RAIRO-Oper. Res. 55 (2021) S1113–S1127
https://doi.org/10.1051/ro/2020061

RAIRO Operations Research
www.rairo-ro.org

AN EFFICIENT THREE-TERM CONJUGATE GRADIENT-TYPE ALGORITHM FOR MONOTONE NONLINEAR EQUATIONS

Jamilu Sabi'u and Abdullah Shah*

Abstract. In this article, we proposed two Conjugate Gradient (CG) parameters using the modified Dai–Liao condition and the descent three-term CG search direction. Both parameters are incorporated with the projection technique for solving large-scale monotone nonlinear equations. Using the Lipschitz and monotone assumptions, the global convergence of methods has been proved. Finally, numerical results are provided to illustrate the robustness of the proposed methods.

Mathematics Subject Classification. 90C30, 90C26.

Received April 7, 2020. Accepted June 6, 2020.

1. Introduction

This paper will address the problem

$$F(x) = 0, \tag{1.1}$$

where $F : R^n \to R^n$ is continuous and monotone function. The nonlinear monotone equations arise in different applications, such as Bregman distances [17], chemical equilibrium systems [22], financial forecasting problems [9] and signal reconstruction problems in compressive sensing [12]. Also, some variational inequality problems can be transformed into a system of monotone nonlinear equations [31].

The Newton method, the quasi-Newton methods and their variants are considered to be efficient methods for solving (1.1) despite the Jacobian inverse or its approximate requirement, see [10, 25, 33, 34]. However, these methods are not suitable for large-scale problems due to the computing and storage of the Jacobian matrix or its approximate for each iteration [8, 20, 29]. Nevertheless, in an effort to solve large-scale monotone nonlinear equations, Zhang and Zhou [30] proposed a spectral approach for (1.1) that combines the two-point gradient method [3] with the projection method [27]. The global convergence of the method [30] is provided using the monotone and Lipschitz continuous assumptions. The spectral method [30] has fairly low computational costs because it does not require the computing and storage of the Jacobian matrix or its approximation at each iteration.

Conjugate gradient methods are considered to be the most reliable numerical methods for solving large-scale problems due to their low memory requirements and good global convergence properties. They are iterative

Keywords. Monotone equations, three-term conjugate gradient method, conjugacy condition, descent condition.

Department of Mathematics, COMSATS University Islamabad, Park Road, Islamabad 44000, Pakistan.
*Corresponding author: abdullah_shah@comsats.edu.pk

Article published by EDP Sciences

methods that generate a sequence of solutions

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

where x_{k-1} is the initial approximation, α_k is the positive step length to be determine using an appropriate line search and d_k is the CG search direction defined by

$$d_k = -F(x_k), \quad k = 0, \quad d_k = -F(x_k) + \beta_k d_{k-1} \quad k \geq 1, \tag{1.3}$$

with β_k known as the CG update and it is the parameter that characterizes the CG method. The widely known CG update included the Hestenes–Stiefel (HS) formula [15], *i.e.*,

$$\beta_k^{\mathrm{HS}} = \frac{F_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \tag{1.4}$$

where $F_k = F(x_k)$ and $y_{k-1} = F_k - F_{k-1}$. However, the numerical performance of the CG methods depended on the appropriate choice of the parameter β_k [6]. Some important results on the global convergence of the CG methods were reviewed in [14,23]. Dai and Liao [5] incorporated the second-order information into CG method and proposed the following condition

$$d_k^T y_{k-1} = -tF_k^T s_{k-1}, \tag{1.5}$$

where t is a nonnegative scalar and $s_k = x_k - x_{k-1}$. However, in order to guarantee that the search direction d_k satisfies condition (1.5), they used the CG direction (1.3) into (1.5) and derived the following update

$$\beta_k^{\mathrm{DL}} = \frac{F_k^T (y_{k-1} - ts_{k-1})}{d_{k-1}^T y_{k-1}}. \tag{1.6}$$

The choice of the parameter $t \geq 0$ is considered by Andrei [1] to be an open problem for nonlinear CG method. This motivated Babaie-Kafaki and Ghanbari [2], and they proposed the following optimal choices for the nonnegative parameter "t" in the β_k^{DL}:

$$t_k^1 = \frac{\|y_{k-1}\|}{\|s_{k-1}\|}, \tag{1.7}$$

and

$$t_k^2 = \frac{y_{k-1}^T s_{k-1}}{\|s_{k-1}\|^2} + \frac{\|y_{k-1}\|}{\|s_{k-1}\|}. \tag{1.8}$$

By several numerical tests, it has proved that the choices t_k^1 and t_k^2 are robust compared to some other CG algorithms. Narushima *et al.* [24] suggested a descent three-term CG method with the search direction defined by

$$d_k = -F_k, \quad k = 0, \quad d_k = -F_k + \beta_k \left(F_k^T p_k\right)^{\dagger} \left\{\left(F_k^T p_k\right) d_{k-1} - \left(F_k^T d_{k-1}\right) p_k\right\}, \quad k \geq 1, \tag{1.9}$$

where p_k is any vector in R^n, and

$$A^{\dagger} = \begin{cases} \frac{1}{A}, & \text{if} \quad A \neq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1.10}$$

Furthermore, among the nice properties of the search direction (1.9) is that, it satisfies the following condition

$$F_k^T d_k = -\|F_k\|^2, \quad \forall k \geq 0, \tag{1.11}$$

which is independent of the line search and choices of β_k and p_k. This shows that the sufficient descent condition is always satisfied for $r = 1$, *i.e.*,

$$F_k^T d_k \leq -r\|F_k\|^2, \quad \forall k \geq 0. \tag{1.12}$$

Many other conjugate gradient methods have been combined with the projection method [27] to solve the large-scale monotone nonlinear equations [4, 7, 8, 19–21, 26, 29].

The main aim of this paper is to develop a robust three-term conjugate gradient method such that the β_k update can be derived by using the Dai–Liao condition (1.5) with the three-term CG search direction (1.9). The resulting CG update will be used to develop an effective CG method for monotone nonlinear equations using the projection technique.

The rest of the paper is structured as follows: In Section 2, we suggest a new β_k for the three-term (1.9) method based on the Dai–Liao condition. The convergence of algorithms is shown in Section 3. The numerical results are presented in Section 4. Section 5 is the conclusion.

2. Three-term conjugate gradient-type algorithm

This section will present two formulas for the update β_k based on the second Dai–Liao condition (1.5) and the three-term CG method (1.9). Now, by incorporating the optimal value (1.7) into the condition (1.5), we get

$$d_k^T y_{k-1} = -W_k, \tag{2.1}$$

where $W_k = \frac{\|y_{k-1}\|}{\|s_{k-1}\|} F_k^T s_{k-1}$. Now substituting (1.9) into (2.1), we have

$$\left(-F_k + \beta_k d_{k-1} - \beta_k \frac{F_k^T d_{k-1}}{F_k^T p_k} p_k\right)^T y_{k-1} = -W_k. \tag{2.2}$$

Then the above equation (2.2) can be rewritten as

$$\beta_k \left(d_{k-1} - \frac{F_k^T d_{k-1}}{F_k^T p_k} p_k\right)^T y_{k-1} = F_k^T y_{k-1} - W_k, \tag{2.3}$$

where $y_{k-1} = F_k - F_{k-1}$. Now, by using the definition of y_{k-1} in (2.3) gives

$$\beta_k \left(-F_{k-1}^T d_{k-1} + \frac{F_{k-1}^T p_k}{F_k^T p_k} F_k^T d_{k-1}\right) = F_k^T y_{k-1} - W_k. \tag{2.4}$$

Applying condition (1.11) into (2.4) we have

$$\beta_k = \frac{F_k^T y_{k-1} - W_k}{\|F_{k-1}\|^2 + Q_k}, \tag{2.5}$$

where $Q_k = \frac{F_{k-1}^T p_k}{F_k^T p_k} F_k^T d_{k-1}$. We avoided an undefined denominator by modifying (2.5) as

$$\beta_k^1 = \frac{F_k^T y_{k-1} - W_k}{\|F_{k-1}\|^2 + \xi_k Q_k}. \tag{2.6}$$

Similarly, using the optimal choice (1.8) into the condition (1.5) and following the same procedure we obtain

$$\beta_k^2 = \frac{F_k^T y_{k-1} - H_k}{\|F_{k-1}\|^2 + \xi_k Q_k}, \tag{2.7}$$

where

$$H = \left(\frac{y_{k-1}^T s_{k-1}}{\|s_{k-1}\|^2} + \frac{\|y_{k-1}\|}{\|s_{k-1}\|}\right) F_k^T s_{k-1}. \tag{2.8}$$

We chose the parameter ξ_k such that

$$\xi_k = \begin{cases} \min\left\{1, -(1-\xi_0)\frac{\|F_{k-1}\|^2}{Q_k}\right\}, & \text{if} \quad Q_k < 0 \\ 1, & \text{otherwise,} \end{cases} \tag{2.9}$$

where $\xi_0 \in (0,1)$. Kobayashi *et al.* [18] guaranteed that the denominator of β_k^1 and β_k^2 always satisfy

$$\|F_{k-1}\|^2 + \xi_k Q_k \geq \xi_0 \|F_{k-1}\|^2. \tag{2.10}$$

Now, we assume that:

(i) Function F is monotone, that is,

$$(F(x) - F(y))^T (x-y) \geq 0, \quad \forall x, y \in R^n. \tag{2.11}$$

(ii) Function F is Lipschitz continuous, that is, for some $m > 0$:

$$\|F(x) - F(y)\| \leq m \|x-y\|, \quad \forall x, y \in R^n. \tag{2.12}$$

Furthermore, we adopted the concept of Solodov and Svaiter [27] to let the next iterate

$$x_{k+1} = x_k - \frac{F(w_k)^T (x_k - w_k)}{\|F(w_k)\|^2} F(w_k), \tag{2.13}$$

where $w_k = x_k + \alpha_k d_k$. Finally, the proposed algorithm is as follows:

Algorithm 2.1 (Efficient three-term CG (ETCG))**.**

Step 0. Select the starting point $x_0 \in R^n$, and initialize the constants $\gamma \in (0,1)$, $\epsilon, \tau, \delta \geq 0$. Set $k = 0$ and $p_k = F_k$.
Step 1. If $\|F_k\| \leq \epsilon$, stop, if not go to Step 2.
Step 2. Calculate the search direction

$$d_k = -F_k, \quad k = 0, \quad d_k = -F_k + \beta_k^1 \left(F_k^T p_k\right)^\dagger \left\{\left(F_k^T p_k\right) d_{k-1} - \left(F_k^T d_{k-1}\right) p_k\right\}, \quad k \geq 1,$$

or

$$d_k = -F_k, \quad k = 0, \quad d_k = -F_k + \beta_k^2 \left(F_k^T p_k\right)^\dagger \left\{\left(F_k^T p_k\right) d_{k-1} - \left(F_k^T d_{k-1}\right) p_k\right\}, \quad k \geq 1.$$

Step 3. Determine $\alpha_k = \max\left\{\tau\gamma^j : j = 0, 1, 2, \ldots\right\}$ such that

$$-F(x_k + \alpha_k d_k)^T d_k \geq \delta \alpha_k \|F(x_k + \alpha_k d_k)\| \|d_k\|^2. \tag{2.14}$$

Step 4. Let $w_k = x_k + \alpha_k d_k$, if $\|F(w_k)\| = 0$ stop, otherwise go to Step 5.
Step 5. Compute the projection of x_{k+1} using (2.13).
Step 6. Set $k = k+1$ and go to Step 1.

3. Convergence analysis

This section will provide the global convergence of Algorithms (2.1) using the monotonicity and Lipschitz assumptions. Since the search direction (1.9) satisfied the sufficient descent condition irrespective of the choice β_k, we proceed with our proof as follows

Lemma 3.1 ([27]). *Assume that F is monotone and $x, w \in R^n$ satisfy $F(y)^T(x-y) > 0$. Let*

$$x^+ = x - \frac{F(w)^T(x-w)}{\|F(w)\|^2}F(w). \tag{3.1}$$

For any $x^* \in R^n$ such that $F(x^*) = 0$,

$$\left\|x^+ - x^*\right\|^2 \leq \|x - x^*\|^2 - \left\|x^+ - x\right\|^2. \tag{3.2}$$

Lemma 3.2. *Let F be monotone and Lipschitz continuous and $\{x_k\}$ be generated by the Algorithm 2.1. Assume that the solution set of equation (1.1) is nonempty, for any x^* such that $F(x^*) = 0$, we have*

$$\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2 - \|x_{k+1} - x_k\|^2. \tag{3.3}$$

In fact, $\{x_k\}$ is bounded. This also holds that either $\{x_k\}$ is finite and the last iterate is a solution, or the sequence is infinite and $\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0$. In addition, $\{x_k\}$ converges to some x^* such that $F(x^*) = 0$.

Proof. First of all, if the Algorithm 2.1 ends at some k iteration, then $d_k = 0$, and we get $F(x_k) = 0$, which means that x_k is the solution. Now suppose $d_k \neq 0$ for all k, and then an infinite $\{x_k\}$ is generated. It is obvious from equation (2.14) that

$$F(w_k)^T(x_k - w_k) = -\alpha_k F(w_k)^T d_k \geq \delta\alpha_k^2 \|F(w_k)\| \|d_k\|^2 > 0. \tag{3.4}$$

Let x^* be any point such that $F(x^*) = 0$. By (2.13), (3.4) and Lemma 3.1, we get

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

Therefore the sequence $\{\|x_k - x^*\|\}$ is non-increasing and convergent, hence the sequence $\{x_k\}$ is bounded, and also

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0. \tag{3.6}$$

Now, from (2.13) and (3.4) we have

$$\|x_{k+1} - x_k\| = \frac{\left|F(w_k)^T(x_k - w_k)\right|}{\|F(w_k)\|} \geq \delta\alpha_k^2 \|d_k\|^2. \tag{3.7}$$

Hence, from the inequality (3.7) and (3.6) we have

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

From the continuity of F and the boundedness of $\{x_k\}$, it is clear that $\{x_k\}$ has some accumulation point $\hat{x}$ such that $F(\hat{x}) = 0$. Also by choosing $x^* = \hat{x}$ in (3.5), since x^* is arbitrary, then the sequence $\{\|x_k - \hat{x}\|\}$ converges. Hence, $\{x_k\}$ converges to $\hat{x}$. □

It follows from assumption (ii) that there is a positive constant $\kappa > 0$ such that

$$\|F(x_k)\| \leq \kappa, \quad \forall k \geq 0. \tag{3.9}$$

Remarks

(1) Using the definition of y_k and the Lipschitz continuity of F, we have

$$\|y_k\| = \|F(x_{k+1}) - F(x_k)\| \leq m \|x_{k+1} - x_k\|. \tag{3.10}$$

(2) Also by using the definition of y_k, s_k and the Lipschitz continuity assumption on F, we get

$$\begin{aligned} \left|y_k^T s_k\right| &= \left|(F(x_{k+1}) - F(x_k))^T (x_{k+1} - x_k)\right| \\ &\leq \|F_{k+1} - F_k\| \|x_{k+1} - x_k\| = m \|x_{k+1} - x_k\|^2. \end{aligned} \tag{3.11}$$

It follow from (3.9) to (3.11) that

$$|W_k| = \frac{\|y_{k-1}\|}{\|s_{k-1}\|} \left|F_k^T s_{k-1}\right| \leq \frac{\|y_{k-1}\|}{\|s_{k-1}\|} \|F_k\| \|s_{k-1}\| \leq m\kappa \|x_k - x_{k-1}\|, \tag{3.12}$$

and

$$\begin{aligned} |H_k| &= \left| \left(\frac{y_{k-1}^T s_{k-1}}{\|s_{k-1}\|^2} + \frac{\|y_{k-1}\|}{\|s_{k-1}\|} \right) F_k^T s_{k-1} \right| \\ &\leq \left(\frac{\|y_{k-1}\| \|s_{k-1}\|}{\|s_{k-1}\|^2} + \frac{\|y_{k-1}\|}{\|s_{k-1}\|} \right) \|F_k\| \|s_{k-1}\| \\ &\leq 2m\kappa \|x_k - x_{k-1}\|. \end{aligned} \tag{3.13}$$

Lemma 3.3. *Let our assumptions be maintained, and $\{x_k\}$ be generated by our Algorithm 2.1. If there exists a constant $\theta > 0$ such that $\|F_k\| \geq \theta$, then there exist $M > 0$ such that*

$$\|d_k\| \leq M, \quad \forall k. \tag{3.14}$$

Proof. Considering the choices of $p_k = F_k, \beta_k^1$ and β_k^2, then from (2.10), (3.9), (3.12), (3.13) and the definition of our direction, we have

$$\begin{aligned} \|d_k\| &= \left\| -F_k + \beta_k^1 d_{k-1} - \beta_k^1 \frac{F_k^T d_{k-1}}{F_k^T F_k} F_k \right\| \\ &\leq \|F_k\| + \frac{\|F_k\| \|y_{k-1}\| + |W_k|}{\xi_0 \|F_{k-1}\|^2} \|d_{k-1}\| + \frac{\|F_k\| \|y_{k-1}\| + |W_k|}{\xi_0 \|F_{k-1}\|^2} \frac{\|F_k\| \|d_{k-1}\|}{\|F_k\|^2} \|F_k\| \\ &= \|F_k\| + 2\frac{\|F_k\| \|y_{k-1}\| + |W_k|}{\xi_0 \|F_{k-1}\|^2} \|d_{k-1}\| \\ &\leq \kappa + \frac{4m\kappa \|x_k - x_{k-1}\|}{\xi_0 \theta^2} \|d_{k-1}\|. \end{aligned} \tag{3.15}$$

Now, from (3.6) there exist $c \in (0, 1)$ such that

$$\frac{4m\kappa \|x_k - x_{k-1}\|}{\xi_0 \theta^2} < c. \tag{3.16}$$

Therefore, for $k > k_0$ we have

$$\begin{aligned} \|d_k\| &\leq \kappa + c \|d_{k-1}\| \\ &\leq \kappa(1 + c + c^2 + \ldots + c^{k-k_0+1}) \|d_{k_0}\| \\ &\leq \frac{\kappa}{1-c} + \|d_{k_0}\|. \end{aligned} \tag{3.17}$$

Setting $M = \{\|d_1\|, \|d_2\|, \ldots, \|d_{k_0}\|, \frac{\kappa}{1-c} + \|d_{k_0}\|\}$, we get (3.14). It also follows, in a similar way, that our direction is also bounded by the choice of β_k^2. □

Theorem 3.4. *Suppose that $\{x_k\}$ is generated using Algorithm 2.1. Then*

$$\liminf_{k\to\infty} \|F_k\| = 0. \tag{3.18}$$

Proof. Suppose that (3.18) does not hold, that is, there exists a positive constant θ such that

$$\|F_k\| > \theta, \quad \forall k \geq 0. \tag{3.19}$$

Observe that, from the sufficient descent condition (1.12) and the Cauchy–Schwarz inequality we have

$$\|F_k\| \|d_k\| \geq -F_k d_k \geq \|F_k\|^2. \tag{3.20}$$

Therefore, from (3.20) we get

$$\|d_k\| \geq \|F_k\| > \theta > 0. \tag{3.21}$$

This together with result in (3.8) implies that

$$\lim_{k\to\infty} \alpha_k = 0. \tag{3.22}$$

From the definition of the line search procedure, $\alpha_k\gamma^{-1}$ does not satisfy (2.14), *i.e.*,

$$-F(x_k + \alpha_k\gamma^{-1}d_k)^T d_k < \delta\alpha_k\gamma^{-1} \left\|F(x_k + \alpha_k\gamma^{-1}d_k)\right\| \|d_k\|^2. \tag{3.23}$$

It is therefore obvious from the boundedness of $\{x_k\}$ that $\{x_k\}$ has some accumulation point $\hat{x}$ and an infinite index set Z_1 such that $\lim_{k\in Z_1} x_k = \hat{x}$. It follows from (3.14) that $\{d_k\}_{k\in Z_1}$ is bounded as well. Therefore, there exist an infinite index set $Z_2 \subset Z1$ and some accumulation point $\hat{d}$ such that $\lim_{k\in Z_2} d_k = \hat{d}$. Now, if we take the limit in (3.23), we get

$$-F(\hat{x})^T\hat{d} \leq 0. \tag{3.24}$$

Also taking the limit in sufficient descent condition (1.12), we get

$$-F(\hat{x})^T\hat{d} \geq 0. \tag{3.25}$$

This gives rise to a contradiction, so (3.18) holds, and the proof is complete. □

4. Numerical experiment

This section provide the numerical tests using the proposed Algorithm 2.1. The algorithm is coded in Matlab and comparison is provided in term efficiency with the NHZ derivative-free method [7] and the Self adaptive spectral conjugate gradient method for solving nonlinear monotone equations (SASCG) [19]. However, for the proposed algorithms we selected $\tau = 1$, $\gamma = 0.9$, $\delta = 0.0001$ and $\xi_0 = 0.06$. Although for both the NHZ and SASCG methods, we implemented the default parameters used in the respective papers. In addition, iteration is terminated if $\|F(x_k)\| \leq 10^{-11}$ or the number of iteration is higher than 1000 on the following test problems:

Problem 4.1 ([21])**.** The precise description of the $F(x)$ function is described as

$$F(x_i) = \exp(x_i) - 1, \quad \text{for} \quad i = 1, 2, 3, \ldots, n.$$

Problem 4.2 ([19]). The precise description of the function $F(x)$ is described as

$$\begin{aligned} F_1(x) &= hx_1 + x_2 - 1, \\ F_i(x) &= x_{i-1} + hx_i + x_{i-1} - 1, \quad \text{for} \quad i = 2, 3, \ldots, n-1, \quad h = 2.5, \\ F_n(x) &= x_{n-1} + hx_n - 1. \end{aligned}$$

Problem 4.3 ([26]). The precise description of the function $F(x)$ is described as

$$F(x_i) = x_i - \sin|x_i - 1|, \quad \text{for} \quad i = 1, 2, 3, \ldots, n.$$

Problem 4.4 ([26]). The precise description of the function $F(x)$ is described as

$$F(x_i) = 2x_i - \sin|x_i|, \quad \text{for} \quad i = 1, 2, 3, \ldots, n.$$

Problem 4.5 ([16]). The precise description of the function $F(x)$ is described as

$$\begin{aligned} F_1(x) &= x_1\left(x_1^2 + x_2^2\right) - 1, \\ F_i(x) &= x_i\left(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2\right) - 1, \quad \text{for} \quad i = 2, 3, \ldots, n-1, \\ F_n(x) &= x_n\left(x_{n-1}^2 + x_n^2\right). \end{aligned}$$

Problem 4.6 ([13]). The precise description of the function $F(x)$ is described as

$$F_i(x) = x_i - \left(1 - \frac{c}{2n}\sum_{j=1}^{n}\frac{\mu_i x_j}{\mu_i + \mu_j}\right)^{-1}, \quad \text{for} \quad i = 1, 2, \ldots, n, \quad \mu = \frac{i - 0.5}{n}, \quad c = 0.9.$$

Problem 4.7 ([28]). The precise description of the function $F(x)$ is described as

$$\begin{aligned} F(x_1) &= x_1 - \exp\left(\frac{\cos(x_1 + x_2)}{n+1}\right), \\ F(x_i) &= x_i - \exp\left(\frac{\cos(x_{i-1} + x_i + x_{i+1})}{n+1}\right), \quad \text{for} \quad i = 2, 3, \ldots, n-1, \\ F(x_n) &= x_n - \exp\left(\frac{\cos(x_{n-1} + x_n)}{n+1}\right). \end{aligned}$$

Problem 4.8 ([32,33]). The precise description of the function $F(x)$ is described as

$$F(x) = \begin{pmatrix} \frac{5}{2} & 1 & & & \\ 1 & \frac{5}{2} & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & \frac{5}{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \\ x_n \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ \vdots \\ \\ 1 \end{pmatrix}.$$

Tables 1–4 showed the numerical efficiency of the proposed methods compared to the NHZ method [7] and the SASCG method [19]. In our comparison, ITER is set to represent the number of iterations, TIME for the CPU time in the second, FVL for the number of function evaluations, and NORM to indicate the norm of the function evaluation at the stopping point. On the following initial guesses, we considered eight test problems, namely, $x_1 = (1, 1, \ldots, 1)$, $x_2 = \left(1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n}\right)$, $x_3 = (0.1, 0.1, \ldots, 0.1)$, $x_4 = \left(\frac{1}{n}, \frac{2}{n}, \ldots, 1\right)$, $x_5 = \left(1 - \frac{1}{n}, 1 - \frac{2}{n}, \ldots, 0\right)$, $x_6 = (-1, -1, \ldots, -1)$, $x_7 = \left(n - \frac{1}{n}, n - \frac{2}{n}, \ldots, n - 1\right)$ and $x_8 = \left(\frac{1}{2}, 1, \frac{2}{3}, \ldots, \frac{2}{n}\right)$.

TABLE 1. Numerical performance of the Algorithm 2.1 with the choices of β_k^1, β_k^2, NHZ method [7] and SASCG method [19].

Dimension	Initial point	Algorithm (β_k^1)				Algorithm (β_k^2)				NHZ				SASCG			
		ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM
Problem 4.1																	
50 000	x_1	12	42	0.023878	5.06E-12	12	42	0.018648	5.06E-12	100	397	1.241563	7.94E-12	37	233	0.212975	8.14E-12
	x_2	18	68	0.011068	3.45E-12	17	67	0.014021	2.45E-12	97	389	2.93197	7.79E-12	30	211	0.183455	8.68E-12
	x_3	11	34	0.005601	5.2E-12	11	34	0.00588	5.2E-12	98	390	1.242592	8.09E-12	36	229	0.215242	6.26E-12
	x_4	23	104	0.005954	6.53E-12	27	151	0.007233	1.99E-12	101	398	3.025503	8.13E-12	35	226	0.211963	9.03E-12
	x_5	26	134	0.007001	7.92E-12	30	241	0.011883	1.4E-12	101	398	3.053391	8.13E-12	35	226	0.225004	9.03E-12
	x_6	13	37	0.004515	1.79E-12	13	37	0.003916	1.79E-12	102	401	1.294912	9.28E-12	34	219	0.204756	7.05E-12
	x_7	26	134	0.010372	7.92E-12	30	241	0.013057	1.4E-12	101	398	3.261495	8.13E-12	35	226	0.222699	9.03E-12
	x_8	29	190	0.012262	5.41E-12	17	106	0.007184	1.92E-12	99	399	3.395135	8.34E-12	32	220	0.20428	9.8E-12
100 000	x_1	13	45	0.01028	1.6E-12	13	45	0.007919	1.6E-12	101	401	2.933818	8.64E-12	38	236	0.46628	5.2E-12
	x_2	20	74	0.01486	1.9E-12	17	67	0.01061	8.45E-12	97	389	6.101088	7.79E-12	30	211	0.366805	8.68E-12
	x_3	12	37	0.00942	1.64E-12	12	37	0.007396	1.64E-12	99	394	3.01148	8.85E-12	36	229	0.441187	8.85E-12
	x_4	37	220	0.034193	1.27E-12	45	496	0.046788	1.35E-12	102	402	6.263199	8.86E-12	36	229	0.451104	5.75E-12
	x_5	24	100	0.019468	5.77E-12	41	471	0.045369	1.36E-12	102	402	6.345785	8.86E-12	36	229	0.439895	5.75E-12
	x_6	13	37	0.010442	5.64E-12	13	37	0.007211	5.65E-12	97	380	2.705291	7.79E-12	35	222	0.454007	4.49E-12
	x_7	24	100	0.019606	5.77E-12	41	471	0.042817	1.3E-12	102	402	6.358489	8.86E-12	36	229	0.48022	5.75E-12
	x_8	28	181	0.028915	2.08E-12	19	113	0.014074	1.72E-12	99	399	6.202778	8.34E-12	32	220	0.481163	9.8E-12
Problem 4.2																	
50 000	x_1	58	953	0.072278	6.28E-12	69	1143	0.065879	7.98E-12	174	891	9.535615	8.95E-12	213	1239	3.112138	8.9E-12
	x_2	59	971	0.073565	4.1E-12	69	1141	0.064185	9.6E-12	178	901	9.627862	9.89E-12	69	421	0.938949	9.85E-12
	x_3	53	873	0.067795	6.95E-12	68	1124	0.065285	3.38E-12	181	896	9.852347	9.77E-12	70	425	0.971654	7.6E-12
	x_4	53	876	0.064057	9.42E-12	67	1110	0.122655	4.6E-12	179	913	9.69382	9.72E-12	77	473	1.083314	9.87E-12
	x_5	51	844	0.063127	9.94E-12	67	1110	0.112762	4.39E-12	179	910	9.927599	9.1E-12	77	473	1.078823	9.86E-12
	x_6	62	1024	0.074996	8.91E-12	72	1191	0.131133	6.88E-12	186	932	10.02757	9.37E-12	71	433	0.948575	9.05E-12
	x_7	51	844	0.063692	9.94E-12	67	1110	0.063645	4.4E-12	179	911	10.08346	9.49E-12	77	473	1.070865	9.86E-12
	x_8	66	1086	0.079644	7.15E-12	68	1126	0.063842	5.52E-12	182	922	10.33873	8.92E-12	70	427	0.926785	7.82E-12
100 000	x_1	62	1018	0.151247	6.02E-12	71	1175	0.128196	8.42E-12	174	879	19.38292	9.71E-12	69	429	1.946296	8.89E-12
	x_2	59	968	0.142828	9.06E-12	71	1174	0.180258	4.23E-12	179	893	19.31256	9.05E-12	70	427	1.936046	9.38E-12
	x_3	56	923	0.133078	4.74E-12	69	1140	0.16548	3.87E-12	183	926	20.35279	9.92E-12	69	420	1.943101	6.95E-12
	x_4	63	1036	0.150231	4.23E-12	68	1126	0.159079	6.64E-12	178	910	19.49065	9.45E-12	174	1020	5.203251	8.8E-12
	x_5	69	1131	0.170577	6.12E-12	68	1126	0.160867	6.64E-12	180	912	19.86274	8.43E-12	174	1020	5.296807	8.8E-12
	x_6	67	1102	0.156554	5.2E-12	74	1223	0.166872	7.98E-12	185	932	20.08118	9.39E-12	76	463	2.198901	8.51E-12
	x_7	69	1131	0.16089	6.12E-12	68	1126	0.15479	6.64E-12	180	894	19.62889	9.02E-12	174	1020	5.536953	8.8E-12
	x_8	79	1299	0.177904	9.9E-12	70	1158	0.159177	6.77E-12	181	919	19.92129	9.14E-12	69	420	1.980738	7.53E-12

TABLE 2. Numerical performance of the Algorithm 2.1 with the choices of β_k^1, β_k^2, NHZ method [7] and SASCG method [19].

Dimension	Initial point	Algorithm (β_k^1)				Algorithm (β_k^2)				NHZ				SASCG			
		ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM
Problem 4.3																	
50 000	x_1	12	109	0.009635	9.53E-12	12	109	0.008861	9.53E-12	46	183	0.968019	9.04E-12	12	168	0.12183	1.14E-12
	x_2	16	140	0.01158	4.77E-12	14	124	0.010001	7.01E-12	46	183	1.805152	6.99E-12	13	172	0.144853	3.05E-12
	x_3	11	98	0.008881	6.06E-12	11	98	0.008309	6.06E-12	44	174	0.910601	8.07E-12	13	172	0.143557	1.74E-12
	x_4	15	135	0.011604	1.86E-12	14	126	0.010497	3.97E-12	46	183	1.80926	9.79E-12	12	168	0.130774	7.68E-12
	x_5	15	135	0.01123	2.1E-12	14	126	0.01102	5.76E-12	46	183	1.825944	9.79E-12	12	168	0.128442	7.68E-12
	x_6	12	104	0.009421	5.19E-12	12	104	0.009719	5.19E-12	48	190	1.038401	7.15E-12	13	171	0.141158	6.19E-12
	x_7	15	135	0.010765	2.1E-12	14	126	0.010048	5.17E-12	46	183	1.746782	9.79E-12	12	168	0.161798	7.68E-12
	x_8	18	157	0.012151	7.04E-12	14	125	0.010696	9E-12	47	187	1.809052	9.12E-12	13	172	0.138182	3.04E-12
100 000	x_1	13	118	0.019286	3.15E-12	13	118	0.01757	3.15E-12	47	187	1.739121	7.07E-12	12	168	0.280865	1.61E-12
	x_2	16	141	0.021518	3.35E-12	15	133	0.01968	1.39E-12	46	182	3.407617	6.01E-12	13	172	0.265705	4.31E-12
	x_3	12	107	0.016837	2E-12	12	107	0.016419	2E-12	45	178	1.647128	6.34E-12	13	172	0.267549	2.46E-12
	x_4	15	135	0.021757	6.24E-12	15	135	0.019672	5.94E-12	47	187	3.269508	7.67E-12	13	172	0.261885	8.35E-13
	x_5	15	135	0.020934	6.31E-12	15	135	0.019532	4.27E-12	47	187	3.298402	7.67E-12	13	172	0.270475	8.35E-13
	x_6	13	113	0.019323	1.72E-12	13	113	0.015986	1.72E-12	49	194	1.816159	5.62E-12	13	171	0.259885	8.76E-12
	x_7	15	135	0.021985	6.31E-12	15	135	0.020804	5.13E-12	47	187	3.324447	7.67E-12	13	172	0.280997	8.35E-13
	x_8	22	187	0.028933	1.58E-12	23	193	0.029482	7.1E-12	48	191	3.565637	7.1E-12	13	172	0.258293	4.31E-12
Problem 4.4																	
50 000	x_1	12	38	0.00683	6.12E-12	12	38	0.005683	6.12E-12	98	388	1.968283	8.14E-12	38	235	0.340916	8.86E-12
	x_2	14	160	0.012508	2.05E-12	20	225	0.015217	5.64E-12	98	393	3.723317	9.33E-12	32	217	0.268808	7.74E-12
	x_3	11	34	0.006156	9.85E-12	11	34	0.004838	9.85E-12	99	394	1.976293	9.72E-12	36	229	0.315441	8.09E-12
	x_4	18	210	0.015279	8.24E-13	21	239	0.015112	1.01E-12	103	428	4.069873	8.81E-12	38	235	0.323797	6.59E-12
	x_5	19	221	0.015186	8.79E-13	21	239	0.016084	9.74E-13	102	423	3.959303	9.31E-12	38	235	0.328913	6.59E-12
	x_6	10	130	0.009703	2.25E-12	10	130	0.009379	2.25E-12	62	306	1.566793	7.47E-12	37	341	0.480497	5.34E-12
	x_7	19	221	0.014531	8.79E-13	21	239	0.015365	9.74E-13	102	423	3.967956	9.31E-12	38	235	0.328609	6.59E-12
	x_8	23	246	0.016813	2.3E-12	22	244	0.016938	1.11E-12	100	401	3.797342	9.05E-12	32	217	0.275852	5.99E-12
100 000	x_1	13	41	0.011691	1.94E-12	13	41	0.010375	1.94E-12	99	392	3.697042	8.87E-12	39	238	0.710392	5.64E-12
	x_2	14	161	0.022202	8.06E-13	20	225	0.0295	5.65E-12	98	393	6.954859	9.33E-12	32	217	0.517626	7.74E-12
	x_3	12	37	0.010159	3.11E-12	12	37	0.010046	3.11E-12	101	402	3.955294	8.16E-12	37	232	0.606095	5.15E-12
	x_4	18	210	0.028669	2.6E-12	21	239	0.031041	3.15E-12	105	434	7.798256	7.9E-12	38	235	0.618574	9.32E-12
	x_5	18	210	0.030947	2.6E-12	21	239	0.031051	3.14E-12	102	425	7.510973	8.16E-12	38	235	0.639584	9.32E-12
	x_6	10	130	0.019486	7.11E-12	10	130	0.01743	7.11E-12	63	311	2.828204	6.73E-12	37	341	0.966156	7.56E-12
	x_7	18	210	0.031324	2.6E-12	21	239	0.031842	3.14E-12	102	425	7.60765	8.16E-12	38	235	0.641613	9.32E-12
	x_8	28	291	0.041367	1.59E-12	22	244	0.031405	1.11E-12	100	401	7.097359	9.05E-12	32	217	0.536265	5.99E-12

TABLE 3. Numerical performance of the Algorithm 2.1 with the choices of β_k^1, β_k^2, NHZ method [7] and SASCG method [19].

Dimension	Initial point	Algorithm (β_k^1)				Algorithm (β_k^2)				NHZ				SASCG			
		ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM
Problem 4.5																	
50 000	x_1	107	1870	0.129382	8.76E-12	95	1594	0.104849	7.54E-12	215	1151	13.66385	9.8E-12	118	698	2.139761	5.3E-12
	x_2	110	1828	0.14291	9.81E-12	95	1552	0.105376	7.85E-12	117	629	7.59128	9.36E-12	51	355	0.867675	9.79E-12
	x_3	102	1700	0.11176	8.53E-12	93	1530	0.108174	6.52E-12	204	1060	12.4266	9.48E-12	67	412	1.104197	6.7E-12
	x_4	130	2356	0.176299	7.93E-12	107	1919	0.124247	7.33E-12	221	1180	13.8908	9.42E-12	71	430	1.157054	6.97E-12
	x_5	126	2184	0.157091	9.26E-12	112	1984	0.126567	2.86E-12	61	307	3.717902	9.96E-12	38	335	0.737079	6.47E-12
	x_6	118	2302	0.15289	8.36E-12	111	2119	0.139796	8.83E-12	221	1189	13.96324	8.8E-12	67	404	1.07935	6.72E-12
	x_7	126	2184	0.156526	8.02E-12	112	1983	0.126876	9.12E-12	61	307	3.690087	9.96E-12	38	335	0.752048	6.47E-12
	x_8	149	2872	0.187058	8.73E-12	119	2326	0.147132	9.17E-12	134	732	10.02112	9.47E-12	74	488	1.334455	8.8E-12
100 000	x_1	133	2563	0.398186	8.79E-12	106	2127	0.328188	9.84E-12	217	1145	32.27987	8.7E-12	64	391	1.987822	6.74E-12
	x_2	130	2540	0.389139	8.1E-12	109	2089	0.309386	9.96E-12	112	587	17.20897	9.43E-12	66	446	2.501087	9.51E-12
	x_3	101	1712	0.257701	9.6E-12	78	1320	0.192954	5.38E-12	203	1074	27.39177	8.84E-12	68	415	2.151276	8.82E-12
	x_4	129	2308	0.361717	9.04E-12	103	1927	0.291205	9.82E-12	221	1173	29.19649	9.04E-12	84	509	2.690442	7.86E-12
	x_5	121	2135	0.342276	9.85E-12	113	2098	0.31565	9.3E-12	59	298	7.555017	9.65E-12	36	325	1.407301	8.48E-12
	x_6	133	2629	0.407104	9.82E-12	116	2183	0.318154	1.86E-12	222	1201	30.16242	8.13E-12	69	417	2.142668	9.29E-12
	x_7	121	2135	0.337977	9.32E-12	124	2296	0.359323	8.87E-12	59	298	7.287248	9.65E-12	36	325	1.393137	8.48E-12
	x_8	128	2803	0.410258	8.78E-12	186	5063	0.729582	9.63E-12	128	655	16.29995	8.71E-12	76	496	2.51642	3.24E-12
Problem 4.6																	
50 000	x_1	11	81	0.011149	5.29E-13	11	81	0.010331	5.29E-13	67	264	2.48742	7.35E-12	37	266	0.664891	6.37E-12
	x_2	87	683	0.070714	9.02E-12	64	521	0.051216	8.43E-12	99	397	5.260201	7.86E-12	30	211	0.460318	2.09E-13
	x_3	10	78	0.011046	5.29E-13	10	78	0.010367	5.29E-13	67	266	2.433352	5.43E-12	34	257	0.653009	4.98E-12
	x_4	84	663	0.063468	9.44E-12	107	920	0.091064	3.79E-12	94	372	4.961414	8.35E-12	39	254	0.656538	8.05E-12
	x_5	91	705	0.079432	3.46E-12	133	1135	0.108182	1.86E-12	94	372	5.007879	8.89E-12	35	244	0.600121	7.24E-12
	x_6	11	81	0.011124	5.29E-13	11	81	0.010752	5.29E-13	67	264	2.407476	7.35E-12	37	266	0.678739	6.37E-12
	x_7	154	1202	0.12253	5.6E-12	134	1129	0.104197	8.84E-12	94	372	4.888267	8.89E-12	35	244	0.591089	7.24E-12
	x_8	70	540	0.052777	3.84E-12	71	577	0.056437	6.81E-12	101	405	5.291847	9.34E-12	30	211	0.487931	4.24E-12
100 000	x_1	11	72	0.019489	2.61E-12	11	72	0.018566	2.61E-12	70	276	5.228083	7.28E-12	39	273	1.583834	4.85E-12
	x_2	53	374	0.082765	4E-12	40	308	0.063694	6.87E-13	99	397	10.42438	7.86E-12	31	214	1.050178	1.96E-12
	x_3	10	69	0.01767	2.61E-12	10	69	0.017339	2.61E-12	70	278	5.039357	5.37E-12	36	264	1.455447	6.16E-12
	x_4	140	1126	0.244398	2.85E-12	99	851	0.181952	9.6E-12	95	376	9.916026	9.09E-12	38	248	1.366727	9.2E-12
	x_5	122	1027	0.221617	2.45E-12	102	865	0.178512	4.62E-12	95	376	9.688934	9.69E-12	29	214	1.030971	9.34E-12
	x_6	11	72	0.018369	2.61E-12	11	72	0.01912	2.61E-12	70	276	4.927796	7.28E-12	39	273	1.577534	4.85E-12
	x_7	173	1374	0.312497	9.9E-12	97	815	0.172876	6.28E-12	95	376	9.908528	9.69E-12	29	214	1.044024	9.34E-12
	x_8	34	241	0.057003	9.81E-13	32	235	0.052739	2.3E-12	101	405	11.15511	9.34E-12	32	217	1.090039	1.7E-12

TABLE 4. Numerical performance of the Algorithm 2.1 with the choices of β_k^1, β_k^2, NHZ method [7] and SASCG method [19].

Dimension	Initial point	Algorithm (β_k^1)				Algorithm (β_k^2)				NHZ				SASCG			
		ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM	ITER	FVL	TIME	NORM
Problem 4.7																	
50 000	x_1	12	37	0.007719	8.6E-12	12	37	0.007328	8.6E-12	102	404	4.537979	8.64E-12	40	241	0.560283	5.76E-12
	x_2	13	40	0.008267	1.27E-12	13	40	0.007529	2.06E-12	104	412	4.888402	8.16E-12	40	241	0.602824	9.06E-12
	x_3	13	40	0.0072	1.25E-12	13	40	0.007934	1.25E-12	104	412	3.619091	7.84E-12	40	241	0.58042	8.74E-12
	x_4	13	40	0.008341	1.06E-12	13	40	0.008102	1.58E-12	103	408	5.03848	8.75E-12	40	241	0.586481	7.47E-12
	x_5	13	40	0.007375	1.06E-12	13	40	0.007827	1.54E-12	103	408	5.090403	8.75E-12	40	241	0.578164	7.47E-12
	x_6	13	40	0.008602	1.8E-12	13	40	0.008491	1.83E-12	105	416	3.69663	8.64E-12	41	244	0.640144	5.56E-12
	x_7	13	40	0.008207	1.07E-12	13	40	0.007772	1.59E-12	103	408	5.233119	8.75E-12	40	241	0.583285	7.47E-12
	x_8	13	40	0.008167	1.25E-12	13	40	0.00778	2.03E-12	104	412	5.028123	8.14E-12	40	241	0.598252	9.04E-12
100 000	x_1	13	40	0.016034	5.39E-12	13	40	0.015226	5.42E-12	103	408	7.252856	9.41E-12	40	241	1.141298	8.15E-12
	x_2	13	40	0.017008	8.52E-12	17	52	0.018898	9.92E-12	105	416	9.1617	8.9E-12	41	244	1.162179	5.76E-12
	x_3	13	40	0.015706	8.23E-12	13	40	0.015067	8.41E-12	105	416	6.447497	8.57E-12	41	244	1.174316	5.62E-12
	x_4	13	40	0.016968	7.03E-12	13	40	0.015481	9.82E-12	104	412	9.679385	9.53E-12	41	244	1.179919	4.76E-12
	x_5	13	40	0.016721	7.03E-12	14	43	0.016411	8.27E-12	104	412	9.84188	9.53E-12	41	244	1.212743	4.76E-12
	x_6	14	43	0.015815	1.17E-12	14	43	0.016184	2.5E-12	106	420	6.564154	9.41E-12	41	244	1.189526	7.86E-12
	x_7	13	40	0.016145	7.03E-12	13	40	0.015599	9.99E-12	104	412	9.68217	9.53E-12	41	244	1.18275	4.76E-12
	x_8	13	40	0.015688	8.5E-12	17	52	0.019894	9.9E-12	105	416	8.795775	8.87E-12	41	244	1.195201	5.79E-12
Problem 4.8																	
10 000	x_1	62	1024	1.667563	8.91E-12	72	1191	1.595509	6.88E-12	185	927	9.311886	9.67E-12	71	432	1.594178	8.85E-12
	x_2	58	955	1.251509	7.29E-12	68	1127	1.431282	7.53E-12	188	940	10.62637	9.24E-12	68	412	1.783951	7.06E-12
	x_3	57	937	1.196726	8.13E-12	71	1173	1.517969	7.74E-12	182	924	10.53226	8.6E-12	72	439	2.01281	7.03E-12
	x_4	59	966	1.223491	6.29E-12	70	1158	1.530446	7.62E-12	184	916	10.38998	9.95E-12	137	805	4.362063	6.77E-12
	x_5	60	987	1.251391	2.33E-12	70	1158	1.544326	7.48E-12	184	927	10.5983	8.91E-12	137	805	4.089713	6.72E-12
	x_6	58	953	1.259473	6.28E-12	69	1143	1.505995	7.98E-12	178	882	10.7822	9.5E-12	181	1063	5.391542	7.02E-12
	x_7	60	987	1.290731	2.33E-12	70	1158	1.467102	7.48E-12	181	937	10.5585	9.47E-12	137	805	3.894939	6.72E-12
	x_8	62	1018	1.360829	9.04E-12	69	1142	1.469638	9.93E-12	189	961	10.86485	9.68E-12	67	406	1.837095	8.48E-12
15 000	x_1	67	1102	2.086426	5.2E-12	74	1223	2.228997	7.98E-12	185	935	17.29935	9.05E-12	202	1167	9.492279	6.99E-12
	x_2	71	1162	2.102345	8.66E-12	71	1174	2.003835	5.35E-12	186	928	18.3168	9.47E-12	69	418	2.991737	8.7E-12
	x_3	53	875	1.463352	8.45E-12	70	1157	1.966213	7.44E-12	182	903	17.19979	9.69E-12	70	425	3.187843	8.51E-12
	x_4	67	1102	1.836233	3.39E-12	70	1158	1.854484	7.43E-12	184	923	17.27476	9.82E-12	68	419	3.07921	8.13E-12
	x_5	64	1054	1.721099	6.12E-12	70	1158	1.758765	7.42E-12	184	927	17.87901	8.71E-12	68	419	3.064562	8.1E-12
	x_6	62	1018	1.621774	6.02E-12	71	1175	1.804555	8.42E-12	176	888	16.45619	9.86E-12	72	444	3.783374	7.12E-12
	x_7	64	1054	1.693863	6.12E-12	70	1158	1.695351	7.42E-12	182	912	16.85967	9.84E-12	68	419	3.269703	8.1E-12
	x_8	69	1134	1.836765	5.63E-12	71	1174	1.793954	9E-12	189	956	19.23637	9.4E-12	69	417	2.939676	7.22E-12

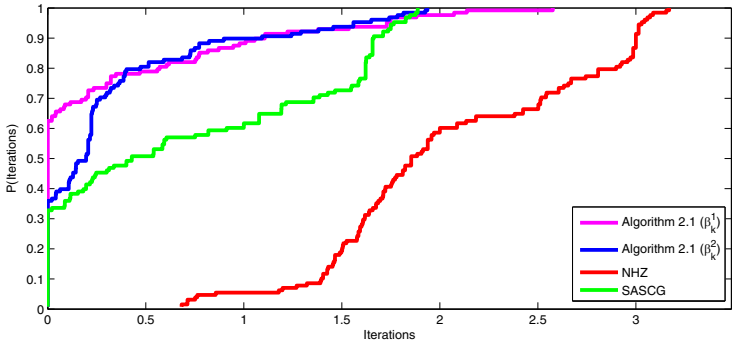

FIGURE 1. Performance of Algorithm 2.1 *versus* NHZ method [7] and SASCG method [19] (with respect to number of iteration).

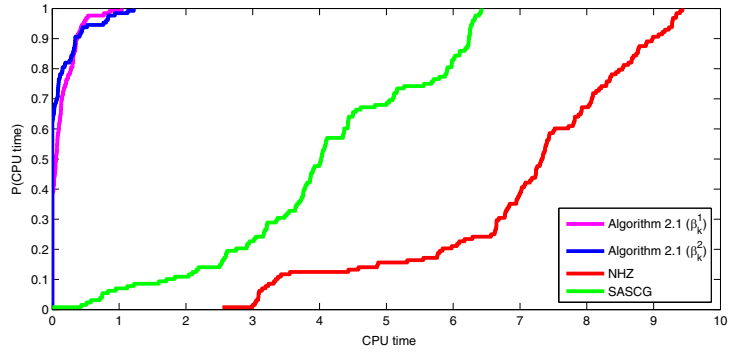

FIGURE 2. Performance of Algorithm 2.1 *versus* NHZ method [7] and SASCG method [19] (with respect to CPU time).

The Algorithm 2.1 with the choice β_k^1 has a relatively minimal number of iterations for the Table 1 compared to the Algorithm 2.1 with the choice of β_k^2, the NHZ method [7] and the SASCG method [19]. However, the Algorithm 2.1 with choice β_k^2 has a minimum number of the CPU time compared to the remaining methods. In addition, for Table 2, the Algorithm 2.1 with two choices of the β_k has a minimal number of iterations compared to the other two methods. For the CPU time, the second-choice algorithm wins over the problems with almost 99%.

In contrast, from Tables 3 and 4, our proposed algorithms also have fewer iterations and CPU time than the remaining methods, especially for Problems 4.7 and 4.8. Nevertheless, the SASCG method [19] has on average shown some impact on the minimum number of iterations for Problems 4.5 and 4.6. However, the overall performance in terms of less number of iterations and CPU time is based on our proposed methods with more than 70% for all the problems considered. However, with regard to the number of function evaluations, the two β_k choices in our algorithm often compete with the NHZ method [7] and the SASCG method [19]. We also plotted the three figures to illustrate the performance of our methods using the performance profiles of Dolan and Moré [11]. It is remarkable to note that, based on the Dolan and Moré procedure, the top curve in the figure has advantages over the remaining curves (Figs. 1–3).

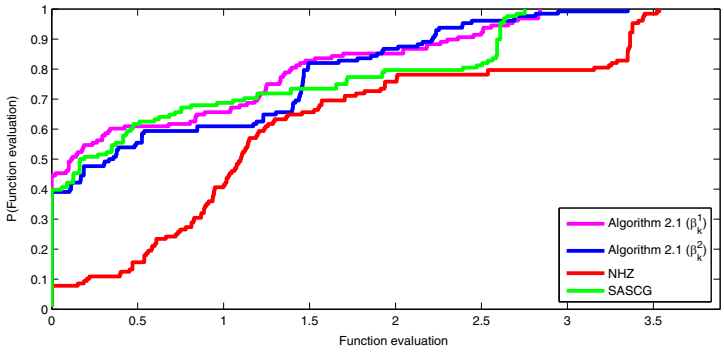

FIGURE 3. Performance of Algorithm 2.1 *versus* NHZ method [7] and SASCG method [19] (with respect to number of function evaluations).

5. Conclusion

We presented promising three-term CG-type methods to solve large-scale monotone nonlinear equations. Based on the idea of modifying the Dai–Liao conjugacy condition using the optimal choices of the non-negative parameter t, we suggested two new CG updates. However, to demonstrate the effectiveness of the suggested CG updates, we incorporated them into the Solodov and Svaiter projection techniques and solved monotone nonlinear equations. We proved the global convergence result of the proposed method and lastly used some test problems for the numerical efficacy of our methods compared to the NHZ Algorithm [7] and the SASCG method [19].

Acknowledgements. The first author is grateful to TWAS-CUI for the FR number award: 3240299486. We would like to thank the editor and all the reviewers for their valuable comments and suggestions.

References

[1] N. Andrei, Open problems in nonlinear conjugate gradient algorithms for unconstrained optimization. *Bull. Malays. Math. Sci. Soc.* **34** (2011) 319–330.
[2] S. Babaie-Kafaki and R. Ghanbari, The Dai–Liao nonlinear conjugate gradient method with optimal parameter choices. *Eur. J. Oper. Res.* **234** (2014) 625–630.
[3] J. Barzilai and J.M. Borwein, Two point stepsize gradient methods. *IMA J. Numer. Anal.* **8** (1988) 141–148.
[4] W. Cheng, A PRP type method for systems of monotone equations. *Math. Comput. Model.* **50** (2009) 15–20.
[5] Y.H. Dai and L.Z. Liao, New conjugacy conditions and related nonlinear conjugate gradient methods. *Appl. Math. Optim.* **43** (2001) 87–101.
[6] Y.H. Dai and Y. Yuan, A nonlinear conjugate gradient method with a strong global convergence property. *SIAM J. Optim.* **10** (1999) 177–182.
[7] Z. Dai and H. Zhu, A modified Hestenes–Stiefel-type derivative-free method for large-scale nonlinear monotone equations. *Mathematics* **8** (2020) 168.
[8] Z.F. Dai, X. Chen and F. Wen, A modifed Perry's conjugate gradient method-based derivative-free method for solving large-scale nonlinear monotone equations. *Appl. Math. Comput.* **270** (2015) 378–386.
[9] Z. Dai, H. Zhou, F. Wen and S. He, Efficient predictability of stock return volatility: the role of stock market implied volatility. *North Am. J. Econ. Finance* **52** (2020) 101174.
[10] J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Englewood Cliffs, NJ (1983).
[11] E.D. Dolan and J.J. Moré, Benchmarking optimization software with performance profiles. *Math. Program.* **91** (2002) 201–213.
[12] M. Figueiredo, R. Nowak and S.J. Wright, Gradient projection for sparse reconstruction, application to compressed sensing and other inverse problems. *IEEE J. Sel. Top. Sign. Proces.* **1** (2007) 586–597.
[13] P. Gao and C. He, An efficient three-term conjugate gradient method for nonlinear monotone equations with convex constraints. *Calcolo* **55** (2018) 53.
[14] W.W. Hager and H. Zhang, A survey of nonlinear conjugate gradient methods. *Pac. J. Optim.* **2** (2006) 35–58.

[15] M.R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.* **49** (1952) 409–436.

[16] Y. Hu and Z. Wei, A modified Liu–Storey conjugate gradient projection algorithm for nonlinear monotone equations. *Int. Math. Forum* **9** (2014) 1767–1777.

[17] A.N. Iusem and M.V. Solodov, Newton-type methods with generalized distances for constrained optimization. *Optimization* **41** (1997) 257–278.

[18] H. Kobayashi, Y. Narushima and H. Yabe, Descent three-term conjugate gradient methods based on secant conditions for unconstrained optimization. *Optim. Methods Softw.* **32** (2017) 1313–1329.

[19] M. Koorapetse and P. Kaelo, Self adaptive spectral conjugate gradient method for solving nonlinear monotone equations. *J. Egypt. Math. Soc.* **28** (2020) 4.

[20] Q. Li and D.H. Li, A class of derivative-free methods for large-scale nonlinear monotone equations. *IMA J. Numer. Anal.* **31** (2011) 1625–1635.

[21] J. Liu and S. Li, Spectral DY-type projection method for nonlinear monotone systems of equations. *J. Comput. Math.* **33** (2015) 341–354.

[22] K. Meintjes and A.P. Morgan, A methodology for solving chemical equilibrium systems. *Appl. Math. Comput.* **22** (1987) 333–361.

[23] Y. Narushima and H. Yabe, A survey of sufficient descent conjugate gradient methods for unconstrained optimization. *SUT J. Math.* **50** (2014) 167–203.

[24] Y. Narushima, H. Yabe and J.A. Ford, A three-term conjugate gradient method with sufficient descent property for unconstrained optimization. *SIAM J. Optim.* **21** (2011) 212–230.

[25] J.M. Ortega and W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables. SIAM, Philadelphia, PA (1970).

[26] J. Sabi'u, A. Shah and M.Y. Waziri, Two optimal Hager–Zhang conjugate gradient methods for solving monotone nonlinear equations. *Appl. Numer. Math.* **153** (2020) 217–233.

[27] M.V. Solodov and B.F. Svaiter, A globally convergent inexact Newton method for systems of monotone equations. Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods, edited by M. Fukushima and L. Qi. In Vol. 22 of *Applied Optimization*. Springer (1998) 355–369.

[28] S. Wang and H. Guan, A scaled conjugate gradient method for solving monotone nonlinear equations with convex constraints. *J. Appl. Math.* **2013** (2013) 286486.

[29] M.Y. Waziri, K.A. Hungu and J. Sabi'u, Descent Perry conjugate gradient methods for systems of monotone nonlinear equations. *Numer. Algorithms* **85** (2020) 763–785.

[30] L. Zhang and W. Zhou, Spectral gradient projection method for solving nonlinear monotone equations. *J. Comput. Appl. Math.* **196** (2006) 478–484.

[31] Y.B. Zhao and D.H. Li, Monotonicity of fixed point and normal mapping associated with variational inequality and its application. *SIAM J. Optim.* **4** (2001) 962–973.

[32] W.J. Zhou and D.H. Li, Limited memory BFGS method for nonlinear monotone equations. *J. Comput. Math.* **25** (2007) 89–96.

[33] W.J. Zhou and D.H. Li, A globally convergent BFGS method for nonlinear monotone equations without any merit functions. *Math. Comput.* **77** (2008) 2231–2240.

[34] G. Zhou and K.C. Toh, Superlinear convergence of a Newton-type algorithm for monotone equations. *J. Optim. Theory App.* **125** (2005) 205–221.