SeMA Journal
https://doi.org/10.1007/s40324-020-00228-9

Enhanced Dai–Liao conjugate gradient methods for systems of monotone nonlinear equations

M. Y. Waziri[1] · K. Ahmed[1] · J. Sabi'u[2] · A. S. Halilu[3]

Received: 15 March 2020 / Accepted: 25 July 2020

Abstract
In this paper, we propose two conjugate gradient methods for solving large-scale monotone nonlinear equations. The methods are developed by combining the hyperplane projection method by Solodov and Svaiter (Reformulation: nonsmooth, piecewise smooth, semismooth and smoothing methods. Springer, pp 355–369, 1998) and two modified search directions of the famous Dai and Liao (Appl Math Optim 43(1): 87–101, 2001) method. It is shown that the proposed schemes satisfy the sufficient descent condition. The global convergence of the methods are established under mild conditions, and computational experiments on some benchmark test problems show that the methods are promising.

Keywords Nonlinear equations · Eigenvalue analysis · Hyperplane projection · Monotonicity property · Global convergence

Mathematics Subject Classification 90C30 · 65K05 · 90C53 · 49M37 · 15A18

1 Introduction

Due to the important role they play in many areas of human endeavors such as sciences, engineering and industry, systems of nonlinear equations have gained enormous attention of researchers in recent decades and various examples have been considered in this areas.

This interest of this article is on system of nonlinear equations of the form

$$F(x) = 0, \quad \text{subject to} \quad x \in \mathbf{R}^n, \tag{1.1}$$

where $F : \mathbf{R}^n \rightarrow \mathbf{R}^n$ is a continuously differentiable mapping in the neighborhood of $\mathbf{R}^n$. The general iterative approach for solving (1.1) is

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, \ldots, \tag{1.2}$$

✉ M. Y. Waziri
mywaziri.mth@buk.edu.ng

1 Department of Mathematical Sciences, Faculty of Science, Bayero University, Kano, Kano, Nigeria

2 Department of Mathematics, Faculty of Sciences, Yusuf Maitama Sule University, Kano, Kano, Nigeria

3 Department of Mathematics and Computer Science, Sule Lamido University, Kafin Hausa, Nigeria

where x_{k+1} is the next iterative point, α_k is the step length to be obtained via line search procedure, and d_k is the search direction. In addition, F is said to be monotone if

$$(F(x) - F(y))^T(x - y) \geq 0, \forall x, y \in \mathbf{R}^n. \tag{1.3}$$

Systems of nonlinear monotone equations have appeared in various applications, for example, the Chandrsekhar integral equations, which arise in radiactive transfer and transport theory [30] is discretized and expressed as (1.1). Monotone nonlinear systems are also used as subproblems in the generalized proximal algorithms with Bregman distances [47]. By means of fixed point mappings or normal mappings [71], some monotone variational inequality problems can also be converted into nonlinear monotone equations. Monotone equations can also be reformulated as some ℓ_1-*norm* regularized optimization problems in compressive sensing [36,57]. For more examples of the method's applications, we refer the reader to [42,71]. Some iterative methods for solving these problems include Newton and quasi-Newton schemes [9,14,31,54], the Gauss–Newton methods [16,31], the Levenberg–Marquardt methods [25,28,40], the derivative-free methods [55], the subspace methods [64], the tensor methods [8], and the trust-region methods [51,65,69].

Conjugate gradient (CG) methods represent an ideal choice for mathematicians and engineers engaged in large-scale problems because of their low memory requirement and strong global convergence properties [5,37]. Generally, the nonlinear CG method is used to solve large-scale problems in the following form;

$$\min_{x \in \mathbf{R}^n} f(x), \tag{1.4}$$

where $f : \mathbf{R}^n \longrightarrow \mathbf{R}$ is a continuously differentiable function that is bounded from below and its gradient is available. The method generates a sequence of iterates x_k from an initial starting point $x_0 \in \mathbf{R}^n$ using the iterative formula

$$x_{k+1} = x_k + s_k, \quad s_k = \alpha_k d_k, k = 0, 1, \ldots, \tag{1.5}$$

where x_k is the current iterate, α_k is a step length computed via suitable line search procedure, and d_k is the CG search direction defined by

$$\begin{aligned} d_0 &= -F_0, \\ d_k &= -F_k + \beta_k d_{k-1}, \quad k = 1, 2, \ldots, \end{aligned} \tag{1.6}$$

and β_k is a scalar, with $F_k = \nabla f(x_k)$. A vital factor in CG algorithm is the parameter β_k [1] as it influences numerical performance of the scheme. In past decades, various CG methods with different choices of β_k in 1.6 have been presented (see [12,41,49] and the review paper [21]). These methods are different in terms of efficiency and convergence properties [12,21,41].

One of the most effective CG algorithms was proposed by Dai and Liao (DL) [11] with the following formula for β_k:

$$\beta_k^{DL} = \frac{(y_{k-1} - ts_{k-1})^T F_k}{d_{k-1}^T y_{k-1}}, t \geq 0, \tag{1.7}$$

where $y_{k-1} = F_k - F_{k-1}, s_{k-1} = x_k - x_{k-1} = \alpha_{k-1} d_{k-1}$.

The DL method has been shown to be numerically effective, but it fails to satisfy sufficient descent [5], which is required for global convergence of CG methods. That is, the method may not satisfy the sufficient descent condition

$$F_k^T d_k \leq -\lambda \|F_k\|^2, \quad \forall k. \tag{1.8}$$

Also, in [1], Andrei noted the DL method depends so much on the nonnegative parameter t, which has no optimal value. In recent years, however, DL methods with optimal parameter choices have been presented by researchers (see [6,7,17]). Furthermore, in order to develop methods that possess good computational efficiency as well as strong convergence properties, researchers applied the DL approach to propose CG methods based on modified secant equations [3,4,18,19,26,33,38,39,56,58,67,68,75]. However, like the DL method, these methods also fail to ensure sufficient descent.

Recently, by employing Perry's idea [44], efficient CG methods with descent directions have been proposed. Liu and Shang [34] proposed a Perry conjugate gradient method, which provides prototypes for developing other special forms of the Perry method like the Hestenes–Stiefel (HS) [22] method and the DL method [11]. Liu and Xu [35] presented a new Perry CG method with sufficient descent properties, which is independent of any line search. Also, based on the self-scaling memoryless BFGS update, Andrei [2] proposed an accelerated adaptive class of Perry CG algorithms, whose search direction is determined by symmetrization of the scaled Perry CG direction [44].

In recent years, CG methods for unconstrained optimization have been extended to large-scale nonlinear systems of equations. Using a combination of the Polak–Ribieré–Polyak (PRP) method [45,46] and the hyperplane projection technique [48], Cheng [10] proposed a PRP-type method for systems of monotone equations. Yu [60,61] extended the PRP method [45,46] to solve large-scale nonlinear systems with monotone line search strategies, which are modifications of the Grippo et al. [20] and Li-Fukushima [32] schemes. As a further research of the Perry CG method, Dai et al. [13] combined the modified Perry CG method [37] and the hyperplane projection technique of Solodov and Svaiter [48] to propose a derivative-free method for solving large-scale nonlinear monotone equations. Also, by replacing the gradients of the unmodified PRP method [45,46] with the residuals combined with the hyperplane projection technique, Zhou and Wang [74] presented a derivative-free residual method for large-scale monotone nonlinear equations, which may also be nonsmooth. Waziri et al. [52] proposed a family of Hager–Zhang conjugate gradient methods for nonlinear monotone equations by combining two modified Hager–Zhang schemes with the hyperplane projection technique in [48]. Recently, Waziri et al. [53] proposed a DL CG method via modified secant equation for systems of nonlinear equations. The method was shown to satisfy the sufficient descent condition and its global convergence is also established under some mild conditions. This article intends is to add to the limited pool of CG methods for solving large-scale monotone nonlinear systems as most of the ones developed over the decades are for unconstrained optimization problem.

Here, motivated by [52,53], the classical DL method [11], and the hyperplane projection technique [48], we propose extensions of [11] for solving large-scale monotone nonlinear equations.

The remaining sections of the paper are arranged as follows. Section 2 deals with preliminaries and derivation of the methods. Convergence analysis is presented in Sect. 3, while numerical results and their discussions are given in Sect. 4. Finally, concluding remarks are made in Sect. 5.

2 Preliminaries

In this section, two new DL-type conjugate gradient methods are presented by incorporating a spectral coefficient and an extended quasi-Newton condition in the classical DL method.

We begin by recalling the hyperplane projection method in [48]. The technique generates a sequence of iterates $\{z_k\}$ such that $z_k = x_k + \alpha_k d_k$, where $\alpha_k > 0$ is a step size computed via some suitable line search technique along the direction d_k such that

$$F(z_k)^T(x_k - z_k) > 0. \tag{2.1}$$

Also, by monotonicity of F, we have

$$F(z_k)^T(x^* - z_k) = (F(z_k) - F(x^*))^T(x^* - z_k) \leq 0, \tag{2.2}$$

for every solution x^* of the system in (1.1). Thus, from (2.1) and (2.2) it's clear that the hyperplane

$$H_k = \{x \in \mathbf{R}^n | F(z_k)^T(x - z_k) = 0\}, \tag{2.3}$$

strictly separates the current iterate x_k from the solution x^* of the system in (1.1). Solodov and Svaiter [48] used this fact to suggest the next iterate as the projection of x_k onto the hyperplane H_k. And the next iterate is computed by

$$x_{k+1} = x_k - \frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|^2} F(z_k). \tag{2.4}$$

We shall employ the above strategy to develop algorithms of our proposed methods for solving (1.1).

Throughout this paper, $\|.\|$ represents the Euclidean norm, $F_k = F(x_k)$, $F_{k-1} = F(x_{k-1})$, and f in (1.4) is specified by

$$f(x) := \frac{1}{2}\|F(x)\|^2. \tag{2.5}$$

In addition, we assume the function F is uniformly monotone, i.e.,

$$(F(x) - F(y))^T(x - y) \geq m\|x - y\|^2, m \text{ is a positive constant.} \tag{2.6}$$

Next, we recall that for quasi-Newton methods, a square matrix B_{k-1} approximates the Hessian $\nabla^2 f(x_{k-1})$ and is updated by the matrix B_k such that

$$B_k s_{k-1} = y_{k-1}, \tag{2.7}$$

known as the standard secant condition is satisfied. Zhang et al. [67] and Zhang and Xu [68] expanded (2.7) and gave the following extension

$$B_k s_{k-1} = \hat{z}_{k-1}, \tag{2.8}$$

with

$$\begin{cases} \hat{z}_{k-1} = y_{k-1} + \left(\frac{\vartheta_{k-1}}{s_{k-1}^T \mu_{k-1}}\right)\mu_{k-1}, \\ \vartheta_{k-1} = 6(f_{k-1} - f_k) + 3s_{k-1}^T(F_{k-1} + F_k), \end{cases} \tag{2.9}$$

where $\mu_{k-1} \in \mathbf{R}^n$ is a vector parameter such that $s_{k-1}^T \mu_{k-1} \neq 0$ (see [67]). In [38], it was noted that the modified secant equation proposed in (2.8) is superior to the one given in (2.7) as the former gives a better approximation of the Hessian than y_{k-1}.

Similarly, Wei et al. [56] gave the following extension of the standard secant equation (2.7)

$$B_k s_{k-1} = \bar{z}_{k-1}, \tag{2.10}$$

with

$$\begin{cases} \bar{z}_{k-1} = y_{k-1} + \left(\frac{\varsigma_{k-1}}{s_{k-1}^T \mu_{k-1}}\right) \mu_{k-1}, \\ \varsigma_{k-1} = 2(f_{k-1} - f_k) + s_{k-1}^T (F_{k-1} + F_k), \end{cases} \tag{2.11}$$

where $\mu_{k-1}\mathbf{R}^n$ is a vector parameter such that $s_{k-1}^T \mu_{k-1} \neq 0$ (see [63]). Here, in order to exploit the theoretical advantage of the above modified secant equations, we propose the following as an extension of (2.7), (2.8), and (2.10):

$$B_k s_{k-1} = u_{k-1} = y_{k-1} + \xi \frac{\varsigma_{k-1}}{s_{k-1}^T \mu_{k-1}} \mu_{k-1}, \tag{2.12}$$

where $\xi \in (0, 3)$, ς_{k-1} is as defined by (2.11) and $s_{k-1}^T \mu_{k-1} \neq 0$. We observe that for $\xi = 0$, (2.12) reduces to (2.7), and if $\xi = 1$, (2.12) reduces to (2.10). Also, for $\xi = 3$, we see that (2.12) reduces to the modified secant equation (2.8).

Recently, by employing a modified version of the classical spectral coefficient θ_k combined with a modified Dai–Liao conjugacy condition, Sun et. al. [50] proposed two modified spectral conjugate gradient methods for unconstrained optimization with the following conjugate gradient parameters:

$$\beta_k^{1*}(t) = \frac{(\theta_k^{1*} y_{k-1}^* - t s_{k-1})^T F_k}{d_{k-1}^T y_{k-1}^*}, \ (t \geq 0), \tag{2.13}$$

where θ_k^{1*} is the modified spectral coefficient defined by

$$\theta_k^{1*} = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}^*}, \tag{2.14}$$

and

$$y_{k-1}^* = y_{k-1} + \frac{\varsigma_{k-1}}{\|s_{k-1}\|^2} s_{k-1}, \tag{2.15}$$

with ς_{k-1} as defined in (2.11). By redefining y_{k-1}^* to avoid a situation where $d_{k-1}^T y_{k-1}^* = 0$, the following parameter is given

$$\beta_k^{2*}(t) = \frac{(\theta_k^{2*} z_{k-1}^* - t s_{k-1})^T F_k}{d_{k-1}^T z_{k-1}^*}, \ (t \geq 0), \tag{2.16}$$

where

$$\theta_k^{2*} = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T z_{k-1}^*}, \tag{2.17}$$

is a second spectral coefficient with z_{k-1} given by

$$z_{k-1}^* = y_{k-1} + \frac{\max\{\varsigma_{k-1}, 0\}}{\|s_{k-1}\|} s_{k-1}. \tag{2.18}$$

The first modified spectral CG method is developed for uniformly convex functions with the following search direction

$$d_k = \begin{cases} -F_k, & k = 0; \\ -\theta_k^{2*} F_k + \beta_k^{2*}(t) d_{k-1}, & k \geq 1, \end{cases} \tag{2.19}$$

where $\beta_k^{2*}(t)$ and θ_k^{2*} are as defined in (2.16) and (2.17) respectively.
The second modified spectral CG method is obtained for general functions with search direction given by

$$d_k = \begin{cases} -F_k, & k = 0; \\ -\theta_k^{2*} F_k + \beta_k^{3*}(t) d_{k-1}, & k \geq 1, \end{cases} \tag{2.20}$$

where $\beta_k^{3*}(t)$ is given by

$$\beta_k^{3*}(t) = max\left\{\frac{\theta_k^{2*} F_k^T z_{k-1}^*}{d_{k-1}^T z_{k-1}^*}, 0\right\} - t\frac{F_k^T s_{k-1}}{d_{k-1}^T z_{k-1}^*}, \quad (t \geq 0). \tag{2.21}$$

In this paper, using a revised form of the extended secant condition in (2.10), and two modified spectral coefficients in the classical DL scheme, we propose two enhanced DL-type methods for nonlinear monotone equations.

2.1 The first enhanced Dai–Liao method

Motivated by the above modified spectral coefficients, we suggest the following modified DL update parameter:

$$\hat{\beta}_k = \frac{F_k^T u_{k-1} - \hat{\theta}_k t F_k^T s_{k-1}}{d_{k-1}^T u_{k-1}}, (t \geq 0), \tag{2.22}$$

where $\hat{\theta}_k$ is a modified spectral coefficient given by

$$\hat{\theta}_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T u_{k-1}}. \tag{2.23}$$

As in [50], we redefine u_{k-1} to safeguard $d_{k-1}^T u_{k-1}$ from being zero and obtain its revised form as

$$w_{k-1} = y_{k-1} + \xi \frac{\max\{\varsigma_{k-1}, 0\}}{s_{k-1}^T \mu_{k-1}} \mu_{k-1}. \tag{2.24}$$

Consequently, we get the revised form of (2.22) as

$$\beta_k = \frac{(w_{k-1} - \theta_k t s_{k-1})^T F_k}{d_{k-1}^T w_{k-1}}, (t \geq 0), \tag{2.25}$$

where θ_k is given by

$$\theta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T w_{k-1}}. \tag{2.26}$$

Lemma 2.1 *The revised update parameter in (2.25) is well defined*

Proof For (2.25) to be well-defined, we require $d_{k-1}^T w_{k-1} > 0$. So, from (2.6) and setting $\mu_{k-1} = s_{k-1}$, we obtain

$$\begin{aligned} d_{k-1}^T w_{k-1} &= d_{k-1}^T \left(y_{k-1} + \xi \frac{\max\{\varsigma_{k-1}, 0\}}{s_{k-1}^T s_{k-1}} s_{k-1} \right) \\ &= \alpha_{k-1}^{-1} s_{k-1}^T \left(y_{k-1} + \xi \frac{\max\{\varsigma_{k-1}, 0\}}{s_{k-1}^T s_{k-1}} s_{k-1} \right) \\ &= \alpha_{k-1}^{-1} s_{k-1}^T y_{k-1} + \alpha_{k-1}^{-1} \xi \frac{\max\{\varsigma_{k-1}, 0\}}{s_{k-1}^T s_{k-1}} s_{k-1}^T s_{k-1} \\ &\geq \alpha_{k-1}^{-1} (m \|s_{k-1}\|^2 + \xi \max\{\varsigma_{k-1}, 0\}) > 0. \end{aligned} \tag{2.27}$$

In what follows, we employ the idea in [5] to investigate the descent property of our proposed method. Following Perry's approach [44], search direction of the proposed method can be written as

$$d_k = -H_k F_k \tag{2.28}$$

where H_k, is called the search direction matrix and is given by

$$H_k = I - \frac{s_{k-1} w_{k-1}^T}{s_{k-1}^T w_{k-1}} + \theta_k t \frac{s_{k-1} s_{k-1}^T}{s_{k-1}^T w_{k-1}}. \tag{2.29}$$

By utilizing (2.6) and employing similar procedure as (2.27) we have

$$\begin{aligned} s_{k-1}^T w_{k-1} &= s_{k-1}^T y_{k-1} + \xi \frac{\max\{\varsigma_{k-1}, 0\}}{s_{k-1}^T s_{k-1}} s_{k-1}^T s_{k-1} \\ &\geq m \|s_{k-1}\|^2 + \xi \max\{\varsigma_{k-1}, 0\} \geq m \|s_{k-1}\|^2 > 0. \end{aligned} \tag{2.30}$$

It is clear from (2.29) that H_k is not a symmetric matrix. From (2.28) and applying the approach in [5], we can write

$$d_k^T F_k = -F_k^T H_k^T F_k = -F_k^T \bar{H}_k F_k, \tag{2.31}$$

where

$$\bar{H}_k = \frac{H_k^T + H_k}{2} = I + \frac{s_{k-1}(2\theta_k t s_{k-1} - w_{k-1})^T}{2 s_{k-1}^T w_{k-1}} - \frac{w_{k-1} s_{k-1}^T}{2 s_{k-1}^T w_{k-1}}. \tag{2.32}$$

So, the matrix $\bar{H}_k$ defined by (2.32) is a symmetric matrix. And so, to analyze the descent property of the proposed method, we need to find eigenvalues of $\bar{H}_k$ and their structure using the following theorem.

Theorem 2.1 *Let the symmetric matrix $\bar{H}_k$ be defined by (2.32). Then, its eigenvalues consists of* 1 *with* $(n-2$ *multiplicity) and* φ_k^+ *and* φ_k^-, *where*

$$\varphi_k^+ = \frac{1}{2}\left[(1 + a_k) + \sqrt{(a_k - 1)^2 + b_k - 1}\right] \tag{2.33}$$

$$\varphi_k^- = \frac{1}{2}\left[(1 + a_k) - \sqrt{(a_k - 1)^2 + b_k - 1}\right] \tag{2.34}$$

with

$$a_k = \theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} \tag{2.35}$$

$$b_k = \frac{\|s_{k-1}\|^2 \|w_{k-1}\|^2}{(s_{k-1}^T w_{k-1})^2}. \tag{2.36}$$

Furthermore, eigenvalues of the matrix $\bar{H}_k$ are positive real numbers.

Proof Since $s_{k-1}^T w_{k-1} > 0$, then the vectors s_{k-1} and w_{k-1} are nonzero vectors. Suppose $\text{span}\{s_{k-1}, w_{k-1}\} = \Gamma \subset R^n$, then $dim(\Gamma) \leq 2$ and $dim(\Gamma^{\perp}) \geq n-2$, where $\Gamma^{\perp}$ is the orthogonal complement of Γ. Hence, there exists a set of mutually orthogonal vectors $\{v_{k-1}^i\}_{i=1}^{n-2} \subset \Gamma^{\perp}$ satisfying

$$s_{k-1}^T v_{k-1}^i = w_{k-1}^T v_{k-1}^i = 0, \tag{2.37}$$

which from (2.32) leads to

$$\bar{H}_k v_{k-1}^i = \bar{H}_k^T v_{k-1}^i = v_{k-1}^i, \quad i = 1, \ldots, n-2. \tag{2.38}$$

So, v_{k-1}^i, for $i = 1, \ldots, n-2$ are the eigenvectors of $\bar{H}_k$ with eigenvalue 1 each. Let φ_k^+ and φ_k^- be the remaining two eigenvalues respectively.

Clearly, $\bar{H}_k$ represents a rank-two update, and since from the fundamental algebra formula (see inequality (1.2.70) of [49])

$$det(I + u_1 u_2^T + u_3 u_4^T) = (1 + u_1^T u_2)(1 + u_3^T u_4) - (u_1^T u_4)(u_2^T u_3), \tag{2.39}$$

we have,

$$det(\bar{H}_k) = \frac{1}{4} + \theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - \frac{\|s_{k-1}\|^2 \|w_{k-1}\|^2}{4(s_{k-1}^T w_{k-1})^2}. \tag{2.40}$$

Also, since trace of a square matrix equals to sum of its eigenvalues, we have

$$trace(\bar{H}_k) = n - 1 + \theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} = \underbrace{1 + \cdots + 1}_{\text{(n-2) times}} + \varphi_k^+ + \varphi_k^-, \tag{2.41}$$

for which we have

$$\varphi_k^+ + \varphi_k^- = 1 + \theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}}. \tag{2.42}$$

Using the relationship between trace and determinant of a matrix and its eigenvalues, we obtain φ_k^+ and φ_k^- as roots of the following quadratic polynomial:

$$\varphi^2 - \left(1 + \theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}}\right)\varphi + \frac{1}{4} + \theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - \frac{\|s_{k-1}\|^2 \|w_{k-1}\|^2}{4(s_{k-1}^T w_{k-1})^2} = 0. \tag{2.43}$$

By applying the quadratic formula with some rearrangements, we obtain

$$\varphi_k^{\pm} = \frac{1}{2}\left[\left(1 + \theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}}\right) \pm \sqrt{\left(\theta_k t \frac{\|s_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - 1\right)^2 + \frac{\|w_{k-1}\|^2 \|s_{k-1}\|^2}{(s_{k-1}^T w_{k-1})^2} - 1}\,\right]. \tag{2.44}$$

We can write (2.44) as

$$\varphi_k^{\pm} = \frac{1}{2}\left[(1+a_k) \pm \sqrt{(a_k-1)^2 + b_k - 1}\right], \tag{2.45}$$

which proves (2.33) and (2.34).
Applying Cauchy–Schwarz inequality to (2.44), we see that $\varphi_k^+ > 0$. And after some algebra, it can be seen that $\varphi_k^- > 0$ for

$$t > \frac{1}{4\theta_k}\left(\frac{\|w_{k-1}\|^2}{s_{k-1}^T w_{k-1}} - \frac{s_{k-1}^T w_{k-1}}{\|s_{k-1}\|^2}\right). \tag{2.46}$$

Next, using $\theta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T w_{k-1}}$ and performing some algebra, we suggest the following two parameter choices for t in the proposed method.

$$t_{k1} = p\frac{\|w_{k-1}\|^2}{\|s_{k-1}\|^2} - q\frac{(s_{k-1}^T w_{k-1})^2}{\|s_{k-1}\|^4}, \tag{2.47}$$

where $p > \frac{1}{4}$ and $q < \frac{1}{4}$. Following (2.47), we make the following important remark

Remark 2.1 Since the DL parameter t is nonnegative, we restrict the values of the parameter q in (2.47) to be negative so as to avoid a numerically unreasonable approximation [27]. By using (2.47) and applying this remark, we perform some algebraic manipulations in (2.44) and obtain that $0 < \varphi_k^- \leq 1 \leq \varphi_k^+$. Hence, $\varphi_k^- \in (0, 1]$ and so, all eigenvalues of the matrix $\bar{H}_k$ are positive real numbers, which ensures that it is a positive-definite matrix. Consequently, from (2.31), we have

$$d_k^T F_k = -F_k^T \bar{H}_k F_k \leq -\varphi_k^- \|F_k\|^2 < 0, \tag{2.48}$$

which shows that the descent condition is satisfied. We therefore, write the modified Dai–Liao update parameter as

$$\beta_{k1} = \frac{(w_{k-1} - t_{k1}s_{k-1})^T F_k}{d_{k-1}^T w_{k-1}}, \tag{2.49}$$

with $p \geq \frac{1}{4}$ and $q \leq 0$ satisfying (2.47) and guaranteeing the descent condition.
We also write the search direction for the proposed method as

$$d_k = -F_k + \left(\frac{(w_{k-1} - t_{k1}s_{k-1})^T F_k}{d_{k-1}^T w_{k-1}}\right) d_{k-1}. \tag{2.50}$$

We list steps of the algorithm as follows

Algorithm 2.1 Enhanced Dai–Liao method 1 (EDLM1)
Step 0: Choose $x_0 \in \mathbf{R}^n$, $\epsilon > 0$, $\sigma \in (0, 1)$, $\rho \in (0, 1)$. Set $k = 0$ and $d_0 = -F_0$.
Step 1: Compute $F(x_k)$. If $\|F(x_k)\| \leq \epsilon$, stop, otherwise goto **Step** 2.
Step 2: Set $z_k = x_k + \alpha_k d_k$, where the step length$\alpha_k = \rho^{\bar{m}}$ with $\bar{m}$ being the smallest nonnegative integer $\bar{m}$ satisfying

$$-F(x_k + \rho^{\bar{m}} d_k)^T d_k \geq \sigma\rho^{\bar{m}}\|d_k\|^2. \tag{2.51}$$

Step 3 : If $\|F(z_k)\| = 0$ stop, else let $x_{k+1} = x_k - \frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|^2} F(z_k)$.
Step 4: Compute the search direction d_k by using (2.50).
Step 5: Set $k = k + 1$ and goto **Step** 1.

2.2 The second enhanced Dai–Liao method

To further obtain a more enhanced modified DL method, we employ an extended version of the modified quasi-Newton equation presented in [67,68] and another modified spectral parameter to define the second update parameter as:

$$\bar{\beta}_k = \frac{F_k^T \bar{y}_{k-1} - \bar{\theta}_k t F_k^T s_{k-1}}{d_{k-1}^T \bar{y}_{k-1}}, \tag{2.52}$$

where $\bar{\theta}_k$ and $\bar{y}_{k-1}$ are given by

$$\bar{\theta}_k = \frac{\bar{y}_{k-1}^T \bar{y}_{k-1}}{s_{k-1}^T \bar{y}_{k-1}}, \tag{2.53}$$

and

$$\bar{y}_{k-1} = y_{k-1} + \kappa \frac{\max\{\vartheta_{k-1}, 0\}}{s_{k-1}^T \mu_{k-1}} \mu_{k-1}, \tag{2.54}$$

with ϑ_{k-1} as defined in (2.9). We adopt similar approach as in the previous case to investigate the descent property of the second method, and consequently obtain the following estimation for the modified DL parameter and definition for the corresponding update parameter as follows:

$$t_{k2} = p* - q* \frac{(s_{k-1}^T \bar{y}_{k-1})^2}{\|s_{k-1}\|^2 \|\bar{y}_{k-1}\|^2}, \tag{2.55}$$

$$\beta_{k2} = \frac{(\bar{y}_{k-1} - t_{k2} s_{k-1})^T F_k}{d_{k-1}^T \bar{y}_{k-1}}, \tag{2.56}$$

with $p* \geq \frac{1}{4}$ and $q* \leq 0$ satisfying (2.55) and guaranteeing the descent condition. Applying similar argument as in (2.27) and (2.30), it's obvious to see that

$$d_{k-1}^T \bar{y}_{k-1} \geq \alpha_{k-1}^{-1} (m \|s_{k-1}\|^2 + \kappa \max\{\vartheta_{k-1}, 0\}) > 0, \tag{2.57}$$

and

$$\begin{aligned} s_{k-1}^T \bar{y}_{k-1} &= s_{k-1}^T y_{k-1} + \kappa \frac{\max\{\vartheta_{k-1}, 0\}}{s_{k-1}^T s_{k-1}} s_{k-1}^T s_{k-1} \\ &\geq m \|s_{k-1}\|^2 + \kappa \max\{\varsigma_{k-1}, 0\} \geq m \|s_{k-1}\|^2. \end{aligned} \tag{2.58}$$

We therefore, write the search direction for the second proposed method as

$$d_k = -F_k + \left(\frac{(\bar{y}_{k-1} - t_{k2} s_{k-1})^T F_k}{d_{k-1}^T \bar{y}_{k-1}} \right) d_{k-1}. \tag{2.59}$$

Next, we list the steps of the algorithm for the second method

Algorithm 2.2 Enhanced Dai–Liao method 2 (EDLM2)
Step 0: Choose $x_0 \in \mathbf{R}^n$, $\epsilon > 0$, $\sigma \in (0, 1)$, $\rho \in (0, 1)$. Set $k = 0$ and $d_0 = -F_0$.
Step 1: Compute $F(x_k)$. If $\|F(x_k)\| \leq \epsilon$, stop, otherwise goto **Step** 2.
Step 2: Set $z_k = x_k + \alpha_k d_k$, where the step length$\alpha_k = \rho^{\bar{m}}$ with $\bar{m}$ being the smallest nonnegative integer $\bar{m}$ satisfying

$$-F(x_k + \rho^{\bar{m}} d_k)^T d_k \geq \sigma \rho^{\bar{m}} \|d_k\|^2. \tag{2.60}$$

Step 3: If $\|F(z_k)\| = 0$ stop, else let $x_{k+1} = x_k - \frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|^2} F(z_k)$.
Step 4: Compute the search direction d_k by using (2.59).
Step 5: Set $k = k + 1$ and goto **Step** 1.

3 Convergence result

To discuss the global convergence of **Algorithm** 2.1, we require the following assumptions:

Assumption 3.1 The mapping F is continuous and monotone.

Assumption 3.2 The solution set of F is not empty; i.e., there exists a solution $x^* \in \mathbf{R}^n$ satisfying $F(x^*) = 0$.

Assumption 3.3 The function F is Lipschitz continuous; namely, there exists a positive constant L such that

$$\|F(x) - F(y)\| \leq L\|x - y\|, \quad \forall x, y \in \mathbf{R}^n. \tag{3.1}$$

The next lemma is used to show that the line search used in **Algorithm** 2.1 is well-defined.

Lemma 3.1 *Let the mapping F be continuous and monotone. Then there exists a step-size α_k such that* (2.60) *holds for all $k \geq 0$.*

Proof We go by contradiction and assume the conclusion is not true. It therefore, implies there exists a constant $k_0 \geq 0$, such that for any nonnegative integer $\bar{m}$, we get

$$-\left\langle F(x_{k_0} + \rho^{\bar{m}} d_{k_0}), d_{k_0} \right\rangle < \sigma \rho^{\bar{m}} \|d_{k_0}\|^2. \tag{3.2}$$

Also, employing the monotonicity and continuity of F, with (3.2), and since $\rho \in (0, 1)$, letting $\bar{m} \to \infty$, we get

$$-F(x_{k_0})^T d_{k_0} \leq 0. \tag{3.3}$$

From (2.48), it follows that

$$-F(x_{k_0})^T d_{k_0} \geq \varphi_k \|F(x_{k_0})\|^2 > 0, \tag{3.4}$$

which clearly contradicts (3.3). Hence, the line search is well-defined.

We also used the following Lemma, which originated from Lemma 2.1 in the work of Solodov and Svaiter [48], which also holds for **Algorithm** 2.1. We omit the proof since it is similar to that in [48].

Lemma 3.2 (Solodov and Svaiter [48]) *Suppose F is monotone and Assumptions 3.2, and 3.3 and hold. Let the sequence $\{x_k\}$ be generated by **Algorithm** 2.1 with search direction d_k given by* (2.59). *Then for any x^* satisfying $F(x^*) = 0$, we have*

$$\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2 - \|x_{k+1} - x_k\|^2, \tag{3.5}$$

and the sequence $\{x_k\}$ is bounded. Also, either the sequence $\{x_k\}$ is finite with the solution of (1.1) *being the last iteration or $\{x_k\}$ is infinite and*

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\|^2 < \infty, \tag{3.6}$$

which implies that

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0. \tag{3.7}$$

Lemma 3.3 *Let the sequence $\{x_k\}$ be generated by* ***Algorithm*** *2.1, then*

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{3.8}$$

Proof By Lemma 3.2 the sequence $\{\|x_k - x^*\|\}$ is non-increasing and convergent, hence it is bounded. Using definition of z_k, (2.4) and line search condition (2.60), we obtain

$$\|x_{k+1} - x_k\| = \frac{|F(z_k)^T(x_k - z_k)|}{\|F(z_k)\|} = \frac{|\alpha_k F(z_k)^T d_k|}{\|F(z_k)\|} \le \frac{\alpha_k^2 |\sigma| \|F(z_k)\| \|d_k\|^2}{\|F(z_k)\|} = \alpha_k^2 |\sigma| \|d_k\|^2 \tag{3.9}$$

From (3.7) and (3.9) it follows that

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{3.10}$$

Lemma 3.4 *Suppose F is monotone and Assumptions 3.2, and 3.3 hold. Let the sequence $\{x_k\}$ be generated by* ***Algorithm*** *2.1 with search direction given by (2.59). Then the sequences $\{F(x_k)\}$ and $\{F(z_k)\}$ are bounded; namely there exists a constant $\varpi > 0$ such that*

$$\|F(x_k)\| \le \varpi, \quad \|F(z_k)\| \le \varpi. \tag{3.11}$$

Proof By Lemma 3.2 we obtain

$$\|x_k - x^*\|^2 \le \|x_0 - x^*\|^2 - \|x_k - x_0\|^2 \le \|x_0 - x^*\|^2, \tag{3.12}$$

for which, we consequently have

$$\|x_k - x^*\| \le \|x_0 - x^*\|. \tag{3.13}$$

Also, by (3.10) we see that there exists a constant $\bar{\varpi} > 0$ satisfying $\alpha_k \|d_k\| \le \bar{\varpi}$. Also, using (3.13) and definition of z_k, we have

$$\|z_k - x^*\| \le \|x_k - x^*\| + \alpha_k \|d_k\| \le \|x_0 - x^*\| + \bar{\varpi}. \tag{3.14}$$

Using (3.13) and since F is Lipschitz by Assumption 3.3, it follows that

$$\|F(x_k)\| = \|F(x_k) - F(x^*)\| \le L\|x_k - x^*\| \le L\|x_0 - x^*\|. \tag{3.15}$$

Also, using similar argument, we have

$$\|F(z_k)\| = \|F(z_k) - F(x^*)\| \le L\|z_k - x^*\| \le L(\|x_0 - x^*\| + \bar{\varpi}). \tag{3.16}$$

By setting $\varpi = \max\{L\|x_0 - x^*\|, L(\|x_0 - x^*\| + \bar{\varpi}\}$, the result is established.

Lemma 3.5 *Suppose Assumption 3.3 holds and the sequence $\{x_k\}$ generated by* ***Algorithm*** *2.1 is bounded, then there exists a constant $\bar{M} > 0$ such that*

$$\|d_k\| \le \bar{M}, \quad \forall k. \tag{3.17}$$

From (2.11) and applying the mean-value theorem, we have

$$\begin{aligned} \varsigma_{k-1} &= 2(f_{k-1} - f_k) + (F_{k-1} + F_k)^T s_{k-1} \\ &= (-2\nabla f(\zeta) + \nabla f(x_{k-1}) + \nabla f(x_k))^T s_{k-1}, \end{aligned} \tag{3.18}$$

where $\zeta = \lambda x_{k-1} + (1-\lambda)x_k$, for some $\lambda \in (0, 1)$.
Hence from (3.1), we get

$$\begin{aligned}|\varsigma_{k-1}| &\le (\|\nabla f(x_{k-1}) - \nabla f(\zeta)\| + \|\nabla f(x_k) - \nabla f(\zeta)\|)\|s_{k-1}\| \\ &\le (L(1-\lambda)\|s_{k-1}\| + L\lambda\|s_{k-1}\|)\|s_{k-1}\| \\ &= L\|s_{k-1}\|^2.\end{aligned} \tag{3.19}$$

Utilizing (2.24), (3.1), (3.19) and setting $\mu_{k-1} = s_{k-1}$, we obtain

$$\|w_{k-1}\| \le L\|s_{k-1}\| + \xi L\|s_{k-1}\| = (L + \xi L)\|s_{k-1}\|. \tag{3.20}$$

And from (2.47), (3.20), and the Cauchy–Shwartz inequality, we obtain

$$\begin{aligned}|t_{k1}| &\le p\frac{\|w_{k-1}\|^2}{\|s_{k-1}\|^2} + |q|\frac{(s_{k-1}^T w_{k-1})^2}{\|s_{k-1}\|^4} \\ &\le p\frac{\|w_{k-1}\|^2}{\|s_{k-1}\|^2} + |q|\frac{(\|s_{k-1}\|\|w_{k-1}\|)^2}{\|s_{k-1}\|^4} \\ &\le p\frac{((L+\xi L)\|s_{k-1}\|)^2}{\|s_{k-1}\|^2} + |q|\frac{((L+\xi L)\|s_{k-1}\|^2)^2}{\|s_{k-1}\|^4} \\ &= p(L+\xi L)^2 + |q|(L+\xi L)^2 \\ &= (p + |q|)(L+\xi L)^2).\end{aligned} \tag{3.21}$$

So, from Cauchy–Schwarz inequality, (1.6), (3.1), (3.11), (3.20) and (3.21) we obtain,

$$\begin{aligned}\|d_k\| &= \| - F_k + \beta_k^1 d_{k-1}\| \\ &= \left\| -F_k + \frac{w_{k-1}^T F_k}{s_{k-1}^T w_{k-1}} s_{k-1} - t_{k1}\frac{s_{k-1}^T F_k}{s_{k-1}^T w_{k-1}} s_{k-1}\right\| \\ &\le \|F_k\| + \frac{\|w_{k-1}\|\|F_k\|}{m\|s_{k-1}\|^2}\|s_{k-1}\| + |t_{k1}|\frac{\|s_{k-1}\|\|F_k\|}{m\|s_{k-1}\|^2}\|s_{k-1}\| \\ &\le \varpi + \frac{(L+\xi L)\|s_{k-1}\|^2\varpi}{m\|s_{k-1}\|^2} + \big((p+|q|)(L+\xi L)^2)\big)\frac{\|s_{k-1}\|^2\varpi}{m\|s_{k-1}\|^2} \\ &= \varpi + \frac{(L+\xi L)\varpi}{m} + \big((p+|q|)(L+\xi L)^2)\big)\frac{\varpi}{m}. \\ &\left(1 + \frac{(L+\xi L)}{m} + \frac{\big((p+|q|)(L+\xi L)^2)\big)}{m}\right)\varpi.\end{aligned} \tag{3.22}$$

Setting $\bar{M} := \left(1 + \frac{(L+\xi L)}{m} + \frac{\big((p+|q|)(L+\xi L)^2)\big)}{m}\right)\varpi$, the result is established.
We use the following theorem to establish global convergence of Algorithm 2.1.

Theorem 3.1 *Suppose F is monotone and Assumptions 3.2 and 3.3 hold. Let $\{x_k\}$ and $\{z_k\}$ be sequences generated by* ***Algorithm*** *2.1. Then*

$$\liminf_{k\to\infty} \|F(x_k)\| = 0. \tag{3.23}$$

Proof The proof is established by contradiction. Suppose (3.23) is not true. Then there exists $\psi > 0$ such that $\|F(x_k)\| \ge \psi$ holds. By the sufficient descent condition (2.48) and Cauchy–Schwarz inequality, we have

$$\varphi\|F(x_k)\|^2 \le -F(x_k)^T d_k \le \|F(x_k)\|\|d_k\|, \quad \forall k \in \mathbb{N} \cup \{0\}. \tag{3.24}$$

Hence,

$$\|d_k\| \geq \varphi\psi > 0. \tag{3.25}$$

Therefore, utilizing (3.10) and (3.25), we obtain

$$\lim_{k\to\infty} \alpha_k = 0. \tag{3.26}$$

By the line search rule, and for all k sufficiently large, we have that $\sigma\beta^{\bar{m}_k-1}$ will not satisfy (2.60), which implies that

$$-F(x_k + \beta^{\bar{m}_k-1}d_k)^T d_k < \sigma\beta^{\bar{m}_k-1}\|d_k\|^2. \tag{3.27}$$

Also, the boundedness of the sequences $\{x_k\}$ and $\{d_k\}$ implies there exists accumulation points say $\tilde{x}$ and $\tilde{d}$ and infinite index sets N_1 and N_2 with $N_2 \subset N_1$ such that $\lim_{k\to\infty} x_k = \tilde{x}$, for $k \in N_1$ and $\lim_{k\to\infty} d_k = \tilde{d}$, for $k \in N_2$. Therefore, by taking limits as $k \to \infty$ in (3.27) for $k \in N_2$ we obtain

$$F(\tilde{x})^T\tilde{d} > 0. \tag{3.28}$$

On the other hand, by taking limits as $k \to \infty$ on both sides of the sufficient descent condition (2.48), we have

$$F(\tilde{x})^T\tilde{d} \leq 0. \tag{3.29}$$

Clearly, (3.28) and (3.29) shows a contradiction. Therefore, $\liminf_{k\to\infty} \|F(x_k)\| = 0$ and the proof is established.

Next we prove the global convergence of **Algorithm** 2.2. By employing similar procedure as applied in the case of **Algorithm** 2.1 with the same line search condition, it follows from Lemmas 3.2, 3.3, and 3.4 that the sequence $\{x_k\}$ generated by **Algorithm** 2.2 is bounded.

Lemma 3.6 *Suppose Assumption 3.3 holds. Let the sequence $\{x_k\}$ generated by **Algorithm** 2.2 be bounded, then there exists a constant $\tilde{M} > 0$ such that*

$$\|d_k\| \leq \tilde{M}, \quad \textit{for all } k. \tag{3.30}$$

Proof By the mean-value theorem, we have

$$\begin{aligned}\vartheta_{k-1} &= 6(f_{k-1} - f_k) + 3(F_{k-1} + F_k)^T s_{k-1} \\ &= 3(-2\nabla f(\bar{\xi}) + \nabla f(x_{k-1}) + \nabla f(x_k))^T s_{k-1},\end{aligned} \tag{3.31}$$

where $\bar{\xi} = \bar{\lambda}x_{k-1} + (1-\bar{\lambda})x_k$, for some $\bar{\lambda} \in (0, 1)$.

Thus, from (3.1), we get

$$\begin{aligned}|\vartheta_{k-1}| &\leq 3(\|\nabla f(x_{k-1}) - \nabla f(\bar{\xi})\| + \|\nabla f(x_k) - \nabla f(\bar{\xi})\|)\|s_{k-1}\| \\ &\leq 3(L(1-\bar{\lambda})\|s_{k-1}\| + L\bar{\lambda}\|s_{k-1}\|)\|s_{k-1}\| \\ &= 3L\|s_{k-1}\|^2.\end{aligned} \tag{3.32}$$

Utilizing (2.54), (3.1), (3.32) and setting $\mu_{k-1} = s_{k-1}$, we have

$$\|\bar{y}_{k-1}\| \leq L\|s_{k-1}\| + 3\kappa L\|s_{k-1}\| = L\|s_{k-1}\|(1+3\kappa). \tag{3.33}$$

And from (2.49), (3.33) and the Cauchy inequality, we obtain

$$\begin{aligned}|t_{k2}| &\leq p* + |q*| \frac{(s_{k-1}^T \bar{y}_{k-1})^2}{\|s_{k-1}\|^2 \|\bar{y}_{k-1}\|^2} \\ &\leq p* + |q*| \frac{(\|s_{k-1}\| \|\bar{y}_{k-1}\|)^2}{\|s_{k-1}\|^2 \|\bar{y}_{k-1}\|^2} \\ &\leq p* + |q*| \frac{L^2 \|s_{k-1}\|^4 (1+3k)^2}{L^2 \|s_{k-1}\|^4 (1+3k)^2} \\ &= p* + |q*|.\end{aligned} \tag{3.34}$$

So, from Cauchy–Schwarz inequality, (1.6), (2.58), (3.1), (3.11), (3.33) and (3.34) we obtain,

$$\begin{aligned}\|d_k\| &= \| - F_k + \beta_k^2 d_{k-1}\| \\ &= \left\| -F_k + \frac{\bar{y}_{k-1}^T F_k}{s_{k-1}^T \bar{y}_{k-1}} s_{k-1} - t_{k2} \frac{s_{k-1}^T F_k}{s_{k-1}^T \bar{y}_{k-1}} s_{k-1} \right\| \\ &\leq \|F_k\| + \frac{\|\bar{y}_{k-1}\| \|F_k\|}{m \|s_{k-1}\|^2} \|s_{k-1}\| + |t_{k2}| \frac{\|s_{k-1}\| \|F_k\|}{m\|s_{k-1}\|^2} \|s_{k-1}\| \\ &\leq \|F_k\| + \frac{L(1+3\kappa)\|s_{k-1}\|^2 \|F_k\|}{m\|s_{k-1}\|^2} + (p* + |q*|)) \frac{\|s_{k-1}\|^2 \|F_k\|}{m\|s_{k-1}\|^2} \\ &\leq \varpi + \frac{L(1+3\kappa)\varpi}{m} + (p* + |q*|)) \frac{\varpi}{m} \\ &= \left(1 + \frac{L(1+3\kappa)}{m} + \left(\frac{p* + |q*|}{m}\right)\right) \varpi.\end{aligned} \tag{3.35}$$

Setting $\tilde{M} := \left(1 + \frac{L(1+3\kappa)}{m} + \left(\frac{p*+|q*|}{m}\right)\right) \varpi$, the result is established.
The following theorem is employed to establish global convergence of Algorithm 2.2.

Theorem 3.2 *Let the sequence $\{x_k\}$ be generated by Algorithm 2.2. Then*

$$\liminf_{k \to \infty} \|F(x_k)\| = 0. \tag{3.36}$$

Proof The proof follows the same pattern as that of **Algorithm** 2.1. The search directions of **Algorithm** 2.2 are descent directions, therefore, using similar arguments as for **Algorithm** 2.1, the proof is established.

4 Numerical results

In this section, we present numerical results to show the efficiency of the proposed methods by comparing their performance with the following existing methods in the literature.

- A derivative-free three-term projection algorithm involving spectral quotient for solving nonlinear monotone equations (**DFPA**) [43].
- The Hager–Zhang conjugate gradient algorithm for large-scale nonlinear equations (**HZCGA**) [62].

We used the line search given by (2.60) for all the methods, and set the parameters for the DFPA and HZCGA methods as they are used by the authors in the respective papers. For **Algorithms** 2.1, and 2.2, we set the parameters $\sigma = 10^{-2}$, $\rho = 0.8$, $\xi = 0.1$, $p =$

Table 1 Initial points used for the test problems

INITIAL POINTS (IP)	VALUES
x_1	$\left(\frac{1}{8}, \frac{1}{8}, \dots, \frac{1}{8}\right)^T$
x_2	$\left(\frac{2}{5}, \frac{2}{5}, \dots, \frac{2}{5}\right)^T$
x_3	$(0.1, 0.1, \dots, 0.1)^T$
x_4	$(0.01, 0.01, \dots, 0.01)^T$
x_5	$(0.5, 0.5, \dots, 0.5)^T$
x_6	$(0.2, 0.2, \dots, 0.2)^T$
x_7	$(0.25, 0.25, \dots, 0.25)^T$

$p^* = 0.8$, $q = q^* = -0.25$, and $\mu_{k-1} = s_{k-1}$. Interestingly, selections of the line search parameters for **Algorithms** 2.1, and 2.2 were based on values that yield the best results. Also, the implementation codes were written in Matlab $R2014a$ environment and run on a **PC** (CPU 2.20GHZ, 8GB memory) with windows operating system. Our stopping criteria is $\|F(x_k)\| \leq 10^{-8}$ or $\|F(z_k)\| \leq 10^{-8}$.

Furthermore, using nine test problems with different initial points given in Table 1 and dimensions, numerical results of experiment carried out with all the four methods are reported in Tables 2-5. “Iguess” and “Nvar” indicates the starting point of the iterations and the number of variables for each problem. “NIter” and “Cpu time” stands for total number of iterations and CPU time in seconds respectively. Also, “$\|F_k\|''$ stands for the residual at stopping point.

Also, a summary of the results from Tables 2, 3, 4 and 5 are reported in Tables 6 and 7, where the number of problems for which a method outperforms the other with respect to number of iterations and CPU time are presented. In addition, we use the performance profile of Dolan and Moré [15] as an evaluation tool to approximately assess the performance and efficiency of each of the methods.

4.1 Computation

The following problems were used for the experiments, where the mapping $F(x) : \mathbf{R}^n \to \mathbf{R}^n$ is given by $F(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$, and $x = (x_1, x_2, \dots, x_n)^T$.

Problem 4.1 [29] *Exponential Function.*
$F_1(x) = e^{x_1} - 1$, $F_i(x) = e^{x_i} + x_{i-1} - 1, \quad i = 2, \dots, n-1,$

Problem 4.2 [66] *Logarithmic Function.*
$F_i(x) = \log(x_i + 1) - \frac{x_i}{n}, \quad i = 2, \dots, n.$

Problem 4.3 [66] *Non-smooth Function.*
$F_i(x) = 2x_i - sin|x_i|, \quad i = 1, \dots, n.$

Problem 4.4 [73] *Strictly convex Function.*
$F_i(x) = e^{x_i} - 1, \quad i = 1, 2, \dots, n.$

Problem 4.5 [36] *Tridiagonal exponential Function.*
$F_1(x) = x_1 - e^{\cos(h(x_1+x_2))}$, $F_i(x) = x_i - e^{\cos(h(x_{i-1}+x_i+x_{i+1}))}$,
$i = 2, 3, \dots, n-1$, $F_n(x) = x_n - e^{\cos(h(x_{n-1}+x_n))}$,
where $h = \frac{1}{n+1}$.

Table 2 Numerical results reported for **EDLM1** and **DFPA** [43] methods

Problem	Nvar	Iguess	**EDLM1**				**DFPA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.1	50,000	**x1**	30	33	0.489813	5.75E−09	38	113	0.587227	7.30E−09
		x2	31	34	0.385715	7.05E−09	43	127	0.662345	7.82E−09
		x3	29	32	0.338738	9.45E−09	38	103	0.570218	1.51E−09
		x4	26	29	0.313773	7.73E−09	35	116	0.562481	3.77E−09
		x5	31	34	0.361784	7.22E−09	41	124	0.655133	9.21E−09
		x6	30	33	0.351259	8.97E−09	38	115	0.581933	6.36E−09
		x7	31	34	0.365304	5.47E−09	45	142	0.710301	1.98E−09
	100,000	**x1**	30	33	1.007651	8.01E−09	31	87	1.341232	7.51E−09
		x2	31	34	1.006569	9.81E−09	43	152	2.037529	2.04E−09
		x3	30	33	1.028831	6.54E−09	40	125	1.831282	7.42E−09
		x4	27	30	0.883623	5.26E−09	37	125	1.820474	7.56E−09
		x5	31	34	1.022309	9.99E−09	42	119	1.873276	6.56E−09
		x6	31	34	0.995631	6.13E−09	40	124	1.816753	5.36E−09
		x7	31	34	1.002953	7.50E−09	43	145	2.014189	8.35E−09
4.2	50000	**x1**	29	31	0.414429	5.43E−09	26	92	0.529759	6.06E−09
		x2	30	32	0.422678	6.67E−09	27	104	0.602159	9.27E−09
		x3	28	30	0.393359	9.30E−09	25	90	0.536895	9.44E−09
		x4	25	27	0.374936	9.19E−09	18	59	0.352183	9.28E−09
		x5	30	32	0.424508	7.65E−09	26	97	0.543029	6.44E−09
		x6	29	31	0.408683	8.23E−09	27	93	0.555992	3.01E−09
		x7	29	31	0.411808	9.90E−09	27	82	0.524487	4.31E−09

Table 2 continued

Problem	Nvar	Iguess	EDLM1				DFPA			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
	100,000	**x1**	29	31	1.098678	7.68E−09	26	103	1.515159	9.06E−09
		x2	30	32	1.165211	9.43E−09	29	118	1.721339	5.14E−09
		x3	29	31	1.164342	6.25E−09	26	103	1.540926	5.46E−09
		x4	26	28	0.974166	6.17E−09	21	82	1.209343	8.16E−09
		x5	31	33	1.155565	5.14E−09	26	81	1.337691	6.25E−09
		x6	30	32	1.145223	5.53E−09	27	93	1.475058	4.40E−09
		x7	30	32	1.131641	6.65E−09	27	82	1.406558	6.65E−09
4.3	50,000	**x1**	29	31	0.335977	5.90E−09	37	77	0.485043	8.11E−09
		x2	30	32	0.345104	9.20E−09	39	81	0.508533	8.29E−09
		x3	28	30	0.324328	9.93E−09	37	77	0.488256	6.48E−09
		x4	25	27	0.292336	9.24E−09	33	69	0.438791	6.56E−09
		x5	31	33	0.360661	5.54E−09	40	83	0.525773	5.87E−09
		x6	29	31	0.359175	9.49E−09	38	79	0.529678	7.29E−09
		x7	30	32	0.351792	5.66E−09	38	79	0.499929	9.14E−09
	100,000	**x1**	29	31	0.952556	8.34E−09	38	79	1.456157	6.42E−09
		x2	31	33	1.028698	6.18E−09	40	83	1.528552	6.57E−09
		x3	29	31	0.961137	6.67E−09	37	77	1.425474	9.16E−09
		x4	26	28	0.865574	6.21E−09	33	69	1.270682	9.28E−09
		x5	31	33	1.016479	7.83E−09	40	83	1.562412	8.30E−09
		x6	30	32	1.054507	6.37E−09	39	81	1.486928	5.78E−09
		x7	30	32	0.978444	8.00E−09	39	81	1.570865	7.24E−09

Table 2 continued

Problem	Nvar	Iguess	**EDLM1**				**DFPA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.4	50,000	**x1**	29	31	0.278153	6.28E−09	37	77	0.391758	8.43E−09
		x2	30	32	0.290087	9.98E−09	30	87	0.368519	4.27E−09
		x3	29	31	0.274922	4.97E−09	37	77	0.399297	6.69E−09
		x4	25	27	0.266801	9.30E−09	33	69	0.355835	6.58E−09
		x5	31	33	0.290986	5.78E−09	28	95	0.380252	5.97E−09
		x6	30	32	0.286803	4.90E−09	25	80	0.326691	5.14E−09
		x7	30	32	0.308964	6.20E−09	27	94	0.360582	6.21E−09
	100,000	**x1**	29	31	0.742658	8.88E−09	38	79	1.149996	6.68E−09
		x2	31	33	0.807695	6.71E−09	30	98	1.063903	5.73E−09
		x3	29	31	0.743955	7.03E−09	37	77	1.104908	9.46E−09
		x4	26	28	0.692659	6.25E−09	33	69	0.990238	9.31E−09
		x5	31	33	0.788479	8.17E−09	29	102	1.057749	7.02E−09
		x6	30	32	0.787928	6.93E−09	26	93	0.996223	7.93E−09
		x7	30	32	0.771059	8.77E−09	27	94	0.980553	8.50E−09
4.5	50,000	**x1**	33	35	0.664308	6.21E−09	42	87	0.964105	9.28E−09
		x2	33	35	0.640604	5.55E−09	42	87	0.999451	8.29E−09
		x3	33	35	0.666512	6.27E−09	42	87	0.974345	9.37E−09
		x4	33	35	0.646558	6.49E−09	42	87	0.979913	9.69E−09
		x5	33	35	0.656969	5.31E−09	42	87	0.962812	7.94E−09
		x6	33	35	0.639141	6.03E−09	42	87	0.974965	9.01E−09
		x7	33	35	0.661019	5.91E−09	42	87	0.955133	8.83E−09

Table 2 continued

Problem	Nvar	Iguess	**EDLM1**				**DFPA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
	100,000	**x1**	33	35	1.770074	8.78E−09	43	89	2.906705	7.35E−09
		x2	33	35	1.866593	7.85E−09	43	89	2.796086	6.57E−09
		x3	33	35	1.776667	8.87E−09	43	89	2.810071	7.42E−09
		x4	33	35	1.790015	9.17E−09	43	89	3.004971	7.68E−09
		x5	33	35	1.796489	7.51E−09	43	89	2.962696	6.29E−09
		x6	33	35	1.838625	8.53E−09	43	89	2.784803	7.14E−09
		x7	33	35	1.846387	8.36E−09	43	89	2.835921	7.00E−09

Table 3 Numerical results reported for **EDLM1** and **DFPA** [43] methods Cntd

Problem	Nvar	Iguess	**EDLM1**				**DFPA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.6	50,000	**x1**	30	32	0.347386	7.26E−09	39	182	0.864331	7.27E−09
		x2	28	30	0.327255	5.12E−09	34	150	0.678856	8.65E−09
		x3	30	32	0.356273	7.98E−09	39	182	0.783797	7.49E−09
		x4	31	33	0.364523	5.26E−09	39	171	0.719358	5.91E−09
		x5	24	26	0.297639	9.41E−09	32	147	0.628891	5.83E−09
		x6	30	32	0.351601	5.27E−09	37	167	0.714295	9.49E−09
		x7	29	31	0.342712	8.36E−09	37	167	0.716392	8.91E−09
	100,000	**x1**	31	33	1.087735	5.01E−09	40	184	2.220658	4.86E−09
		x2	28	30	0.994422	7.24E−09	36	165	2.034399	6.17E−09
		x3	31	33	1.020434	5.50E−09	40	184	2.206271	4.82E−09
		x4	31	33	1.043378	7.43E−09	39	182	2.174952	7.69E−09
		x5	25	27	0.827371	6.49E−09	32	147	1.783862	8.68E−09
		x6	30	32	0.987347	7.45E−09	38	168	2.050308	4.67E−09
		x7	30	32	0.996188	5.77E−09	38	169	2.044248	5.63E−09
4.7	50000	**x1**	33	37	0.386509	6.70E−09	25	186	0.652359	8.74E−09
		x2	31	35	0.364842	9.82E−09	27	201	0.699504	2.39E−09
		x3	33	37	0.397593	7.16E−09	25	186	0.652614	7.33E−09
		x4	33	37	0.386167	8.84E−09	23	171	0.593394	7.82E−09
		x5	31	35	0.367987	5.40E−09	27	201	0.726047	1.80E−09
		x6	33	37	0.387557	5.41E−09	27	201	0.700029	2.08E−09
		x7	32	36	0.383034	9.13E−09	27	201	0.710381	2.35E−09

Table 3 continued

Problem	Nvar	Iguess	**EDLM1**				**DFPA**			
			NIter	NFev	Cpu time	$\Vert F(x_k)\Vert$	NIter	NFev	Cpu time	$\Vert F(x_k)\Vert$
	100,000	**x1**	33	37	1.100419	9.48E−09	27	201	2.026483	2.12E−09
		x2	32	36	1.096034	6.99E−09	27	201	2.014769	3.38E−09
		x3	34	38	1.121561	5.09E−09	27	201	2.021784	1.76E−09
		x4	34	38	1.153864	6.29E−09	25	186	1.869027	1.87E−09
		x5	31	35	1.037959	7.64E−09	27	201	2.031946	2.58E−09
		x6	33	37	1.103706	7.64E−09	27	201	2.022043	2.94E−09
		x7	33	37	1.145837	6.50E−09	27	201	2.103289	3.32E−09
4.8	50,000	**x1**	28	30	0.587719	6.43E−09	34	98	0.977923	4.13E−09
		x2	29	31	0.625754	9.61E−09	41	123	1.218687	8.74E−09
		x3	28	30	0.590284	5.19E−09	36	117	1.099731	1.68E−09
		x4	24	26	0.537798	9.92E−09	29	81	0.834871	5.07E−09
		x5	30	32	0.620374	5.83E−09	36	93	0.995575	5.35E−09
		x6	29	31	0.604292	4.94E−09	70	268	2.292185	9.41E−09
		x7	29	31	0.604536	6.12E−09	35	97	0.987468	1.46E−10
	100,000	**x1**	28	30	1.739887	8.71E−09	46	140	3.976532	6.25E−09
		x2	30	32	1.848202	6.37E−09	40	113	3.531795	3.32E−09
		x3	28	30	1.84893	7.03E−09	32	81	2.879789	8.77E−09
		x4	25	27	1.515685	6.58E−09	29	80	2.326289	6.58E−09
		x5	30	32	1.824991	7.89E−09	29	70	2.225352	6.79E−09
		x6	29	31	1.924621	6.69E−09	35	103	2.905366	7.37E−09
		x7	29	31	1.737095	8.29E−09	51	137	4.032241	3.25E−09

Table 3 continued

Problem	Nvar	Iguess	**EDLM1**				**DFPA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.9	50,000	**x1**	32	35	0.439561	6.75E−09	28	119	0.603218	7.13E−09
		x2	32	35	0.439382	9.66E−09	31	136	0.674458	9.90E−09
		x3	32	35	0.428227	6.47E−09	33	141	0.712497	5.55E−09
		x4	32	35	0.446203	5.29E−09	33	141	0.722223	8.14E−09
		x5	33	36	0.449261	5.27E−09	29	121	0.625061	2.91E−09
		x6	32	35	0.431188	7.35E−09	32	138	0.705793	9.45E−09
		x7	32	35	0.449326	8.08E−09	31	136	0.678853	8.30E−09
	100,000	**x1**	32	35	1.253503	9.60E−09	34	142	2.202857	7.07E−09
		x2	33	36	1.300137	6.29E−09	32	137	2.083561	7.51E−09
		x3	32	35	1.265884	9.04E−09	34	142	2.323332	4.72E−09
		x4	32	35	1.268783	7.77E−09	34	153	2.291167	7.06E−09
		x5	33	36	1.395067	7.52E−09	34	140	2.174041	9.85E−09
		x6	33	36	1.290347	5.27E−09	34	152	2.254044	8.40E−09
		x7	33	36	1.303173	5.36E−09	34	142	2.185485	7.74E−09

Table 4 Numerical results reported for **EDLM2** and **HZCGA** [62] methods

Problem	Nvar	Iguess	**EDLM2**				**HZCGA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.1	50,000	**x1**	25	93	0.505055	6.19E−09	58	119	0.854308	9.55E−09
		x2	27	94	0.479709	6.62E−09	63	129	0.883234	8.26E−09
		x3	27	91	0.492753	7.84E−09	57	117	0.793115	9.73E−09
		x4	25	101	0.519661	8.38E−09	47	97	0.675189	9.99E−09
		x5	22	75	0.397295	6.70E−09	63	129	1.036221	9.65E−09
		x6	31	98	0.585416	4.24E−09	60	123	0.829162	9.27E−09
		x7	121	787	2.978936	5.88E−09	61	125	0.866401	8.95E−09
	100,000	**x1**	25	72	1.197482	7.81E−09	57	117	2.262321	9.95E−09
		x2	31	125	1.923596	7.20E−09	62	127	2.593527	8.59E−09
		x3	30	124	1.738039	2.59E−09	57	117	2.224119	8.10E−09
		x4	22	78	1.115062	5.30E−09	47	97	1.838609	8.44E−09
		x5	33	114	1.661462	9.70E−09	63	129	2.459103	8.02E−09
		x6	28	94	1.388776	2.29E−09	59	121	2.315582	9.64E−09
		x7	32	128	1.736303	7.83E−09	60	123	2.354098	9.30E−09
4.2	50000	**x1**	29	31	0.426039	8.12E−09	85	172	1.420072	9.39E−09
		x2	32	34	0.474619	5.49E−09	95	192	1.580515	9.19E−09
		x3	29	31	0.433464	6.10E−09	83	168	1.368684	9.52E−09
		x4	25	27	0.444062	9.50E−09	86	175	1.376307	9.38E−09
		x5	32	34	0.469434	8.76E−09	97	196	1.783032	8.92E−09
		x6	30	32	0.445441	7.44E−09	89	180	1.509647	9.48E−09
		x7	31	33	0.461453	5.00E−09	91	184	1.536461	9.27E−09

Table 4 continued

Problem	Nvar	Iguess	**EDLM2**				**HZCGA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
	100,000	**x1**	30	32	1.149371	5.45E−09	87	176	3.82665	8.50E−09
		x2	32	34	1.405715	7.76E−09	97	196	4.264982	8.32E−09
		x3	29	31	1.131779	8.62E−09	85	172	4.070032	8.62E−09
		x4	26	28	0.984784	6.38E−09	88	179	3.921813	8.49E−09
		x5	33	35	1.700002	5.88E−09	99	200	4.385109	8.07E−09
		x6	31	33	1.233735	5.00E−09	91	184	4.025002	8.58E−09
		x7	31	33	1.316151	7.07E−09	93	188	4.108819	8.39E−09
4.3	50,000	**x1**	29	31	0.346843	5.82E−09	97	197	1.401274	9.99E−09
		x2	30	32	0.446416	8.02E−09	103	209	1.656151	8.25E−09
		x3	28	30	0.382721	9.84E−09	96	195	1.433826	1.00E−08
		x4	25	27	0.335068	9.24E−09	86	175	1.178682	9.32E−09
		x5	30	32	0.514401	9.44E−09	104	211	1.448206	8.17E−09
		x6	29	31	0.377271	9.16E−09	100	203	1.382061	8.16E−09
		x7	30	32	0.364189	5.35E−09	101	205	1.411491	8.14E−09
	100,000	**x1**	29	31	1.000751	8.23E−09	99	201	3.846954	9.04E−09
		x2	31	33	1.086291	5.39E−09	104	211	4.020651	9.33E−09
		x3	29	31	0.999362	6.61E−09	98	199	3.976286	9.05E−09
		x4	26	28	0.902691	6.21E−09	88	179	3.418625	8.44E−09
		x5	31	33	1.094576	6.34E−09	105	213	4.296581	9.24E−09
		x6	30	32	1.028169	6.15E−09	101	205	4.254667	9.23E−09
		x7	30	32	1.097193	7.57E−09	102	207	3.936049	9.21E−09

Table 4 continued

Problem	Nvar	Iguess	**EDLM2**				**HZCGA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.4	50,000	**x1**	28	30	0.285688	8.75E−09	97	197	1.032416	9.27E−09
		x2	29	31	0.305701	5.74E−09	102	207	1.108237	8.25E−09
		x3	28	30	0.288467	7.51E−09	96	195	0.995156	9.41E−09
		x4	25	27	0.346683	8.99E−09	86	175	0.889628	9.26E−09
		x5	29	31	0.296837	4.86E−09	102	207	1.054617	9.73E−09
		x6	29	31	0.409841	5.38E−09	99	201	1.063082	9.07E−09
		x7	29	31	0.357786	5.81E−09	100	203	1.056124	8.80E−09
	100,000	**x1**	29	31	0.770093	5.88E−09	99	201	2.974819	8.39E−09
		x2	29	31	0.761794	8.12E−09	103	209	3.067058	9.33E−09
		x3	29	31	1.104645	5.04E−09	98	199	2.914009	8.52E−09
		x4	26	28	0.701912	6.04E−09	88	179	2.633265	8.38E−09
		x5	29	31	0.778757	6.88E−09	104	211	3.097101	8.80E−09
		x6	29	31	0.780514	7.61E−09	101	205	3.014429	8.21E−09
		x7	29	31	0.811894	8.21E−09	101	205	3.067376	9.96E−09
4.5	50000	**x1**	33	35	0.702851	6.21E−09	111	225	2.634085	9.13E−09
		x2	33	35	0.704072	5.55E−09	111	225	2.621711	8.16E−09
		x3	33	35	0.690265	6.27E−09	111	225	2.650444	9.22E−09
		x4	33	35	0.780309	6.49E−09	111	225	2.673704	9.54E−09
		x5	33	35	0.691742	5.31E−09	110	223	2.624212	9.76E−09
		x6	33	35	0.700124	6.03E−09	111	225	2.974886	8.87E−09
		x7	33	35	0.802443	5.91E−09	111	225	2.634274	8.69E−09

Table 4 continued

Problem	Nvar	Iguess	**EDLM2**				**HZCGA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
	100,000	**x1**	33	35	1.922363	8.78E−09	113	229	7.712202	8.27E−09
		x2	33	35	1.940334	7.85E−09	112	227	7.20988	9.24E−09
		x3	33	35	1.904203	8.87E−09	113	229	7.280158	8.35E−09
		x4	33	35	1.985556	9.17E−09	113	229	7.262626	8.63E−09
		x5	33	35	1.829754	7.51E−09	112	227	7.251583	8.84E−09
		x6	33	35	1.811047	8.53E−09	113	229	7.271465	8.03E−09
		x7	33	35	1.859011	8.36E−09	112	227	7.194617	9.83E−09

Table 5 Numerical results reported for **EDLM2** and **HZCGA** [62] methods Cntd

Problem	Nvar	Iguess	**EDLM2**				**HZCGA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.6	50,000	**x1**	7	30	0.196452	1.91E−09	50	103	0.806002	8.76E−09
		x2	6	26	0.164314	3.82E−09	47	97	0.635904	8.19E−09
		x3	7	30	0.169746	2.28E−09	50	103	0.734982	9.42E−09
		x4	7	30	0.177922	4.16E−09	51	105	0.703398	7.43E−09
		x5	5	22	0.122853	7.97E−09	43	89	0.605423	6.46E−09
		x6	7	30	0.236555	1.08E−09	50	103	0.734005	6.82E−09
		x7	7	30	0.177381	7.05E−10	49	101	0.705166	8.91E−09
	100,000	**x1**	7	30	0.426635	2.70E−09	51	105	2.021498	7.75E−09
		x2	6	26	0.366163	5.40E−09	48	99	1.889733	7.25E−09
		x3	7	30	0.608261	3.23E−09	51	105	2.026053	8.33E−09
		x4	7	30	0.433895	5.89E−09	52	107	2.109968	6.58E−09
		x5	6	26	0.380494	3.55E−10	43	89	1.842022	9.14E−09
		x6	7	30	0.418639	1.53E−09	50	103	1.991889	9.64E−09
		x7	7	30	0.429102	9.97E−10	50	103	2.096161	7.88E−09
4.7	50,000	**x1**	7	60	0.253259	3.43E−09	28	59	0.427891	9.84E−09
		x2	7	60	0.249664	5.14E−10	28	59	0.464283	4.35E−09
		x3	7	60	0.243637	3.99E−09	29	61	0.414704	4.39E−09
		x4	7	60	0.233421	6.81E−09	29	61	0.453511	5.30E−09
		x5	6	52	0.296699	7.16E−09	27	57	0.408699	6.18E−09
		x6	7	60	0.263567	2.16E−09	28	59	0.401184	8.22E−09
		x7	7	60	0.236933	1.56E−09	28	59	0.396051	7.19E−09

Table 5 continued

Problem	Nvar	Iguess	EDLM2				HZCGA			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
	100,000	**x1**	7	60	0.617018	4.85E−09	29	61	1.191382	5.88E−09
		x2	7	60	0.651218	7.27E−10	28	59	1.150763	6.15E−09
		x3	7	60	0.671942	5.64E−09	29	61	1.609017	6.21E−09
		x4	7	60	0.636022	9.63E−09	29	61	1.171391	7.49E−09
		x5	7	60	0.656395	2.77E−10	27	57	1.131718	8.73E−09
		x6	7	60	0.640913	3.05E−09	29	61	1.221237	4.91E−09
		x7	7	60	0.618291	2.20E−09	29	61	1.259039	4.30E−09
4.8	50000	**x1**	22	36	0.550811	2.81E−09	76	154	2.012518	8.62E−09
		x2	23	35	0.570065	9.52E−09	80	162	2.146476	9.93E−09
		x3	21	32	0.517284	4.64E−09	75	152	2.292631	8.62E−09
		x4	19	35	0.485118	1.65E−09	66	134	1.768896	6.62E−09
		x5	23	33	0.597401	3.37E−09	81	164	2.337694	9.93E−09
		x6	23	39	0.588045	2.08E−09	78	158	2.131861	7.40E−09
		x7	23	37	0.568596	1.19E−09	79	160	2.155367	7.40E−09
	100,000	**x1**	22	30	1.681471	5.71E−09	80	162	6.078064	8.53E−09
		x2	25	39	1.847684	1.29E−09	85	172	6.057528	7.08E−09
		x3	23	37	1.772977	4.91E−09	79	160	5.679959	8.53E−09
		x4	20	34	1.423102	2.08E−09	65	132	4.681855	6.65E−09
		x5	24	33	1.576301	8.37E−09	86	174	6.096488	7.08E−09
		x6	24	38	1.654674	2.64E−09	82	166	6.544013	7.89E−09
		x7	24	37	2.357426	2.47E−09	83	168	5.919489	7.89E−09

Table 5 continued

Problem	Nvar	Iguess	**EDLM2**				**HZCGA**			
			NIter	NFev	Cpu time	$\|F(x_k)\|$	NIter	NFev	Cpu time	$\|F(x_k)\|$
4.9	50,000	**x1**	11	58	0.309633	1.21E−09	47	97	0.880154	8.37E−09
		x2	11	58	0.318659	1.70E−09	48	99	0.972062	7.20E−09
		x3	11	58	0.303102	1.28E−09	47	97	0.828071	8.11E−09
		x4	10	53	0.290291	8.93E−09	47	97	1.127957	6.75E−09
		x5	11	58	0.322201	1.85E−09	48	99	0.940589	8.11E−09
		x6	11	58	0.320601	1.32E−09	47	97	0.976655	9.45E−09
		x7	11	58	0.339402	1.60E−09	48	99	0.826931	6.17E−09
	100,000	**x1**	11	58	0.840513	1.48E−09	48	99	2.345389	7.30E−09
		x2	11	58	0.891278	2.43E−09	49	101	2.434187	6.09E−09
		x3	11	58	1.068283	1.60E−09	48	99	2.331005	6.84E−09
		x4	11	58	0.885285	1.41E−09	47	97	2.813861	9.74E−09
		x5	11	58	0.862842	2.80E−09	49	101	2.411252	6.84E−09
		x6	11	58	0.855419	2.13E−09	48	99	2.321585	7.91E−09
		x7	11	58	0.860265	2.13E−09	48	99	2.322277	8.37E−09

Table 6 Summary of results from Tables 2 and 3 for the EDLM1 and DFPA [43] methods

Number of problems and percentage for which each method is a winner with respect to iterations, function evaluations and CPU time

Method	NIter	Percentage	NFev	Percentage	CPU time	Percentage
EDLM1	83	65.87	126	100	125	99.20
DFPA	41	32.54	–	–	1	0.8
Undecided	2	1.59	–	–	–	–

Table 7 Summary of results from Tables 4 and 5 for the EDLM2 and HZCGA [62] methods

Number of problems and percentage for which each method is a winner with respect to iterations, function evaluations and CPU time

Method	NIter	Percentage	NFev	Percentage	CPU time	Percentage
EDLM2	125	99.20	116	92.06	125	99.20
HZCGA	1	0.8	10	7.94	1	0.8

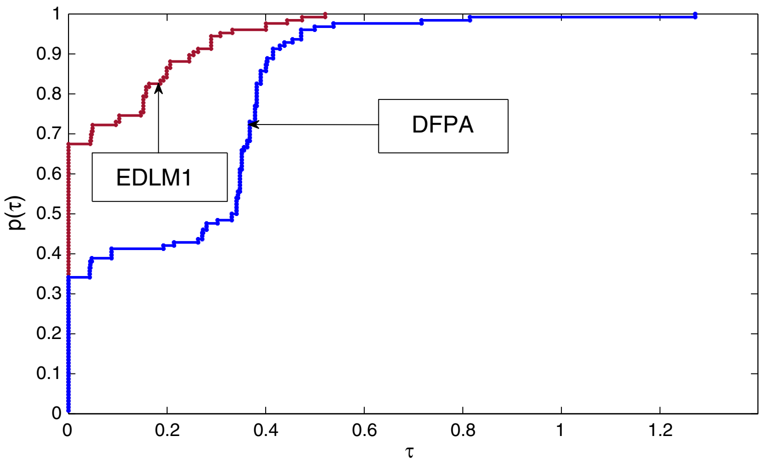

Fig. 1 Performance profile based on number of iterations for EDLM1 and DFPA [43] methods

Problem 4.6 [57] *Non-smooth function.*

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

Problem 4.7 [70] *Non-smooth function.*

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

Problem 4.8 [72] *Decretized Chandrasekhar equation.*

$$F_i(x) = x_i - \left(1 - \tfrac{c}{2n}\sum_{j=1}^{n}\frac{\mu_i x_j}{\mu_i + \mu_j}\right)^{-1}, \quad i = 1, 2, \ldots, n,$$

with $c \in [0, 1)$ *and* $\mu_i = \frac{i-0.5}{n}$, *for* $1 \le i \le n$, (*c is taken as* 0.999).

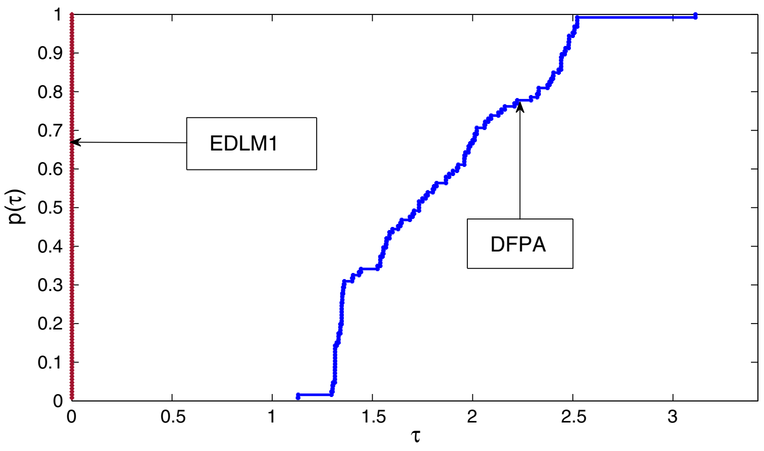

Fig. 2 Performance profile based on function evaluation for EDLM1 and DFPA [43] methods

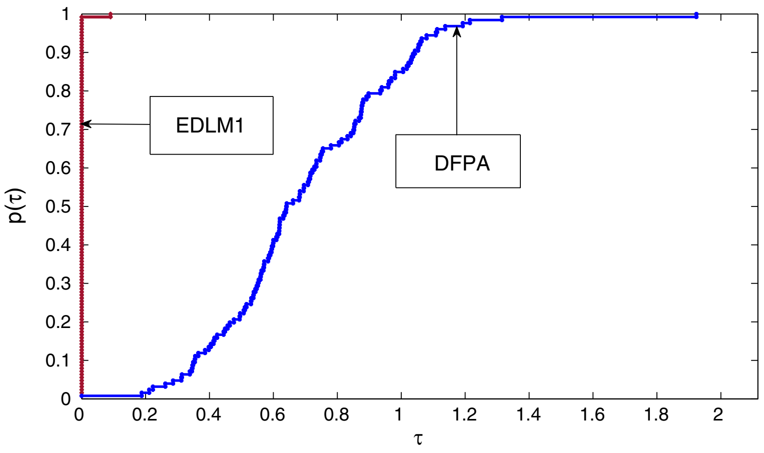

Fig. 3 Performance profile based on CPU time for EDLM1 and DFPA [43] methods

Problem 4.9 [59].

$$F_i(x) = x_i - \frac{1}{n}x_i^2 + \frac{1}{n}\sum_{i=1}^{n} x_i + i, \quad i = 1, 2, \ldots, n..$$

The summarized data in Table 6 shows that EDLM1 method performs better than the DFPA scheme as it solves more problems with least number of iterations, functions evaluations and CPU time respectively. In the iteration case, it is observed that the EDLM1 method successfully solved over 60% of the problem with least number of iterations compared to the DFPA method, which successfully solved 32.54%. The summarized table also shows that both methods competed in solving two of the problems by recording the same number of iteration, which is reported as undecided. In terms of functions evaluations and least CPU time, the EDLM1 method outperforms the DFPA scheme with a record of almost 100% in both cases. The summarized results in Table 7 shows that the second proposed method EDLM2 successfully solved 99.20% of the test problems with less iterations, 92.06% with

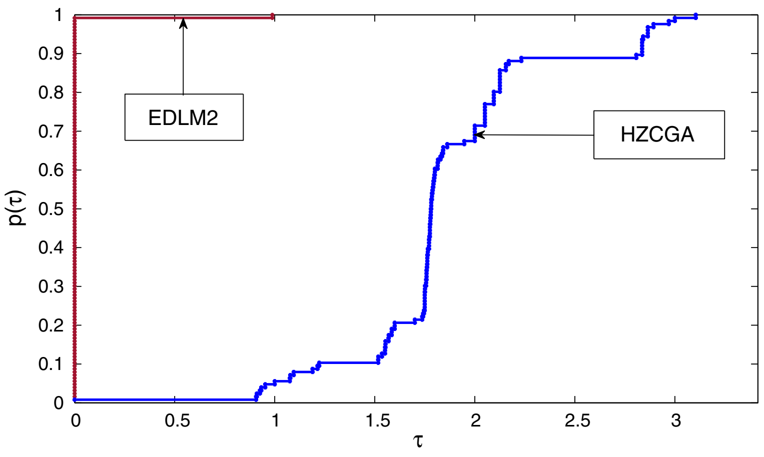

Fig. 4 Performance profile based on number of iterations for EDLM2 and HZCGA [62] methods

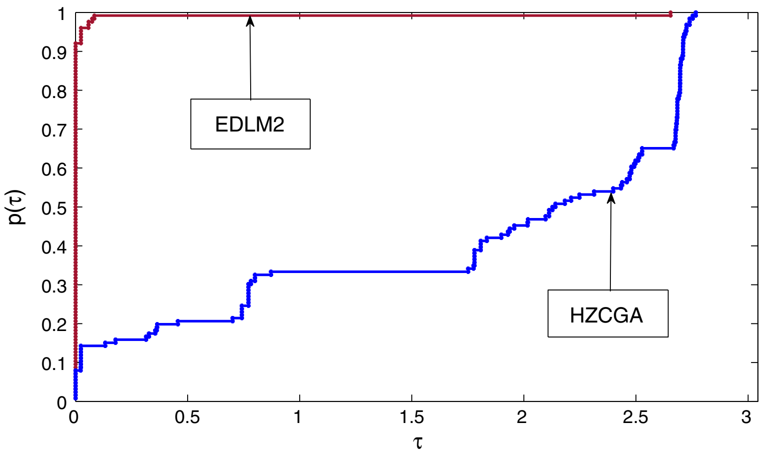

Fig. 5 Performance profile based on function evaluation time for EDLM2 and HZCGA [62] methods

least functions evaluations and 99.20% with less CPU time compared to the HZCGA method, which solves 0.8% with less number of iterations, 7.94% with less functions evaluations and 0.8% with least CPU time. The performances of the EDLM1 and EDLM2 methods is partly attributed to the enhancement due to the modified spectral parameters θ_k and $\bar{\theta}_k$ and the modified secant conditions involved in the composition of the methods.

In addition to the numerical results reported in Tables 2, 3, 4 and 5 and their summary in Tables 6 and 7, performance profile of the four methods are displayed in Figs. 1, 2, 3, 4, 5 and 6, using the profile of Dolan and Moré [15], which shows the performance of these methods relative to number of iterations, functions evaluations and CPU time respectively. For each of the methods, the fraction $P(\tau)$ of the problems for which it is within a factor τ of the best time is plotted. As indicated by the figures, both EDLM1 and EDLM2 methods are more effective and perform much better with respect to number of iterations, functions evaluations and CPU time than the DFPA and HZCGA methods . This can be seen as the top

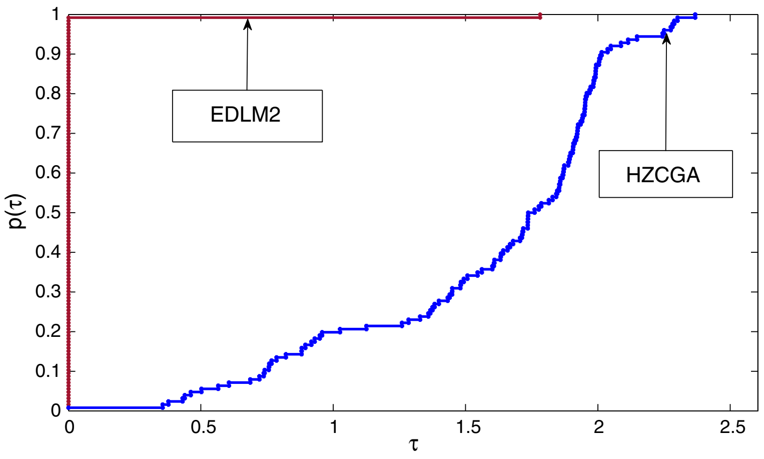

Fig. 6 Performance profile based on CPU time for EDLM2 and HZCGA [62] methods

curves in all six figures represents the EDLM1 and EDLM2 methods. Therefore, it is clear from Tables 2, 3, 4 and 5, the summarized result in Tables 6 and 7 and Figs. 1, 2, 3, 4, 5 and 6 that our methods are more effective than the DFPA and HZCGA methods and better for large-scale monotone nonlinear systems than the other two methods.

Furthermore, It is worth nothing that as part of applications of our proposed methods, they are quite effective for solving the discretized version of Chandrasekhar Integral equation (see Problem 4.8), which is important in the role it plays in radiactive transfer and transport theory [23]. The Chandrasekhar Integral equation is represented as

$$H(\mu) = 1 + H(\mu) \int_0^1 \frac{\mu}{\mu + t} \psi(t) H(t) dt. \tag{4.1}$$

In order to obtain an approximate solution of (4.1), it is discretized by a vector $\bar{x} \in R^n$, and the integrals are replaced by quadrature sums and the derivatives by difference quotients involving only the component of $\bar{x} \in R^n$ (see [24]). Thus, (4.1) becomes a problem of finding the solution of system of n nonlinear equations with n-unknowns as presented in Problem 4.8.

5 Conclusion

This paper presents two algorithms for solving large-scale monotone nonlinear equations. The methods can be considered as extensions of the effective Dai–Liao method [11] for unconstrained optimization, which are combined with the hyperplane projection technique of Solodov and Svaiter [48]. By employing modified spectral coefficients and carrying out eigenvalue analysis of two modified Dai–Liao search direction matrices, two improved versions of the classical Dai–Liao schemes are presented. For each of the new schemes, appropriate values are obtained for the DL nonnegative parameter "t", which plays important role in the scheme's performance. Apart from the low memory requirement and derivative-free structure of the two methods, our two choices of the parameter "t" also ensures the methods generate descent search directions and are globally convergent. Also, numerical comparisons with DFPA [43] and HZCGA [62] methods show that the proposed methods are promising.

Finally, as a future research, we anticipate extending our methods to nonlinear systems with convex constrained and applying them to solve signal processing and image de-blurring problems in compressive sensing.

References

1. Andrei, N.: Open problems in conjugate gradient algorithms for unconstrained optimization. Bull. Malays. Math. Sci. Soc. **34**(2), 319–330 (2011)
2. Andrei, N.: Accelerated adaptive Perry conjugate gradient algorithms based on the self-scaling BFGS update. J. Comput. Appl. Math. **325**, 149–164 (2017)
3. Arazm, M.R., Babaie-Kafaki, S., Ghanbari, R.: An extended Dai-Liao conjugate gradient method with global convergence for nonconvex functions. Glasnik matematic **52**(72), 361–375 (2017)
4. Babaie-Kafaki, S., Ghanbari, R., Mahdavi-Amiri, N.: Two new conjugate gradient methods based on modified secant equations. J. Comput. Appl. Math. **234**(5), 1374–1386 (2010)
5. Babaie-Kafaki, S., Ghanbari, R.: A descent family of Dai-Liao conjugate gradient methods. Optim. Methods Softw. **29**(3), 583–591 (2013)
6. Babaie-Kafaki, S., Ghanbari, R.: The Dai-Liao nonlinear conjugate gradient method with optimal parameter choices. Eur. J. Oper. Res. **234**, 625–630 (2014)
7. Babaie-Kafaki, S., Ghanbari, R.: Two optimal Dai-Liao conjugate gradient methods. Optimization **64**, 2277–2287 (2015)
8. Bouaricha, A., Schnabel, R.B.: Tensor methods for large sparse systems of nonlinear equations. Math. Program. **82**, 377–400 (1998)
9. Broyden, C.G.: A class of methods for solving nonlinear simultaneous equations. Math. Comput. **19**, 577–593 (1965)
10. Cheng, W.: A PRP type method for systems of monotone equations. Math. Comput. Model. **50**, 15–20 (2009)
11. Dai, Y.H., Liao, L.Z.: New conjugacy conditions and related nonlinear conjugate gradient methods. Appl. Math. Optim. **43**(1), 87–101 (2001)
12. Dai, Y.H., Yuan, Y.X.: Nonlinear Conjugate Gradient Methods. Shanghai Scientific and Technical Publishers, Shanghai (2000)
13. Dai, Z., Chen, X., Wen, F.: A modified Perry's conjugate gradient method-based derivative-free method for solving large-scale nonlinear monotone equation. Appl. Math. Comput. **270**, 378–386 (2015)
14. Dauda, M.K., Mamat, M., Mohamed, M.A., Waziri, M.Y.: Improved quasi-Newton method via SR1 update for solving symmetric systems of nonlinear equations. Malay. J. Fundam. Appl. Sci. **15**(1), 117–120 (2019)
15. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2, Ser. A), 201–2013 (2002)
16. Fasano, G., Lampariello, F., Sciandrone, M.: A truncated nonmonotone Gauss–Newton method for large-scale nonlinear least-squares problems. Comput. Optim. Appl. **34**, 343–358 (2006)
17. Fatemi, M.: An optimal parameter for Dai-Liao family of conjugate gradient methods. J. Optim. Theory Appl. **169**(2), 587–605 (2016)
18. Ford, J.A., Moghrabi, I.A.: Multi-step quasi-Newton methods for optimization. J. Comput. Appl. Math. **50**(1–3), 305–323 (1994)
19. Ford, J.A., Narushima, Y., Yabe, H.: Multi-step nonlinear conjugate gradient methods for unconstrained minimization. Comput. Optim. Appl. **40**(2), 191–216 (2008)
20. Grippo, L., Lampariello, F., Lucidi, S.: A nonmonotone linesearch technique for Newton's method. SIAM J. Numer. Anal. **23**, 707–716 (1986)
21. Hager, W.W., Zhang, H.: A survey of nonlinear conjugate gradient methods. Pac. J. Optim. **2**(1), 35–58 (2006)
22. Hestenes, M.R., Stiefel, E.L.: Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bur. Stand. **49**, 409–436 (1952)
23. Hively, G.A.: On a class of nonlinear integral equations arising in transport theory. SIAM J. Math. Anal. **9**(5), 787–792 (1978)
24. James, M.O., Werner, C.R.: On discretization and differentiation of operators with application to Newton's method. SIAM J. Numer. Anal. **3**(1), 143–156 (1966)
25. Kanzow, C., Yamashita, N., Fukushima, M.: Levenberg-Marquardt methods for constrained nonlinear equations with strong local convergence properties. J. Comput. Appl. Math. **172**, 375–397 (2004)

26. Khoshgam, Z., Ashrafi, A.: A new modidified scaled conjugate gradient method for large-scale unconstrained optimization with non-convex objective function. Optim. Methods Softw. (2018). https://doi.org/10.1080/10556788.2018.1457152
27. Kincaid, D., Cheney, W.: Numerical Analysis. Brooks/Cole Publishing Company, Pacific Grove (1991)
28. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Q. Appl. Math. **2**, 164–166 (1944)
29. La Cruz, W., Martinez J.M., Raydan M.: Spectral residual method without gradient information for solving large-scale nonlinear systems: theory and experiments, Citeseer, Technical Report RT-04-08(2004)
30. La Cruz, W., Raydan, M.: Nonmonotone spectral methods for large-scale nonlinear systems. Optim. Methods Softw. **18**, 583–599 (2003)
31. Li, D.H., Fukushima, M.: A globally and superlinearly convergent Gauss–Newton-based BFGS method for symmetric nonlinear equations. SIAM J. Numer. Anal. **37**(1), 152–172 (2000)
32. Li, D.H., Fukushima, M.: A derivative-free linesearch and global convergence of Broyden-like method for nonlinear equations. Optim. Methods Softw. **13**, 583–599 (2000)
33. Li, G., Tang, C., Wei, Z.: New conjugacy condition and related new conjugate gradient methods for unconstrained optimization. J. Comput. Appl. Math. **202**(2), 523–539 (2007)
34. Liu, D.Y., Shang, Y.F.: A new Perry conjugate gradient method with the generalized conjugacy condition. In: 2010 International Conference on Computational Intelligence and Software Engineering, Wuhan, pp. 1–4 (2010). https://doi.org/10.1109/CISE.2010.5677114
35. Liu, D.Y., Xu, G.Q.: A Perry descent conjugate gradient method with restricted spectrum, pp. 1–19. Optimization Online, Nonlinear Optimization (unconstrained optimization) (2011)
36. Liu, J.K., Li, S.J.: A projection method for convex constrained monotone nonlinear equations with applications. Comput. Math. Appl. **70**(10), 2442–2453 (2015)
37. Livieris, I.E., Pintelas, P.: Globally convergent modified Perrys conjugate gradient method. Appl. Math. Comput. **218**, 9197–9207 (2012)
38. Livieris, I.E., Pintelas, P.: A new class of spectral conjugate gradient methods based on a modified secant equation for unconstrained optimization. J. Comput. Appl. Math. **239**, 396–405 (2013)
39. Livieris, I.E., Pintelas, P.: A descent Dai-Liao conjugate gradient method based on a modified secant equation and its global convergence. ISRN Computational Mathematics (2012)
40. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. SIAM J. Appl. Math. **11**, 431–441 (1963)
41. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
42. Ortega, J.M., Rheinboldt, W.C.: Iterative Solution of Nonlinear Equations in Several Variables. Academic Press, New York (1970)
43. Peiting, G., Chuanjiang, H.: A derivative-free three-term projection algorithm involving spectral quotient for solving nonlinear monotone equations. Optim. J. Math. Program. Oper. Res. **67**, 1–18 (2018)
44. Perry, A.: A modified conjugate gradient algorithm. Oper. Res. Tech. Notes. **26**(6), 1073–1078 (1978)
45. Polak, B.T.: The conjugate gradient method in extreme problems. USSR Comput. Math. Math. Phys. **4**, 94–112 (1969)
46. Polak, E., Ribiere, G.: Note sur la convergence de methodes de directions conjugutes. Rev. Fr. Inform. Rech. Oper. **16**, 35–43 (1969)
47. Solodov, V.M., Iusem, A.N.: Newton-type methods with generalized distances for constrained optimization. Optimization **41**(3), 257–27 (1997)
48. Solodov, M.V., Svaiter, B.F.: A globally convergent inexact Newton method for systems of monotone equations. In: Fukushima, M., Qi, L. (eds.) Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods, pp. 355–369. Springer, Boston, MA (1998)
49. Sun, W., Yuan, Y.X.: Optimization Theory and Methods: Nonlinear Programming. Springer, New York (2006)
50. Sun, Z., Li, H., Wang, J., Tian, Y.: Two modified spectral conjugate gradient methods and their global convergence for unconstrained optimization. Int. J. Comput. Math. **95**(1), 1–23 (2017)
51. Tong, X.J., Qi, L.: On the convergence of a trust-region method for solving constrained nonlinear equations with degenerate solutions. J. Optim. Theory Appl. **123**(1), 187–211 (2004)
52. Waziri, M.Y., Ahmed, K., Sabi'u, J.: A family of Hager–Zhang conjugate gradient methods for system of monotone nonlinear equations. Appl. Math. Comput. **361**, 645–660 (2019)
53. Waziri, M.Y., Ahmed, K., Sabi'u, J.: A Dai-Liao conjugate gradient method via modified secant equation for system of nonlinear equations. Arab. J. Math. (2019). https://doi.org/10.1007/s40065-019-0264-6(2019)
54. Waziri, M.Y., Leong, W.J., Hassan, M.A.: Jacobian free-diagonal Newton's method for nonlinear systems with singular Jacobian. Malays. J. Math. Sci. **5**(2), 241–255 (2011)

55. Waziri, M.Y., Sabiu, J.: A Derivative-free conjugate gradient method and its global convergence for solving symmetric nonlinear equations. Hindawi Publ. Corp. Int. J. Math. Math. Sci. **2015**, 8 (2015)
56. Wei, Z., Li, G., Qi, L.: New quasi-Newton methods for unconstrained optimization problems. Appl. Math. Comput. **175**(2), 1156–1188 (2006)
57. Xiao, Y., Zhu, H.: A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. J. Math. Anal. Appl. **405**, 310–319 (2013)
58. Yabe, H., Takano, M.: Global convergence properties of nonlinear conjugate gradient methods with modified secant condition. Comput. Optim. Appl. **28**(2), 203–225 (2004)
59. Yan, Q.R., Peng, X.Z., Li, D.H.: A globally convergent derivative-free method for solving large-scale nonlinear monotone equations. J. Comput. Appl. Math. **234**, 649–657 (2010)
60. Yu, G.: A derivative-free method for solving large-scale nonlinear systems of equations. J. Ind. Manag. Optim. **6**, 149–160 (2010)
61. Yu, G.: Nonmonotone spectral gradient-type methods for large-scale unconstrained optimization and nonlinearsystems of equations. Pac. J. Optim. **7**, 387–404 (2011)
62. Yuan, G., Wang, B., Sheng, Z.: The Hager–Zhang conjugate gradient algorithm for large-scale nonlinear equations. Int. J. Comput. Math. **96**(8), 1533–1547 (2018). https://doi.org/10.1080/00207160.2018.1494825
63. Yuan, Y.X.: A modified BFGS algorithm for unconstrained optimization. IMA J. Numer. Anal. **11**, 325–332 (1991)
64. Yuan, Y.: Subspace methods for large scale nonlinear equations and nonlinear least squares. Optim. Eng. **10**, 207–218 (2009)
65. Yuan, G.L., Wei, Z.X., Lu, X.W.: A BFGS trust-region method for nonlinear equations. Computing **92**(4), 317–333 (2011)
66. Yuan, G., Zhang, M.: A three term Polak–Ribiere–Polyak conjugate gradient algorithm for large-scale nonlinear equations. J. Comput. Appl. Math. **286**, 186–195 (2015)
67. Zhang, J.Z., Deng, N.Y., Chen, L.H.: New quasi-Newton equation and related methods for unconstrained optimization. J. Optim. Theory Appl. **102**(1), 147–157 (1999)
68. Zhang, J., Xu, C.: Properties and numerical performance of quasi-Newton methods with modified quasi-Newton equations. J. Comput. Appl. Math. **137**(2), 269–278 (2001)
69. Zhang, J., Wang, Y.: A new trust region method for nonlinear equations. Math. Methods Oper. Res. **58**, 283–298 (2003)
70. Zhang, L., Zhou, W.: Spectral gradient projection method for solving nonlinear monotone equations. J. Comput. Appl. Math. **196**, 478–484 (2006)
71. Zhao, Y.B., Li, D.: Monotonicity of fixed point and normal mappings associated with variational inequality and its application. SIAM J. Optim. **11**, 962–973 (2001)
72. Zhou, W., Shen, D.: Convergence properties of an iterative method for solving symmetric non-linear equations. J. Optim. Theory Appl. **164**(1), 277–289 (2015)
73. Zhou, W., Li, D.: Limited memory BFGS method for nonlinear monotone equations. J. Comput. Math. **25**(1), 89–96 (2007)
74. Zhou, W., Wang, F.: A PRP-based residual method for large-scale monotone nonlinear equations. Appl. Math. Comput. **261**, 1–7 (2015)
75. Zhou, W., Zhang, L.: A nonlinear conjugate gradient method based on the MBFGS secant condition. Optim. Methods Softw. **21**(5), 707–714 (2006)