{z_k\}$ be sequences generated by Algorithm 2 [37]. Then

$$\liminf_{k \rightarrow \infty} \|F(x_k)\| = 0. \quad (22)$$

Proof The proof follows from Theorem 3.5 because it is independent of the definition of d_k . \square

4 Numerical experiment

This section reports some numerical result of the proposed method (DLPM) compared to (FCG) algorithm [37]. All codes were written in MATLAB R2017a and run on a PC with intel COREi5 processor with 4GB of RAM and CPU 2.3GHZ. We test problems 1 to 8 with different initial points and dimensions of 10000 for some initial points and 100000 for others.

In DLPM algorithm, we set $\sigma = 0.01$, $r = 0.6$, $p = 0.8$, $q = -0.1$, while in FCG [37], $\sigma = 0.01$, $r = 0.5$, $\rho = 1$, $t = 1$. All runs were stopped whenever $\|F(x_k)\|_{\infty} < 10^{-6}$.

In Tables 1, 2, 3, 4, 5, 6, 7, and 8, the number of iterations (ITER) needed to converge to an approximate solution, the CPU time in seconds (TIME), the number of function evaluations (Feval) and the norm of the objective function F at the approximate solution x^* (Norm) for problems 1–8 were presented.

Table 9 summarizes the results obtained from Tables 1, 2, 3, 4, 5, 6, 7, and 8 based on which method is a winner in terms of CPU time (TIME), number of iterations (ITER), and number of function evaluations (Feval). It can be observed from Table 9 that DLPM is the most efficient in terms of number of iterations because it solve and wins about 95.7% with less number of iterations than FCG (4.3%). Also, DLPM method solve and wins about 68% of the total problems within shorter time than FCG (32%).

Inaddition, we use the performance profile of Dolan and Moré [25] to support the results in Table 9. Figures 1, 2 and 3 show the performance profile of DLPM and

Table 1 Numerical comparison of DLPM and FCG methods for Problem 1

Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
1	$1 * \mathbf{ones}(n, 1)$	10000	9	27	0.032497	0.18069	27	81	6.54E-07	6.03E-07
	$-0.5 * \mathbf{ones}(n, 1)$	10000	9	14	0.073868	0.05479	27	42	3.65E-07	5.84E-07
	$0.1 * \mathbf{ones}(n, 1)$	10000	9	24	0.03197	0.105991	27	72	1.79E-07	5.95E-07
	$-10 * \mathbf{ones}(n, 1)$	10000	19	15	0.222063	0.052713	57	45	1.52E-07	3.08E-07
	$-1 * \mathbf{ones}(n, 1)$	100000	13	16	1.018153	0.487464	39	48	1.17E-07	2.60E-07
	$0.5 * \mathbf{ones}(n, 1)$	100000	10	28	0.313869	0.698384	30	84	6.10E-07	5.58E-07
	$-0.1 * \mathbf{ones}(n, 1)$	100000	9	14	0.713879	0.426649	27	42	1.33E-07	3.54E-07
	$10 * \mathbf{ones}(n, 1)$	100000	25	30	2.229709	0.747827	75	90	5.05E-07	9.28E-07

Table 2 Numerical comparison of DLPM and FCG methods for Problem 2

Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
2	$1 * \mathbf{ones}(n, 1)$	10000	64	124	0.941832	0.537937	192	372	8.59E-07	7.58E-07
	$-0.5 * \mathbf{ones}(n, 1)$	10000	71	262	0.913216	1.130091	213	786	7.50E-07	9.87E-07
	$0.1 * \mathbf{ones}(n, 1)$	10000	64	191	0.836011	0.827377	192	573	7.30E-07	8.24E-07
	$0.5 * \mathbf{ones}(n, 1)$	100000	75	202	8.866332	7.745228	225	606	7.62E-07	9.91E-07
	$-0.1 * \mathbf{ones}(n, 1)$	100000	77	240	9.687362	9.016119	231	720	6.00E-07	6.76E-07

Table 3 Numerical comparison of DLPM and FCG methods for Problem 3

Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
3	$1 * \mathbf{ones}(n, 1)$	10000	12	28	0.078181	0.154395	36	84	4.91E-07	6.40E-07
	$-0.5 * \mathbf{ones}(n, 1)$	10000	14	29	0.101245	0.171046	42	87	3.95E-07	5.99E-07
	$0.1 * \mathbf{ones}(n, 1)$	10000	13	28	0.086709	0.156555	39	84	5.41E-07	9.75E-07
	$-10 * \mathbf{ones}(n, 1)$	10000	24	31	0.3646	0.156339	72	93	5.94E-07	5.92E-07
	$-1 * \mathbf{ones}(n, 1)$	100000	21	31	2.77424	1.455615	63	93	9.60E-07	5.48E-07
	$0.5 * \mathbf{ones}(n, 1)$	100000	16	30	1.503665	1.423351	48	90	5.45E-07	6.53E-07
	$-0.1 * \mathbf{ones}(n, 1)$	100000	18	30	1.955514	1.39973	54	90	5.52E-07	8.30E-07
	$10 * \mathbf{ones}(n, 1)$	100000	30	32	4.632616	1.457045	90	96	2.85E-07	5.36E-07

Table 4 Numerical comparison of DLPM and FCG methods for Problem 4

Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
4	$1 * \mathbf{ones}(n, 1)$	10000	9	25	0.035215	0.05842	27	75	4.61E-07	7.50E+01
	$-0.5 * \mathbf{ones}(n, 1)$	10000	10	22	0.026366	0.05214	30	66	1.51E-07	5.25E-07
	$0.1 * \mathbf{ones}(n, 1)$	10000	9	24	0.025049	0.0597	27	72	1.32E-07	5.39E-07
	$-10 * \mathbf{ones}(n, 1)$	10000	21	32	0.056799	0.073275	63	96	2.67E-07	5.36E-07
	$-1 * \mathbf{ones}(n, 1)$	100000	13	23	0.358432	0.463394	39	69	1.08E-07	5.34E-07
	$0.5 * \mathbf{ones}(n, 1)$	100000	10	27	0.257967	0.528792	30	81	9.87E-07	6.93E-07
	$-0.1 * \mathbf{ones}(n, 1)$	100000	9	22	0.185913	0.428291	27	66	1.41E-07	7.33E-07

Table 5 Numerical comparison of DLPM and FCG methods for Problem 5

Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
5	$1 * \mathbf{ones}(n, 1)$	10000	17	192	0.202546	0.720841	51	576	5.29E-07	9.64E-07
	$-0.5 * \mathbf{ones}(n, 1)$	10000	20	201	0.216263	0.733857	60	603	4.67E-07	7.72E-07
	$0.1 * \mathbf{ones}(n, 1)$	10000	20	198	0.231259	0.731433	60	594	6.48E-07	9.59E-07
	$-10 * \mathbf{ones}(n, 1)$	10000	23	270	0.283339	0.976731	69	810	7.48E-07	9.64E-07
	$-1 * \mathbf{ones}(n, 1)$	100000	18	189	1.906852	6.363033	54	567	5.46E-07	9.84E-07
	$0.5 * \mathbf{ones}(n, 1)$	100000	16	192	1.608839	6.387715	48	576	6.84E-07	9.35E-07
	$-0.1 * \mathbf{ones}(n, 1)$	100000	18	195	1.809536	6.538223	54	585	9.29E-07	9.80E-07
	$10 * \mathbf{ones}(n, 1)$	100000	42	250	4.548315	8.452533	126	750	8.63E-07	7.08E-07

Table 6 Numerical comparison of DLPM and FCG methods for Problem 6

Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
6	$1 * \mathbf{ones}(n, 1)$	10000	10	20	0.494183	0.084542	30	60	8.02E-07	5.78E-07
	$-0.5 * \mathbf{ones}(n, 1)$	10000	9	25	0.044261	0.092218	27	75	3.69E-07	7.80E-07
	$0.1 * \mathbf{ones}(n, 1)$	10000	8	20	0.041824	0.068603	24	60	4.36E-07	9.02E-07
	$0.5 * \mathbf{ones}(n, 1)$	100000	11	23	0.391894	0.715981	33	69	1.64E-07	6.37E-07
	$-0.1 * \mathbf{ones}(n, 1)$	100000	9	25	0.343348	0.804374	27	75	4.13E-07	8.50E-07
	$10 * \mathbf{ones}(n, 1)$	100000	44	26	4.66066	0.807222	132	78	3.58E-07	6.42E-07

Table 7 Numerical comparison of DLPM and FCG methods for Problem 7

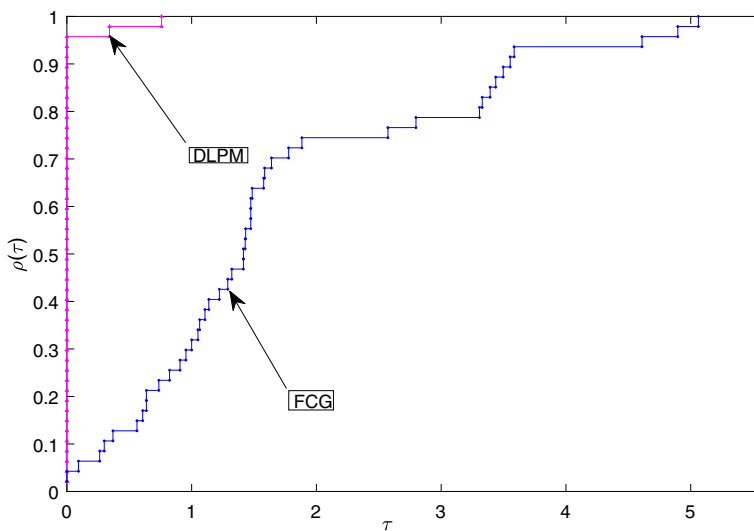
Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
7	$1 * \mathbf{ones}(n, 1)$	10000	28	834	0.414038	3.5541	84	2502	$9.45\text{E}-07$	$7.54\text{E}-07$
	$-1 * \mathbf{ones}(n, 1)$	10000	25	835	0.362725	3.7921	75	2505	$5.73\text{E}-07$	$7.98\text{E}-07$
	$-0.1 * \mathbf{ones}(n, 1)$	100000	24	586	2.789953	22.28455	72	1758	$6.43\text{E}-07$	$9.55\text{E}-07$

Table 8 Numerical comparison of DLPM and FCG methods for Problem 8

Problem	Initial point	n	ITER		TIME		Feval		Norm	
			DLPM	FCG	DLPM	FCG	DLPM	FCG	DLPM	FCG
8	$0.1 * \mathbf{ones}(n, 1)$	10000	35	120	3.941798	4.076348	105	360	$8.21\text{E}-07$	$7.89\text{E}-07$
	$-0.1 * \mathbf{ones}(n, 1)$	100000	44	306	50.2077	124.3842	132	918	$7.24\text{E}-07$	$7.99\text{E}-07$

Table 9 Winner with respect to number of iterations, CPU time and number of function evaluations

Winner	DLPM	FCG
ITER	45	2
TIME	32	15
Feval	45	2

**Fig. 1** Performance profile with respect to number of iterations

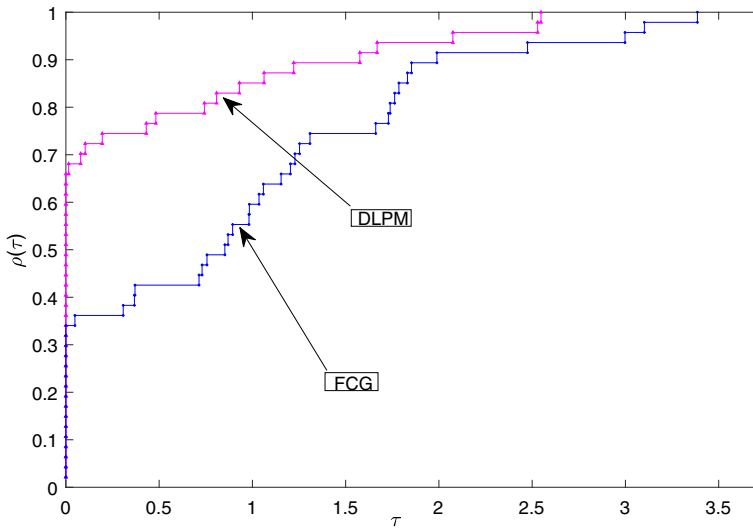


Fig. 2 Performance profile with respect to CPU time

FCG methods with respect to number of iterations, CPU time, and number of function evaluations respectively. The figures show the efficiency of the DLPM method because the curve that stays longer on the y-axis corresponds to it.

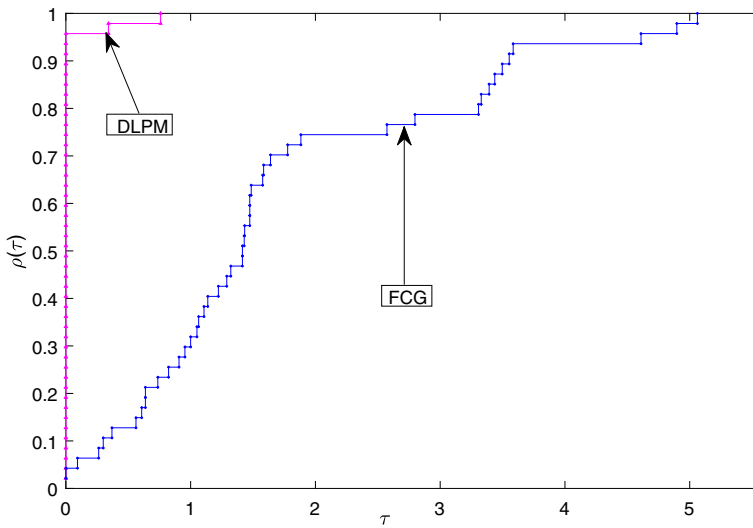


Fig. 3 Performance profile with respect to number of function evaluations

The test functions $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$, where $x = (x_1, x_2, \dots, x_n)^T$, are listed as follows:

Problem 1 [43]

$$F_i(x) = 2x_i - \sin |x_i|, \quad i = 1, 2, 3, \dots, n.$$

Problem 2 [43]

$$F_1(x) = 2x_1 + \sin(x_1) - 1,$$

$$F_i(x) = -2x_{i-1} + 2x_i + \sin(x_i) - 1, \quad \text{for } i = 2, 3, \dots, n-1,$$

$$F_n(x) = 2x_n + \sin(x_n) - 1.$$

Problem 3 Tridiagonal Exponential Problem [5]

$$F_1(x) = x_1 - e^{\cos(h(x_1+x_2))}$$

$$F_i(x) = x_i - e^{\cos(h(x_{i-1}+x_i+x_{i+1}))} \quad \text{for } i = 2, 3, \dots, n-1$$

$$F_n(x) = x_n - e^{\cos(h(x_{n-1}+x_n))},$$

$$\text{where } h = \frac{1}{n+1}.$$

Problem 4 [31]

$$F_i(x) = e^{x_i} - 1, \quad \text{for } i = 1, 2, 3, \dots, n.$$

Problem 5

$$F_1(x) = 2.5x_1 + x_2 - 1$$

$$F_i(x) = x_{i-1} + 2.5x_i + x_{i+1} - 1 \quad \text{for } i = 2, 3, \dots, n-1$$

$$F_n(x) = x_{n-1} + 2.5x_n - 1.$$

Problem 6 Logarithmic Function

$$F_i(x) = \ln(x_i + 1) - \frac{x_i}{n}, \quad \text{for } i = 2, 3, \dots, n.$$

Problem 7 [43]

$$F_1(x) = x_1(x_1^2 + x_2^2) - 1$$

$$F_i(x) = x_i(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2) - 1 \quad \text{for } i = 2, 3, \dots, n-1$$

$$F_n(x) = x_n(x_{n-1}^2 + x_n^2).$$

Problem 8 [44]

$$F(x) = \begin{pmatrix} 2 & -1 & \\ -1 & 2 & -1 \\ \ddots & \ddots & \ddots \\ \ddots & \ddots & -1 \\ & -1 & 2 \end{pmatrix} x + (e^{x_1} - 1, \dots, e^{x_n} - 1)^T.$$

5 Conclusions

In this paper, we proposed a descent Dai-Liao conjugate gradient method for solving nonlinear equations. DLPM can be regarded as an extension of the descent Dai-Liao method by Kafaki and Gambari [3] for unconstrained optimization combine with the hyperplane projection strategy. Under mild assumptions, the method was proved to be globally convergent. Numerical comparison shows that the proposed method is effective and promising because it converges faster compared to recent algorithm.

Acknowledgements The authors are grateful to the referees for their helpful suggestions which provide an improvement of the paper.

Funding information This study was funded by the King Mongkut's University of Technology Thonburi through the "KMUTT 55th Anniversary Commemorative Fund." The first author was supported by the Petchra Pra Jom Klao Doctoral Scholarship Academic for Ph.D. Program at KMUTT. This project is supported by the Theoretical and Computational Science (TaCS) Center under Computational and Applied Science for Smart Innovation (CLASSIC), Faculty of Science, KMUTT.

References

1. Abubakar, A.B., Mohammad, H.: A positive spectral gradient-like method for large-scale nonlinear monotone equations. *Bull. Comput. Appl. Math.* **5**(1), 97–113 (2017)
2. Al-Baali, M., Spedicato, E., Maggioni, F.: Broyden's quasi-newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems. *Optim. Meth. Soft.* **29**(5), 937–954 (2014)
3. Babaie-Kafaki, S., Gambari, R.: A descent family of Dai-Liao conjugate gradient methods. *Optim. Meth. Soft.* **29**(3), 583–591 (2014)
4. Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. *IMA J. Numer. Anal.* **8**(1), 141–148 (1988)
5. Bing, Y., Lin, G.: An efficient implementation of merrill's method for sparse or partially separable systems of nonlinear equations. *SIAM J. Optim.* **1**(2), 206–221 (1991)
6. Broyden, C.G.: A class of methods for solving nonlinear simultaneous equations. *Math. Comput.* **19**, 577–593 (1965)
7. Cheng, W.: A prp type method for systems of monotone equations. *Math. Comput. Model.* **50**, 15–20 (2009)
8. Cordero, A., Torregrosa, J.R.: Variants of Newton's method for functions of several variables. *Appl. Math. Comput.* **183**, 199–208 (2006)
9. Dai, Y.-H., Kou, C.-X.: A nonlinear conjugate gradient algorithm with an optimal property and an improved wolfe line search. *SIAM J. Optim.* **23**(1), 296–320 (2013)
10. Dai, Y.H., Liao, L.Z.: New conjugacy conditions and related nonlinear conjugate gradient methods. *Appl. Math. Optim.* **43**(1), 87–101 (2001)
11. Dembo, R.S., Eisenstat, S.C., Steihaug, T.: Inexact Newton methods. *SIAM J. Numer. Anal.* **19**(2), 400–408 (1982)
12. Dirvashi, M.T., Barati, A.: A third-order Newton-type method to solve systems of nonlinear equations. *Appl. Math. Comput.* **187**, 630–635 (2007)
13. Fukushima, M.: Equivalent differentiable optimization problems and descent methods for asymmetric variational inequality problems. *Math. Program.* **53**(1–3), 99–110 (1992)
14. Hager, W.W., Zhang, H.: Algorithm 851: Cg_descent, a conjugate gradient method with guaranteed descent. *ACM Trans. Math. Softw. (TOMS)* **32**(1), 113–137 (2006)
15. Hassan, M., Waziri, M.Y.: On broyden-like update via some quadratures for solving nonlinear systems of equations. *Turk. J. Math.* **39**(3), 335–345 (2015)

16. Hu, Q., Xiao, Y., Wang, Q.: Non-smooth equations based methods for l_1 -norm problems with applications to compressed sensing. *Nonlinear Anal.* **74**, 3570–3577 (2011)
17. Kelly, C.T.: *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia (1995)
18. Kelly, C.T.: *Solving Nonlinear Equations with Newton's Method*. SIAM, Philadelphia (2003)
19. Keyes, D., Knoll, D.: Jacobian-free Newton-Krylov methods: a survey of approaches and applications. *J. Comput. Phys.* **193**, 357–397 (2004)
20. La Cruz, W., Martínez, J., Raydan, M.: Spectral residual method without gradient information for solving large-scale nonlinear systems of equations. *Math. Comput.* **75**(255), 1429–1448 (2006)
21. Lampariello, F., Lucidi, S., Grippo, L.: A nonmonotone line search technique for newton's method. *SIAM J. Numer. Anal.* **23**(4), 707–716 (1986)
22. Li, D., Zhao, Y.-B.: Monotonicity of fixed point and normal mappings associated with variational inequality and its application. *SIAM J. Optim.* **11**(4), 962–973 (2001)
23. Marcos, R., La Cruz, W.: Nonmonotone spectral methods for large-scale nonlinear systems. *Optim. Meth. Soft.* **18**(5), 583–599 (2003)
24. Martínéz, J.M.: Practical quasi-newton methods for solving nonlinear systems. *J. Comput. Appl. Math.* **124**, 97–121 (2000)
25. Moré, J.J., Dolan, E.D.: Benchmarking optimization software with performance profiles. *Math. Program., Ser.* **91**, 201–213 (2002)
26. Moré, J.J., Trangenstein, J.A.: On the global convergence of Broyden's method. *Math. Comput.* **30**, 523–540 (1976)
27. Natasa, K., Zorana, L.: Newton-like method with modification of the right-hand-side vector. *Math. Comput.* **71**, 237–250 (2002)
28. Perry, A.: A modified conjugate gradient algorithm. *Oper. Res.* **26**(6), 1073–1078 (1978)
29. Polyak, B.T.: The conjugate gradient method in extreme problems. *USSR Comp. Math. Math. Phys.* **9**, 94–112 (1969)
30. Rapajić, S., Papp, Z.: Fr type methods for systems of large-scale nonlinear monotone equations. *Appl. Math. Comput.* **269**, 816–823 (2015)
31. Raydan, M.: The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM J. Optim.* **7**(1), 26–33 (1997)
32. Reeves, C.M., Fletcher, R.: Function minimization by conjugate gradients. *Comput. J.* **7**(2), 149–154 (1964)
33. Ribière, G., Polak, E.: Note sur la convergence de directions conjuguées. *Rev. Francaise Informat Recherche Opertionelle* **16**, 35–43 (1969)
34. Shin, B.C., Dirvashi, M.T., Kim, C.H.: A comaprison of the Newton-Krylov method with high order Newton-like methods to solve nonlinear systems. *Appl. Math. Comput.* **217**, 3190–3198 (2010)
35. Solodov, V.M., Iusem, A.N.: Newton-type methods with generalized distances for constrained optimization. *Optimization* **41**(3), 257–278 (1997)
36. Solodov, M.V., Svaiter, B.F.: A globally convergent inexact Newton method for systems of monotone equations. In: *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pp 355–369. Springer (1998)
37. Sun, M., Wang, X., Feng, D.: A family of conjugate gradient methods for large-scale nonlinear equations. *Journal of Inequalities and Applications* **236**, 1–8 (2017)
38. Waziri, M.Y., Leong, W.J., Hassan, M.A.: Diagonal Broyden-like method for large-scale systems of nonlinear equations. *Malaysian Journal of Mathematical Sciences* **6**(1), 59–73 (2012)
39. Waziri, M.Y., Leong, W.J., Hassan, M.A., Monsi, M.: A new Newton's method with diagonal Jacobian approximation for systems of nonlinear equations. *J. Math. Stat.* **6**(3), 246–252 (2010)
40. Yuan, G., Zhang, M.: A three-term Polak-Rebiere-Poylak conjugate gradient algorithm for large-scale nonlinear equations. *J. Comput. Appl. Math.* **286**, 186–195 (2015)
41. Zhang, H., Hager, W.W.: A survey of nonlinear conjugate gradient methods. *Pac. J. Optim.* **2**(1), 35–58 (2006)
42. Zhang, L., Zhou, W.: Spectral gradient projection method for solving nonlinear monotone equations. *J. Comput. Appl. Math.* **196**(2), 478–484 (2006)
43. Zhou, W., Li, D.: Limited memory bfgs method for nonlinear monotone equations. *J. Comput. Math.* **25**(1), 89–96 (2007)
44. Zhou, W.-J., Li, D.-H.: A globally convergent bfgs method for nonlinear monotone equations without any merit functions. *Math. Comput.* **77**(264), 2231–2240 (2008)