Journal of King Saud University – Science 34 (2022) 101923

Contents lists available at ScienceDirect

Journal of King Saud University – Science

journal homepage: www.sciencedirect.com

Original article

A Liu-Storey-type conjugate gradient method for unconstrained minimization problem with application in motion control

Auwal Bala Abubakar [a,b,c], Maulana Malik [d], Poom Kumam [a,e,f,*], Hassan Mohammad [b], Min Sun [g], Abdulkarim Hassan Ibrahim [a], Aliyu Ibrahim Kiri [b]

[a] Fixed Point Research Laboratory, Fixed Point Theory and Applications Research Group, Center of Excellence in Theoretical and Computational Science (TaCS-CoE), Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha Uthit Rd., Bang Mod, Thung Khru, Bangkok 10140, Thailand
[b] Numerical Optimization Research Group, Department of Mathematical Sciences, Faculty of Physical Sciences, Bayero University, Kano, Kano, Nigeria
[c] Department of Mathematics and Applied Mathematics, Sefako Makgatho Health Sciences University, Ga-Rankuwa, Pretoria, Medunsa-0204, South Africa
[d] Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Indonesia (UI), Depok 16424, Indonesia
[e] Center of Excellence in Theoretical and Computational Science (TaCS-CoE), Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha Uthit Rd., Bang Mod, Thung Khru, Bangkok 10140, Thailand
[f] Departments of Medical Research, China Medical University Hospital, China Medical University, Taichung 40402, Taiwan
[g] School of Mathematics and Statistics, Zaozhuang University, Zaozhuang, Shandong 277160, China

ARTICLE INFO

Article history:
Received 25 December 2020
Revised 17 February 2022
Accepted 21 February 2022
Available online 25 February 2022

Mathematics Subject Classification:
65K05
90C52
90C26

Keywords:
Unconstrained optimization
Conjugate gradient method
Line search
Global convergence

ABSTRACT

Conjugate gradient methods have played a vital role in finding the minimizers of large-scale unconstrained optimization problems due to the simplicity of their iteration, convergence properties and their low memory requirements. Based on the Liu-Storey conjugate gradient method, in this paper, we present a Liu-Storey type method for finding the minimizers of large-scale unconstrained optimization problems. The direction of the proposed method is constructed in such a way that the sufficient descent condition is satisfied. Furthermore, we establish the global convergence result of the method under the standard Wolfe and Armijo-like line searches. Numerical findings indicate that our presented approach is efficient and robust in solving large-scale test problems. In addition, an application of the method is explored.

1. Introduction

In this paper, we present alternative nonlinear conjugate gradient (CG) method for solving unconstrained optimization problems.

Conjugate gradient methods are preferable among other optimization methods because of low storage requirements (Hager and Zhang, 2006). Based on these, a considerable amount of research articles have been published, most of these articles considered global convergence of the conjugate gradient methods that are previously difficult to globalize (see, for example, Zhang et al., 2006; Gilbert and Nocedal, 1992; Al-Baali, 1985; Dai et al., 2000). In other perspective, some articles considered widening the applicability as well as improving the numerical performance of the existing conjugate gradient methods (see for example Yuan et al., 2019; Yuan et al., 2020; Hager and Zhang, 2013; Ding et al., 2010; Xie et al., 2009; Xie et al., 2010; Abubakar et al., 2021; Abubakar et al., 2021; Malik et al., 2021).

Consider the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ is continuously differentiable and its gradient $g(x)=\nabla f(x)$ is Lipschitz continuous. The conjugate gradient method for solving (1) generates a sequence $\{x_k\}$ via the iterative formula

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

* Corresponding author at: Fixed Point Research Laboratory, Fixed Point Theory and Applications Research Group, Center of Excellence in Theoretical and Computational Science (TaCS-CoE), Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha Uthit Rd., Bang Mod, Thung Khru, Bangkok 10140, Thailand.

E-mail address: poom.kum@kmutt.ac.th (P. Kumam).

https://doi.org/10.1016/j.jksus.2022.101923
1018-3647/

with d_k given by

$$d_k := \begin{cases} -g(x_k), & \text{if } k = 0, \\ -g(x_k) + \beta_k d_{k-1}, & \text{if } k > 0, \end{cases} \tag{2}$$

and the search direction d_k satisfies the descent condition

$$g_k^T d_k \leqslant -c\|g_k\|^2, \tag{3}$$

for some $c > 0$.

The parameter $\alpha_k > 0$ is the steplength obtained by either the standard Wolfe line search conditions (Sun and Yuan, 2006), namely,

$$f(x_k + \alpha_k d_k) - f(x_k) \leqslant \vartheta \alpha_k g_k^T d_k, \tag{4}$$

$$g(x_k + \alpha_k d_k)^T d_k \geqslant \sigma g_k^T d_k, \tag{5}$$

or the Armijo-like line search condition proposed by Grippo and Lucidi (1997), where α_k is obtained as: $\rho, \vartheta \in (0,1)$, $\alpha_k = \rho^i$, where i is the least nonnegative integer such that the following inequality

$$f(x_k + \alpha_k d_k) \leqslant f(x_k) - \vartheta \alpha_k^2 \|d_k\|^2 \tag{6}$$

hold.

The scalar β_k is known as the conjugate gradient parameter. Some notable parameters are; the Hestenes-Stiefel (HS) (Hestenes and Stiefel, 1952), Polak-Ribiére-Polyak (PRP) (Polak and Ribiere, 1969; Polyak, 1969) and Liu-Storey (LS) (Liu and Storey, 1991). They are defined as follows:

$$\beta_k^{HS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \quad \beta_k^{PRP} = \frac{g_k^T y_{k-1}}{\|g_{k-1}\|^2}, \quad \beta_k^{LS} = \frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}}.$$

where $g_k = g(x_k)$, $y_{k-1} = g_k - g_{k-1}$.

It can be observed that the HS, PRP, and LS method have similarity in their numerator, this similarity makes them have some common features in terms of computational performance. There exists many works on the development of the HS and PRP methods (see, for example, Zhang et al., 2007; Dong et al., 2017; Yu et al., 2008; Saman and Ghanbari, 2014; Andrei, 2011; Yuan et al., 2021), however, as noticed by Hager and Zhang (2006) in their review article, the research on the LS method is rare. Nevertheless, some works on the LS parameter have been presented in the references (Shi and Shen, 2007; Tang et al., 2007; Zhang, 2009; Liu and Feng, 2011; Li and Feng, 2011. In Shi and Shen, 2007; Tang et al., 2007; Liu and Feng, 2011), the authors proposed a new form of Armijo-like line search strategy and used it to investigate the global convergence of the LS method. Zhang (2009), developed a new descent LS-type method based on the memoryless BFGS update and proved the global convergence of the method for general function using the Grippo and Lucidi (1997) line search. Li and Feng (2011), remarked that because of the similarity of the HS, PRP, and LS conjugate parameters, the techniques applied to the HS and PRP can be applied to the LS method to improve its convergence and computational performance. In this regard, Li and Feng, proposed a modification of the LS method similar to the well-known Hager and Zhang CG-DESCENT (Hager and Zhang, 2005).

Noticing the above development, in this paper, we seek another alternative LS-type method following the modifications of the HS and PRP methods by Liang et al. (2015) and Aminifard and Saman (2019), respectively. Motivated by the above proposals and the similarity of the LS CG parameter with HS and PRP parameters, we present an efficient descent LS-type method for nonconvex functions. The direction satisfy the sufficient descent property without any line search requirement and the method is globally convergent under the standard Wolfe line search as well as the Armijo-like line search. The rest of the paper is organized as follows. In Section 2 we provide the algorithm of the descent LS - type method together with its convergence result. In Section 3, we present the numerical experiments, and conclusions are given in Section 4. Unless otherwise stated, throughout this paper we denote $g_k = g(x_k)$.

2. Algorithm and theoretical results

Recently, some proposals have been made to modify the HS and PRP methods. The proposal to modify the HS method was done by Liang et al. (2015) and that to modify the PRP (named as NMPRP method) was done by Aminifard and Saman (2019). In Liang et al. (2015), some CG methods with a descent direction was presented. Moreover, the conjugacy condition given by Dai and Liao (2001) was extended. Likewise in Aminifard and Saman (2019), a descent extension of the PRP method was presented and the global convergence was shown under the Wolfe and Armijo-like line search, respectively. In this section, we build on the analysis given in Liang et al. (2015) and Aminifard and Saman (2019) on the LS method to guarantee a sufficient descent direction (3).

Note that if $\beta_k^{LS} \geqslant 0$, then condition (3) is satisfied when $g_k^T d_{k-1} \leqslant 0$. Now, our interest is to consider the case where $g_k^T d_{k-1} > 0$ and the direction satisfies (3). To achieve this, we attach a scaling parameter γ_k to the term $-g_k$ in (2) and propose the following search direction

$$d_0 = -g_0, \; d'_k := \begin{cases} -\gamma_k g_k + \beta_k^{MLS} d_{k-1}, & \text{if } g_k^T d_{k-1} > 0, \; k > 0, \\ -g_k + \beta_k^{LS} d_{k-1}, & \text{if } g_k^T d_{k-1} \leqslant 0, \; k > 0, \end{cases} \tag{7}$$

where

$$\gamma_k = 1 + \frac{g_k^T d_{k-1}}{\|g_k\|^2} \beta_k^{LS}$$

Table 1
Test Problems.

No	Functions	No	Functions
F1	Extended White and Holst	F29	Extended Quadratic Penalty QP2
F2	Extended Rosenbrock	F30	Extended Quadratic Penalty QP1
F3	Extended Freudenstein and Roth	F31	Quartic
F4	Extended Beale	F32	Matyas
F5	Raydan 1	F33	Colville
F6	Extended Tridiagonal 1	F34	Dixon and Price
F7	Diagonal 4	F35	Sphere
F8	Extended Himmelblau	F36	Sum Squares
F9	FLETCHCR	F37	DENSCHNA
F10	Extended Powel	F38	DENSCHNF
F11	NONSCOMP	F39	Extended Block-Diagonal BD1
F12	Extended DENSCHNB	F40	HIMMELBH
F13	Extended Penalty Function U52	F41	Extended Hiebert
F14	Hager	F42	DQDRTIC
F15	Extended Maratos	F43	ENGVAL1
F16	Six Hump Camel	F44	ENGVAL8
F17	Three Hump Camel	F45	Linear Perturbed
F18	Booth	F46	QUARTICM
F19	Trecanni	F47	DIAG-AUP1
F20	Zettl	F48	ARWHEAD
F21	Shallow	F49	Brent
F22	Generalized Quartic	F50	Deckkers-Aarts
F23	Quadratic QF2	F51	El-Attar-Vidyasagar-Dutta
F24	Leon	F52	Price 4
F25	Generalized Tridiagonal 1	F53	Zirilli or Aluffi-Pentini's
F26	Generalized Tridiagonal 2	F54	Diagonal Double Border Arrow Up
F27	POWER	F55	HARKERP
F28	Quadratic QF1	F56	Extended Quadratic Penalty QP3

and

$$\beta_k^{MLS} = \left(1 - \frac{g_k^T s_{k-1}}{-g_{k-1}^T d_{k-1}}\right)\beta_k^{LS} - t\frac{\|y_{k-1}\|^2 g_k^T s_{k-1}}{(g_{k-1}^T d_{k-1})^4},\ t \geqslant 0.$$

It is worth noting that if the exact line search is used, the parameter β_k^{MLS} reduces to β_k^{LS}. If $g_k^T y_{k-1} \leqslant 0$ the direction d_k' defined in (7) may not necessarily be descent, in this case we set $d_k' = -g_k$. Therefore, we defined the proposed direction by

$$d_0 = -g_0,\ d_k := \begin{cases} -g_k, & \text{if } g_k^T y_{k-1} \leqslant 0,\ k > 0, \\ d_k', & \text{if } g_k^T y_{k-1} > 0,\ k > 0. \end{cases} \tag{8}$$

It can be observed that a restart process for the CG method with direction d_k' defined by (7) occur when $g_k^T y_{k-1} \leqslant 0$.

Below we give the algorithm flow framework of the proposed scheme.

Algorithm 1: New Modified LS method (NMLS)

Input : Given $x_0 \in \mathbb{R}^n$, $\epsilon > 0$ $\rho \in (0,1)$, $0 < \vartheta < \sigma < 1$, and $t \geq 0$.

Step 1 : **if** $\|g_k\| \leq \epsilon$, **then**
  stop.
end
Step 2 : **if** $k = 0$, **then**
  set $d_k := -g_k$;
else
  Compute d_k using equations (7)-(8).
end
Step 3 : Compute the step length α_k satisfying either the standard Wolfe (4)-(5) or Armijo (6) line search conditions.
Step 4 : Compute the next iterate $x_{k+1} = x_k + \alpha_k d_k$.
Step 5 : Set $k := k + 1$ and go to Step 1.

Lemma 2.1. *The search direction d_k defined by (8) satisfy (3) with $c = 1$.*

Proof. For $k = 0$ or $g_k^T y_{k-1} \leqslant 0$, it is easy to see that the sufficient descent condition (3) is satisfied. For $k > 0$, we consider the following two cases:

1[st]case : For $g_k^T d_{k-1} > 0$, and $g_k^T y_{k-1} > 0$, then from (7) and (8)

$$\begin{aligned} g_k^T d_k &= -\gamma_k \|g_k\|^2 + \beta_k^{MLS} g_k^T d_{k-1} \\ &= -\left(1 + \frac{g_k^T d_{k-1}}{\|g_k\|^2}\frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}}\right)\|g_k\|^2 \\ &\quad + \left(1 - \frac{g_k^T s_{k-1}}{-g_{k-1}^T d_{k-1}}\right)\frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}} g_k^T d_{k-1} - t\frac{\|y_{k-1}\|^2 g_k^T s_{k-1}}{(g_{k-1}^T d_{k-1})^4} g_k^T d_{k-1} \\ &\leqslant -\|g_k\|^2 - \frac{(g_k^T d_{k-1})(g_k^T y_{k-1})}{-g_{k-1}^T d_{k-1}} + \left(1 - \frac{g_k^T s_{k-1}}{-g_{k-1}^T d_{k-1}}\right)\frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}} g_k^T d_{k-1} \\ &= -\|g_k\|^2 - \frac{(g_k^T d_{k-1})(g_k^T y_{k-1})}{-g_{k-1}^T d_{k-1}} + \frac{(g_k^T d_{k-1})(g_k^T y_{k-1})}{-g_{k-1}^T d_{k-1}} - \frac{(g_k^T s_{k-1})(g_k^T d_{k-1})(g_k^T y_{k-1})}{(g_{k-1}^T d_{k-1})^2} \\ &= -\|g_k\|^2 - \frac{(g_k^T s_{k-1})(g_k^T d_{k-1})(g_k^T y_{k-1})}{(g_{k-1}^T d_{k-1})^2} \\ &\leqslant -\|g_k\|^2. \end{aligned}$$

2[nd]case : For $g_k^T d_{k-1} \leqslant 0$, we proceed the proof of this case by induction. For $k = 0$, the result follows trivially. Suppose it is true for $k = k - 1$, that is, $g_{k-1}^T d_{k-1} \leqslant -\|g_{k-1}\|^2$. This implies that $-g_{k-1}^T d_{k-1} > \|g_{k-1}\|^2 > 0$. Now, we prove it is true for $k = k$.Since $g_k^T y_{k-1} > 0$ and by induction hypothesis $-g_{k-1}^T d_{k-1} > 0$, then from (7) and (8)

$$g_k^T d_k = -\|g_k\|^2 + \beta_k^{LS} g_k^T d_{k-1} \leqslant -\|g_k\|^2.$$

□

Remark 2.2. Note that with an exact line search, the CG scheme with the search direction defined by (7) collapses to the LS scheme.

Next, we prove the convergence of Algorithm 1 under the standard Wolfe line search conditions (4)-(5). In addition, we employ the following assumptions.

Assumption 2.3. The level set $\mathscr{H} = \{x : f(x) \leqslant f(x_0)\}$ is bounded.

Assumption 2.4. In some neighborhood $\mathscr{L}$ of $\mathscr{H}$, the gradient of the function f is Lipschitz continuous. That is, there exists $L > 0$ such that

$$\|g(x) - g(\bar{x})\| \leqslant L\|x - \bar{x}\|, \bar{x} \in \mathscr{L}. \tag{9}$$

Assumptions 2.3 and 2.4 implies that for all $x \in \mathscr{H}$ there exists $c_1, c_2 > 0$ such that;

- $\|x\| \leqslant c_1$.
- $\|g(x)\| \leqslant c_2$.
- $\{x_k\} \in \mathscr{H}$ as $\{f(x_k)\}$ is decreasing.

Theorem 2.5. *Suppose the inequalities (4) and (5) hold. If*

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

Then

$$\liminf_{k\to\infty}\|g_k\| = 0. \tag{10}$$

Proof. Suppose the conclusion of the theorem is not true, then for some $\epsilon > 0$

$$\|g_k\| \geqslant \epsilon. \tag{11}$$

By Lemma 2.1 together with (4), we have

$$f(x_{k+1}) \leqslant f(x_k) + \vartheta\alpha_k g_k^T d_k \leqslant f(x_k) - \vartheta\alpha_k\|g_k\|^2 \leqslant f(x_k) \leqslant f(x_{k-1}) \leqslant \dots \leqslant f(x_0).$$

Likewise, by Lemma 2.1, inequalities (5) and (9),

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

Joining the above with (4) yield,

$$\frac{\vartheta(1-\sigma)}{L}\frac{(g_k^T d_k)^2}{\|d_k\|^2} \leqslant f(x_k) - f(x_{k+1})$$

and

$$\frac{\vartheta(1-\sigma)}{L}\sum_{k=0}^{\infty}\frac{(g_k^T d_k)^2}{\|d_k\|^2} \leqslant (f(x_0) - f(x_1)) + (f(x_1) - f(x_2)) + \dots \leqslant f(x_0) < +\infty.$$

This implies that

$$\sum_{k=0}^{\infty}\frac{(g_k^T d_k)^2}{\|d_k\|^2} < +\infty. \tag{12}$$

Now, inequality (11) with (3) implies that

$$\begin{aligned} g_k^T d_k &\leqslant -\|g_k\|^2 \\ &\leqslant -\epsilon^2. \end{aligned} \tag{13}$$

From (13), we have

$$\sum_{k=0}^{\infty}\frac{(g_k^T d_k)^2}{\|d_k\|^2} \geqslant \sum_{k=0}^{\infty}\frac{\epsilon^4}{\|d_k\|^2} = +\infty.$$

The above lead to a contradiction with (12). Thus, (10) holds. □

In what follows, we prove the convergence result of Algorithm 1 under the Armijo-like condition (6).

Since $\{f(x_k)\}$ is nonincreasing, using (6) we have

$$\sum_{k=0}^{\infty}\alpha_k^2\|d_k\|^2 < +\infty,$$

which further implies that

$$\lim_{k\to\infty}\alpha_k\|d_k\| = 0. \tag{14}$$

Theorem 2.6. *If α_k satisfies the Armijo-like condition (6), then*

$$\liminf_{k\to\infty}\|g(x_k)\| = 0. \tag{15}$$

Proof. Suppose Eq. (15) does not hold. Then there is $\epsilon > 0$ such that

$$\|g(x_k)\| \geqslant \epsilon \text{ for all } k \in \{0\} \cup \mathbb{N}. \tag{16}$$

We first show that the search direction d_k satisfies $\|d_k\| \leqslant N$ for some $N > 0$, using the following two cases.

1^{st}case : $g_k^T d_{k-1} \leqslant 0, g_k^T y_{k-1} > 0$ and $-g_{k-1}^T d_{k-1} > 0$, then from (7), (8) and (3), we have

$$\begin{aligned} \|d_k\| &= \|-g_k + \beta_k^{LS} d_{k-1}\| \\ &= \left\|-g_k + \frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}} d_{k-1}\right\| \\ &\leqslant \|g_k\| + \frac{\|g_k\|\|y_{k-1}\|\|d_{k-1}\|}{\|g_{k-1}\|^2} \\ &\leqslant c_2 + \frac{c_2 L\alpha_{k-1}\|d_{k-1}\|^2}{\epsilon^2}. \end{aligned}$$

As $\alpha_k\|d_k\| \to 0$ from (14), then we can find a $c_3 \in (0,1)$ and an integer k_0 for which $k \geqslant k_0$ and

$$\frac{c_2 L}{\epsilon^2}\alpha_{k-1}\|d_{k-1}\| \leqslant c_3.$$

So, for every $k > k_0$,

$$\begin{aligned} \|d_k\| &\leqslant c_2 + c_3\|d_{k-1}\| \leqslant c_2(1 + c_3 + c_3^2 + \dots + c_3^{k-k_0-1}) + c_3^{k-k_0}\|d_{k_0}\| \\ &\leqslant \frac{c_2}{1-c_3} + \|d_{k_0}\|. \end{aligned}$$

Hence, we have $\|d_k\| \leqslant N_1$ where $N_1 = \max\{\|d_0\|, \|d_1\|, \dots, \|d_{k_0}\|, \frac{c_2}{1-c_3} + \|d_{k_0}\|\}$.

2^{nd}case : $g_k^T d_{k-1} > 0, g_k^T y_{k-1} > 0$ and $-g_{k-1}^T d_{k-1} > 0$, then from (7) and (8)

$$\begin{aligned} \|d_k\| &= \|-\gamma_k g_k + \beta_k^{MLS} d_{k-1}\| \\ &\leqslant |\gamma_k|\|g_k\| + |\beta_k^{MLS}|\|d_{k-1}\| \\ &= \left|\left(1 + \frac{g_k^T d_{k-1}}{\|g_k\|^2}\frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}}\right)\right|\|g_k\| \\ &\quad + \left|\left(1 - \frac{g_k^T s_{k-1}}{-g_{k-1}^T d_{k-1}}\right)\frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}} - t\frac{\|y_{k-1}\|^2 g_k^T s_{k-1}}{(g_{k-1}^T d_{k-1})^4}\right|\|d_{k-1}\| \\ &\leqslant \left(1 + \frac{\|y_{k-1}\|\|d_{k-1}\|}{\|g_{k-1}\|^2}\right)\|g_k\| + \left(1 + \frac{\|g_k\|\|s_{k-1}\|}{\|g_{k-1}\|^2}\right)\frac{\|g_k\|\|y_{k-1}\|}{\|g_{k-1}\|^2}\|d_{k-1}\| \\ &\quad + t\frac{\|y_{k-1}\|^2\|g_k\|\|s_{k-1}\|}{\|g_{k-1}\|^8}\|d_{k-1}\| \\ &= \|g_k\| + \left(2 + \frac{\|g_k\|\|s_{k-1}\|}{\|g_{k-1}\|^2} + +t\frac{\|y_{k-1}\|\|s_{k-1}\|}{\|g_{k-1}\|^4}\right)\frac{\|g_k\|\|y_{k-1}\|\|d_{k-1}\|}{\|g_{k-1}\|^2}. \end{aligned}$$

Applying same procedure as in 1^{st} **case**, then we can find $N_2 > 0$ such that $\|d_k\| \leqslant N_2$. If we let $N = \max\{N_1, N_2\}$, the

$$\|d_k\| \leqslant N. \tag{17}$$

Next, from the Armijo-like line search condition (6), letting $\alpha_k' := \rho^{-1}\alpha_k = \rho^{i-1}$, we must have

$$f(x_k + \rho^{-1}\alpha_k d_k) > f(x_k) - \vartheta\rho^{-2}\alpha_k^2\|d_k\|^2. \tag{18}$$

By mean value theorem, Cauchy Schwartz inequality and (9), there is $l_k \in (0,1)$ for which

$$\begin{aligned} f(x_k + \rho^{-1}\alpha_k d_k) - f(x_k) &= \rho^{-1}\alpha_k g(x_k + l_k\rho^{-1}\alpha_k d_k)^T d_k \\ &= \rho^{-1}\alpha_k g_k^T d_k \\ &\quad + \rho^{-1}\alpha_k(g(x_k + l_k\rho^{-1}\alpha_k d_k) - g_k)^T d_k \\ &\leqslant \rho^{-1}\alpha_k g_k^T d_k + L\rho^{-2}\alpha_k^2\|d_k\|^2. \end{aligned} \tag{19}$$

Inequalities (17), (16), (18) and (19) implies

$$\alpha_k \geqslant \frac{\rho\|g_k\|^2}{(L+\vartheta)\|d_k\|^2} \geqslant \frac{\rho\epsilon^2}{(L+\vartheta)N^2} > 0.$$

This and (14) gives

$$\lim_{k\to\infty}\|d_k\| = 0. \tag{20}$$

From the sufficient descent condition (3), we get

$$\|d_k\| \geqslant \|g_k\|.$$

Therefore, we have $\lim_{k\to\infty}\|g_k\| = 0$. This contradicts (16), hence (15) holds. □

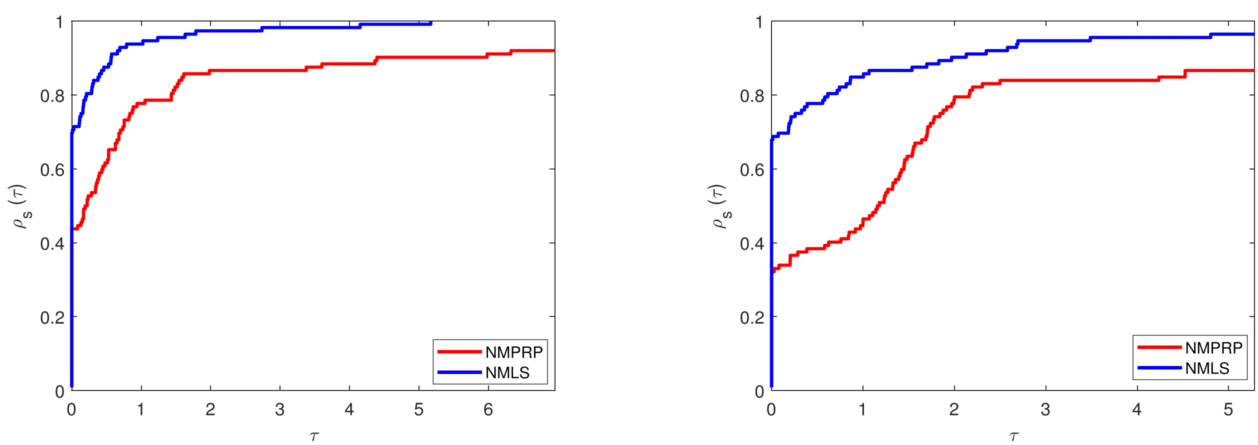
3. Numerical examples

This section demonstrates the numerical performance of Algorithm 1. Algorithm 1 is named as NMLS method. Based on the number of iteration (NOI), the number of function evaluations (NOF), and central processing unit (CPU) time, we compare the performance of the NMLS method with the NMPRP method (Aminifard and Saman, 2019). Fifty-six (56) test functions suggested by Andrei (2017) and Jamil (2013) are selected for testing the computational performance of each method. For each test function, we run two problems with different starting points or variation dimensions of 2, 4, 8, 10, 50, 100, 500, 1000, 5000, 10000, 50000 and 100000. The list of test functions is given in Table 1. The algorithm was coded in MATLAB R2019a and compiled on a PC with specification: processor Intel Core i7, operating system Windows 10 pro 64 bit, and 16 GB RAM. We stop the algorithm when $\|g_k\| \leqslant 10^{-6}$, and the numerical results are said to fail (use "–" to report the failure), if one of the conditions is satisfied: the NOI is more than 10,000 or the problem fail to attained the optimum value.

In this section, we compare the proposed NMLS method with NMPRP method using the strong Wolfe and Armijo-like line searches. This is done as a fair comparison adapted to the NMPRP method in Aminifard and Saman (2019). We use the strong Wolfe conditions comprises of (4) and the following stronger version of (5):

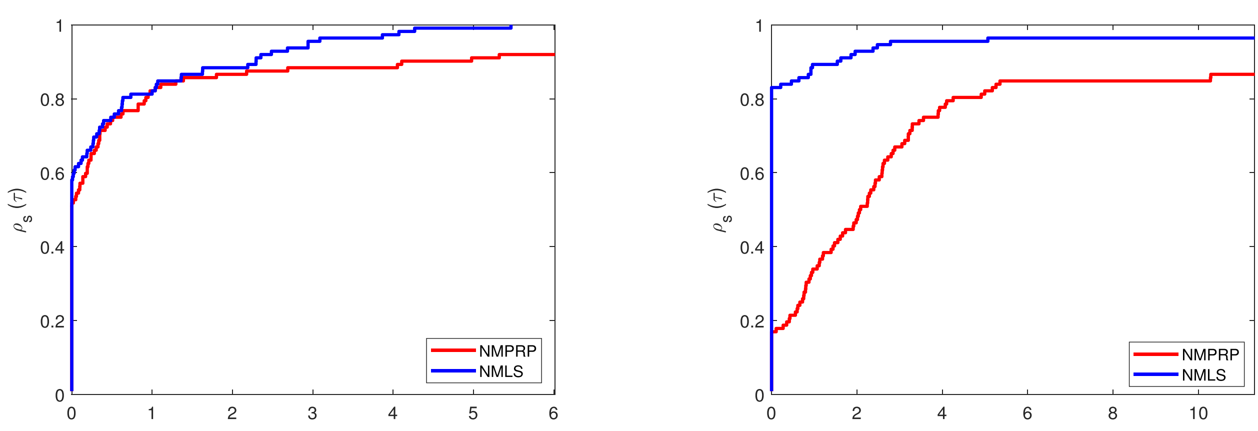
$$|g_{k+1}^T d_k| \leqslant -\sigma g_k^T d_k.$$

To obtain the step-size α_k by using strong Wolfe line search in the NMLS method, the parameters are selected as $\sigma = 0.05$ and $\vartheta = 0.0001$. Meanwhile, under the Armijo-like line search, we use the parameters $\rho = 0.25$ and $\delta = 0.00003$. For good performance, we choose the parameter $t = 0.1$ for evaluating β_k^{MLS}. In the NMPRP method, we still maintain all the parameter values according to the article (Aminifard and Saman, 2019). All numerical results under the strong Wolfe line search are shown in Table 2 and under Armijo-like line search are given in Table 3. The items in each table have the following meanings: Funct.: the test function, Dim.:



(a) Under strong Wolfe line search	(b) Under Armijo-like line search

Fig. 1. Performance profile based on NOI.



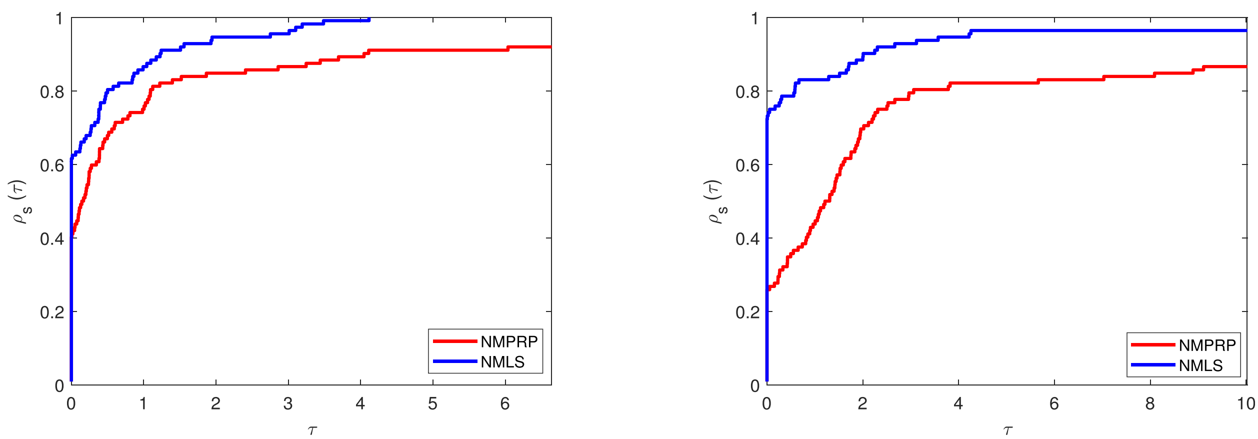
(a) Under strong Wolfe line search	(b) Under Armijo-like line search

Fig. 2. Performance profile based on NOF.

dimension, and IP: arbitrary initial points. From the numerical results in Tables 2 and 3, we can see that the NMLS method in this paper is promising.

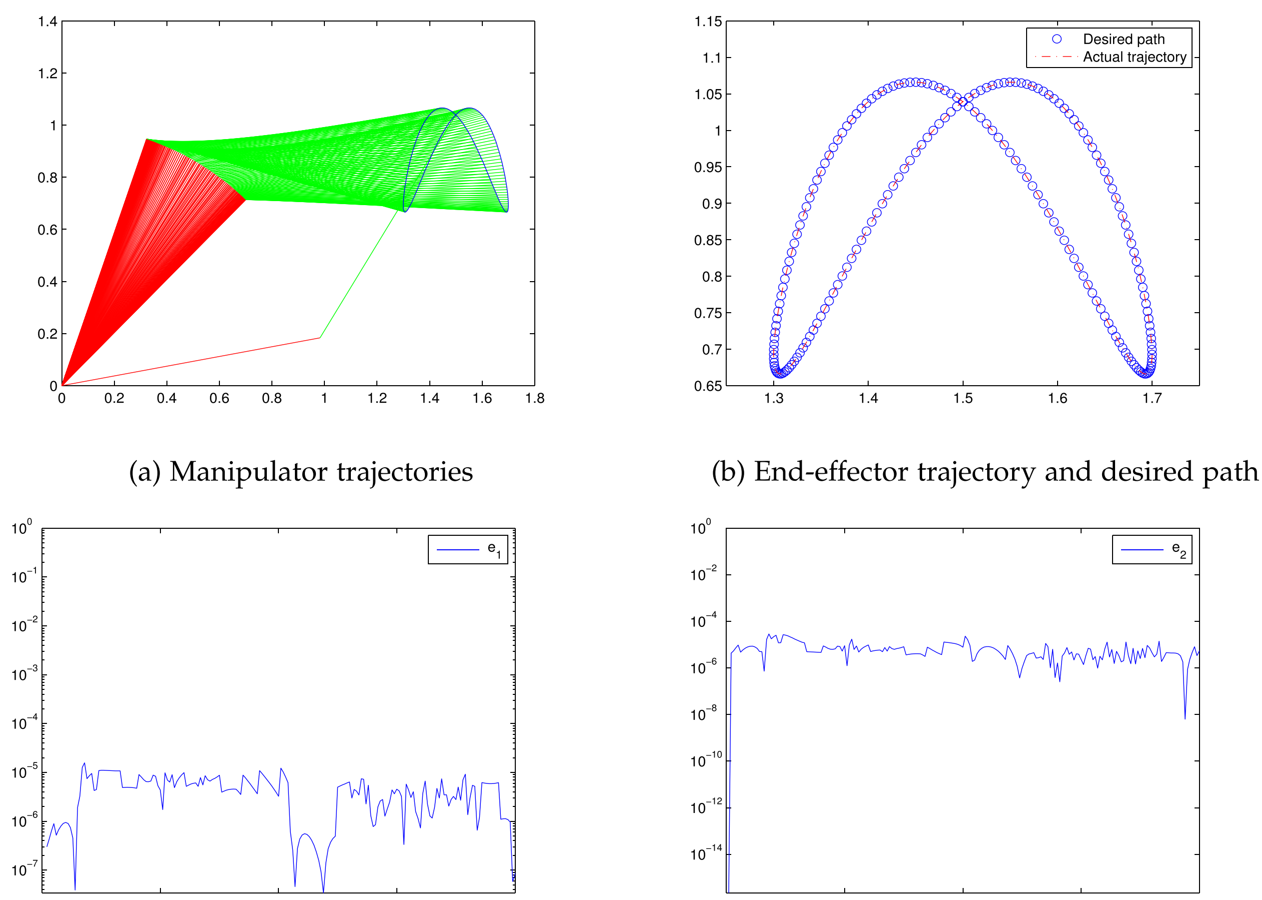
On the other hand, we use the performance profile introduced by Dolan and Moré (2002) to compare and evaluate the performance of NMLS and the NMPRP methods. The performance profile of each method is depicted by a segment of each problem to obtain a profile curve. From the profile curve, the method that top the curve has the best performance.

Figs. 1–3 present the performance profile with respect to the three metrics; NOI, NOF and CPU time of the two methods, respectively. From Fig. 1a, Fig. 1b, Fig. 2a, Fig. 2b, Fig. 3a, and Fig. 3b, the



(a) Under strong Wolfe line search (b) Under Armijo-like line search

Fig. 3. Performance profile based on CPU time.



(a) Manipulator trajectories (b) End-effector trajectory and desired path

(c) Tracking errors on the horizontal x-axis (d) Tracking errors on the vertical y-axis

Fig. 4. Numerical results generated by NMLS.

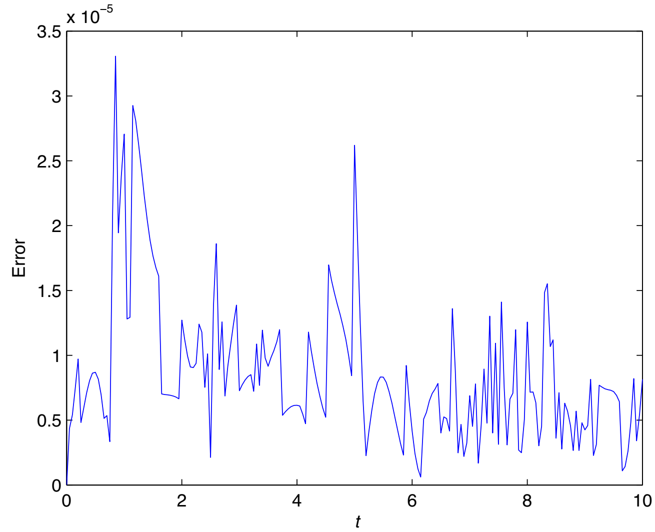


Fig. 5. Absolute error generated by NMLS.

proposed NMLS method has the best performance based on all metrics under strong Wolfe line search and Armijo-like line search.

3.1. Application in motion control

This subsection test the efficiency of Algorithm 1 in solving motion control problem. Algorithm 1 (NMLS method) is implemented using the Armijo-like line search (6), in which we set $\rho = 0.6, t = 10^{-14}$ and $\vartheta = 0.018$. We begin with a brief background on motion control problems (Zhang et al., 2019; Sun et al., 2020). Here we look only at the motion control of two-joint planar robot manipulator. Given the path of the vector desired $r_{dk} \in \mathscr{R}^2$ at time instant $t_k \in [0, t_f]$, the discrete-time kinematics equation is: finding $\theta_k \in \mathscr{R}^2$ such that

$$f(\theta_k) = r_{dk}.$$

f is a mapping given by

$$f(\theta) = \begin{bmatrix} l_1 c_1 + l_2 c_2 \\ l_1 s_1 + l_2 s_2 \end{bmatrix},$$

where $c_1 = \cos(\theta_1), s_1 = \sin(\theta_1), c_2 = \cos(\theta_1 + \theta_2), s_2 = \sin(\theta_1 + \theta_2)$, and l_i denote the i-th rod length. Obviously, this problem can be described as a minimization problem

$$\min_{\theta_k \in \mathscr{R}^2} \frac{1}{2} \|f(\theta_k) - r_{dk}\|^2. \quad (21)$$

For simplicity, we set $l_i = 1 (i = 1, 2)$ and the end-effector is controlled to track the following Lissajous curve (Zhang et al., 2019)

$$r_{dk} = \begin{bmatrix} 1.5 + 0.2\sin(\pi t_k/5) \\ \sqrt{3}/2 + 0.2\sin(2\pi t_k/5 + \pi/3) \end{bmatrix}.$$

The starting point of the initial problem is $\theta_0 = [0, \pi/3]^\top$, and that of any i-th problem is the approximate solution to the $i-1$-th problem $(i = 2, 3, \cdots)$. Furthermore the time for the task is $[0, 10]$, and divided equally into 200 pieces. The outcomes obtained by NMLS are presented in Fig. 4, which shows that NMLS algorithm effectively accomplished the task. More specifically, synthesization of the trajectories of the robot using the NMLS algorithm is plotted in Fig. 4a, and trajectory of end-effector as well as the path desired are plotted in Fig. 4b. Figs. 4c and 4d show the error generated by the NMLS method on x-axis and y-axis, respectively. It is not easy to see the variation of the generated end-effector trajectory and desired path from Fig. 4a and 4b. Actually, Fig. 4c and 4d depict that the errors are not above 10^{-5}. In addition, to show the absolute error generated by NMLS, we show their Euclidean distance of end-effector trajectory and desired path in Fig. 5, from which we find that the absolute error is always smaller than 3.5×10^{-5}. This shows that NMLS completes the task with high precision. In Fig. 4a there is one broken line that is far from other broken lines, which indicates that there is some sudden shift of the manipulator. This may cause damage to the manipulator. Therefore, in addition to the unconstrained optimization model (21), we need to design some new mathematical model of this motion control, which can generate continuous and repetitive motion control. This is one of our further research topics.

4. Conclusions

In this article, we have presented a Liu-Storey type conjugate gradient method that guarantees sufficient descent condition independent of the line search. Using the Wolfe and Armijo-like line search conditions, we proved that the NMLS method converges globally. Moreover, the presented method can inherit a built-in self-restarting mechanism of the LS method. Based on our numerical findings, we conclude that the proposed method is the most efficient and promising.

Acknowledgements

The authors acknowledge the financial support provided by the Center of Excellence in Theoretical and Computational Science (TaCS-CoE), KMUTT. Also, the (first) author, (Dr. Auwal Bala Abubakar) would like to thank the Postdoctoral Fellowship from King Mongkut's University of Technology Thonburi (KMUTT), Thailand. Moreover, this project is funded by National Research Council of Thailand (NRCT) under **Research Grants for Talented Mid-Career Researchers (**Contract no. N41A640089). Also, the first author acknowledge with thanks, the Department of Mathematics and Applied Mathematics at the Sefako Makgatho Health Sciences University.

Appendix A

See Tables 2 and 3.

Table 2
Numerical results for NMLS and NMPRP methods Under strong Wolfe Line Search.

Func.	Dim.	IP	NMPRP			NMLS		
			NOI	NOF	CPU Time	NOI	NOF	CPU Time
F1	50000	(1.1,...,1.1)	27	114	2.1063	10	59	0.9867
F1	100000	(1.1,...,1.1)	27	114	3.8852	10	58	1.9118
F2	50000	(0.1,1,...,0.1,1)	27	133	0.9388	16	75	0.5461
F2	100000	(0.1,1,...,0.1,1)	27	133	1.8728	16	75	1.0683

(*continued on next page*)

Table 2 (*continued*)

Func.	Dim.	IP	NMPRP			NMLS		
			NOI	NOF	CPU Time	NOI	NOF	CPU Time
F3	50000	(0.5,−2,…,0.5,−2)	–	–	–	8	38	0.2869
F3	100000	(0.5,−2,…,0.5,−2)	–	–	–	8	38	0.5809
F4	50000	(1,0.8,…,1,0.8)	16	56	1.0577	10	44	0.8081
F4	100000	(1,0.8,…,1,0.8)	16	56	2.1137	10	44	1.6154
F5	50	(2,…, 2)	66	177	0.0066	82	991	0.0254
F5	100	(2,…,2)	71	216	0.0125	220	1837	0.1142
F6	50000	(−2.1,…,−2.1)	83	217	4.164	8	48	0.7848
F6	100000	(−2.1,…,−2.1)	97	303	11.2897	8	47	1.5565
F7	50000	(0.1,…,0.1)	9	24	0.1955	8	21	0.1716
F7	100000	(0.1,…,0.1)	9	24	0.3747	8	21	0.3468
F8	50000	(5,…,5)	9	40	0.3006	13	103	0.678
F8	100000	(5,…,5)	9	40	0.5847	11	48	0.7318
F9	50000	(−5,…,−5)	39	175	1.222	48	363	2.3116
F9	100000	(−5,…,−5)	37	159	2.2551	55	246	3.3653
F10	50000	(8,…,8)	160	607	11.0454	195	799	14.369
F10	100000	(8,…,8)	245	1103	38.2768	84	421	14.5381
F11	50000	(1.05,…,1.05)	26	81	0.629	41	125	0.9895
F11	100000	(1.05,…,1.05)	30	97	1.5444	39	249	3.3063
F12	50000	(1,…,1)	7	22	0.1765	6	20	0.1677
F12	100000	(1,…,1)	7	22	0.3481	6	20	0.3149
F13	500	(−10,…,−10)	16	120	0.0167	22	152	0.0344
F13	1000	(−10,…,−10)	47	329	0.0638	37	284	0.0629
F14	50	(1.05, …, 1.05)	19	58	0.0018	22	116	0.0053
F14	100	(1.05,…,1.05)	25	76	0.0083	37	347	0.0165
F15	50	(1,…,1)	24	104	0.003	13	68	0.0039
F15	100	(1,…,1)	–	–	–	15	91	0.0095
F16	2	(−1.5,−2)	8	30	4.40E−03	7	28	4.05E−04
F16	2	(−5,−10)	13	70	0.0014	9	45	4.88E−04
F17	2	(−5,−5)	18	66	6.09E−04	18	507	0.0041
F17	2	(5,5)	18	66	0.0011	18	507	0.0026
F18	2	(5,5)	2	6	3.70E−03	2	6	2.86E−04
F18	2	(10,10)	2	6	1.50E−04	2	6	1.49E−04
F19	2	(−1,0.5)	1	3	9.78E−05	1	3	7.70E−05
F19	2	(−5,5)	10	36	5.95E−04	7	74	4.41E−04
F20	2	(0,0)	2	6	7.80E−03	2	6	1.19E−04
F20	2	(−5,5)	25	84	0.0017	14	67	8.19E−04
F21	50000	(1.001,…,1.001)	14	36	0.3292	5	19	0.1546
F21	100000	(1.001,…,1.001)	15	39	0.6534	5	19	0.2997
F22	50000	(1.001,…,1.001)	–	–	–	8	141	0.8757
F22	100000	(1.001,…,1.001)	–	–	–	9	169	1.9728
F23	100	(1.001,…,1.001)	58	176	0.0112	200	2568	0.0903
F23	1000	(1.001,…,1.001)	190	582	0.1224	328	2856	0.4666
F24	2	(−2.5,−2.5)	34	198	0.0016	22	163	0.0016
F24	2	(−8.5,−8.5)	53	429	0.0048	45	412	0.0034
F25	10	(8,…,8)	29	112	0.0033	39	146	0.0038
F25	100	(8,…,8)	37	155	0.0112	40	217	0.0155
F26	50	(8,…,8)	–	–	–	79	532	0.0127
F26	100	(8,…,8)	–	–	–	76	604	0.0388
F27	100	(3,…,3)	143	429	0.0215	146	438	0.0284
F27	1000	(3,…,3)	1574	4722	0.9696	1584	4753	0.9698
F28	1000	(1,…,1)	187	561	0.1345	209	872	0.175
F28	10000	(1,…,1)	606	1818	3.6627	695	2792	5.0326
F29	50	(1,…,1)	22	179	0.0106	29	298	0.0193
F29	500	(1,…,1)	42	401	0.0713	47	510	0.086
F30	50	(2.5,…,2.5)	10	44	0.0019	9	41	0.0023
F30	100	(2.5,…,2.5)	13	72	0.0075	9	61	0.0065
F31	4	(10,10,10,10)	44	222	0.0036	1581	9824	0.0624
F31	4	(100,…,100)	62	346	0.006	1102	6668	0.0515
F32	2	(1,1)	63	252	0.004	1	8	0.0019
F32	2	(20,20)	80	320	0.005	1	8	5.27E−04
F33	4	(1.01,…,1.01)	13	41	0.001	21	201	0.0018
F33	4	(1.05,…,1.05)	31	106	0.0032	63	542	0.0027
F34	10	(2.5,…,2.5)	67	271	0.0065	158	1743	0.0185
F34	100	(2.5,…,2.5)	224	969	0.0445	1494	16304	0.4986
F35	50000	(1,…,1)	1	3	0.0305	1	3	0.0345
F35	100000	(1,…,1)	1	3	0.0629	1	3	0.0534
F36	5000	(−1,…,−1)	435	1305	1.1682	477	1431	1.2797
F36	50000	(−1,…,−1)	1399	4197	32.1167	1578	4778	41.4889
F37	10000	(−1,…,−1)	13	37	0.1518	9	29	0.1121
F37	100000	(−1,…,−1)	13	37	1.2014	8	26	0.8156
F38	5000	(100,…,100)	17	100	0.1517	13	81	0.0767
F38	50000	(100,…,100)	17	100	0.7345	11	75	0.5609
F39	5000	(1.02,…,1.02)	23	72	0.0909	12	78	0.075
F39	50000	(1.02,…,1.02)	23	72	0.7157	13	87	0.7222

Table 2 (*continued*)

Func.	Dim.	IP	NMPRP			NMLS		
			NOI	NOF	CPU Time	NOI	NOF	CPU Time
F40	50000	(−1,...,−1)	1	3	0.041	1	3	0.0353
F40	100000	(−1,...,−1)	1	3	0.0684	1	3	0.064
F41	50000	(1,...,1)	44	213	1.4846	28	201	1.3216
F41	100000	(1,...,1)	39	208	2.84	28	201	2.6386
F42	100	(2.5,...,2.5)	703	3432	0.1174	245	2457	0.0901
F42	1000	(2.5,...,2.5)	877	4989	0.9623	692	7470	1.3607
F43	10	(1,...,1)	27	85	0.0019	20	263	0.0034
F43	50	(1,...,1)	29	91	0.0056	22	281	0.0131
F44	4	(8,...,8)	321	7917	0.0366	361	9535	0.0483
F44	8	(8,...,8)	344	8358	0.0532	269	7078	0.0449
F45	5000	(2.5,...,2.5)	445	1335	1.1661	479	1573	1.2648
F45	10000	(2.5,...,2.5)	633	1899	3.456	686	2355	4.1385
F46	50000	(4,...,4)	103	847	16.2004	5	51	0.9805
F46	100000	(4,...,4)	105	879	33.5407	5	51	1.9413
F47	100	(4,...,4)	245	3071	0.1165	134	2434	0.0815
F47	500	(4,...,4)	444	5685	0.6422	405	6495	0.7
F48	10	(1.1,...,1.1)	201	3610	0.0173	201	3715	0.0241
F48	50	(1.1,...,1.1)	1044	18579	0.0832	1044	18565	0.0705
F49	2	(1,1)	2	6	2.48E−04	2	6	2.07E−04
F49	2	(8,8)	1	3	2.48E−04	1	3	1.52E−04
F50	2	(1,1)	–	–	–	8	67	8.14E−04
F50	2	(−1,−1)	–	–	–	8	67	8.72E−04
F51	2	(2.5,2.5)	10	50	8.75E−04	10	53	8.20E−04
F51	2	(8,8)	27	170	0.0028	13	91	0.0012
F52	2	(4,4)	15	89	0.002	11	79	9.97E−04
F52	2	(1.5,1.5)	55	222	0.0035	18	103	0.0023
F53	2	(1,1)	10	23	5.97E−04	6	20	3.96E−04
F53	2	(−1,−1)	11	27	3.39E−04	4	11	2.95E−04
F54	500	(0.4,...,0.4)	398	5203	0.5884	327	5272	0.5708
F54	5000	(0.4,...,0.4)	2276	32993	24.4614	577	9468	6.7099
F55	10000	(1,2,...,10000)	27	304	0.4694	24	300	0.4533
F55	50000	(1,2,...,50000)	22	353	2.0584	17	260	1.4967
F56	100	(1,...,1)	21	100	0.0118	31	210	0.0108
F56	1000	(1,...,1)	35	203	0.0527	33	293	0.0549

Table 3
Numerical results for NMLS and NMPRP methods Under Armijo-like Line Search.

Func.	Dim.	IP	NMPRP			NMLS		
			NOI	NOF	CPU Time	NOI	NOF	CPU Time
F1	50000	(1.1,...,1.1)	343	8414	36.5449	125	1014	7.9958
F1	100000	(1.1,...,1.1)	334	8160	69.2095	177	1710	34.4755
F2	50000	(0.1,1,...,0.1,1)	274	6083	5.1683	121	1059	1.9011
F2	100000	(0.1,1,...,0.1,1)	274	6101	14.229	162	1282	5.7396
F3	50000	(0.5,−2,...,0.5,−2)	317	8333	8.125	–	–	–
F3	100000	(0.5,−2,...,0.5,−2)	285	7565	13.0259	–	–	–
F4	50000	(1,0.8,...,1,0.8)	193	2408	20.5074	80	640	5.3248
F4	100000	(1,0.8,...,1,0.8)	190	2420	58.852	165	1290	37.4979
F5	50	(2,...,2)	88	558	0.6429	102	770	0.0127
F5	100	(2,...,2)	145	1217	0.0393	217	2210	0.0353
F6	50000	(−2.1,...,−2.1)	722	1328	57.5115	863	867	31.0011
F6	100000	(−2.1,...,−2.1)	864	1718	154.1461	982	986	148.3827
F7	50000	(0.1,...,0.1)	163	2367	2.7513	91	776	1.3088
F7	100000	(0.1,...,0.1)	160	2282	5.3768	68	537	2.4787
F8	50000	(5,...,5)	143	2406	2.5317	54	385	0.8655
F8	100000	(5,...,5)	146	2456	5.1637	50	402	1.8747
F9	50	(−5,...,−5)	345	8110	0.0266	61	530	0.0057
F9	500	(−5,...,−5)	3015	67319	0.924	160	1760	0.0654
F10	100	(5,...,5)	293	4678	0.0934	1879	24291	0.3753
F10	1000	(5,...,5)	373	5820	0.7372	1899	22603	2.9481
F11	50000	(1.05,...,1.05)	170	2366	3.1738	52	438	0.8185
F11	100000	(1.05,...,1.05)	–	–	–	49	456	1.7689
F12	50000	(1,...,1)	86	635	1.2963	29	58	0.4912
F12	100000	(1,...,1)	81	570	2.5959	29	58	0.902
F13	50	(−10,...,−10)	129	1896	0.0146	55	499	0.0057
F13	100	(−10,...,−10)	132	2282	0.0242	101	778	0.018
F14	50	(1.05,...,1.05)	95	685	0.009	40	313	0.0048
F14	100	(1.05,...,1.05)	128	1052	0.0307	50	332	0.0124
F15	50	(1,...,1)	211	4583	0.0353	222	2895	0.0241
F15	100	(1,...,1)	196	4341	0.0433	246	2503	0.0344

(*continued on next page*)

Table 3 (*continued*)

Func.	Dim.	IP	NMPRP			NMLS		
			NOI	NOF	CPU Time	NOI	NOF	CPU Time
F16	2	(−1.5,−2)	91	856	1.76E−01	31	171	6.46E−04
F16	2	(−5,−10)	85	839	0.0045	19	140	0.0012
F17	2	(−5,−5)	–	–	–	22	153	0.0016
F17	2	(5,5)	–	–	–	22	153	9.63E−04
F18	2	(5,5)	90	783	3.70E−03	33	85	1.10E−03
F18	2	(10,10)	103	912	0.0037	41	105	0.0019
F19	2	(−1,0.5)	42	1169	9.60E−02	21	78	7.35E−04
F19	2	(−5,5)	86	613	0.0049	39	334	0.0028
F20	2	(0,0)	42	335	3.60E−03	16	48	0.0171
F20	2	(−5,5)	60	431	0.0021	174	2401	0.006
F21	50000	(1.001,…,1.001)	61	458	0.9017	53	425	0.7686
F21	100000	(1.001,…,1.001)	64	483	1.779	98	915	2.824
F22	50000	(1.001,…,1.001)	–	–	–	25	51	0.3194
F22	100000	(1.001,…,1.001)	–	–	–	26	53	0.6645
F23	100	(1.001,…,1.001)	128	2640	0.0262	121	1352	0.1293
F23	1000	(1.001,…,1.001)	346	9686	0.1551	444	5068	0.1918
F24	2	(−2.5,−2.5)	248	6066	0.1254	243	2652	0.0091
F24	2	(5,5)	210	5233	0.0132	275	3165	0.0062
F25	10	(8,…,8)	156	2024	0.0145	50	334	0.0047
F25	50	(8,…,8)	174	2333	0.0403	54	444	0.007
F26	50	(8,…,8)	–	–	–	88	558	0.0074
F26	100	(8,…,8)	–	–	–	111	721	0.0141
F27	50	(3,…,3)	542	15491	0.0399	1088	11492	0.0476
F27	100	(3,…,3)	1119	37179	0.1394	1871	22067	0.1567
F28	1000	(1,…,1)	351	8188	0.1416	618	5406	1.2305
F28	10000	(1,…,1)	1141	35261	5.9839	7386	108554	37.9789
F29	50	(1,…,1)	311	10782	0.0308	3486	74335	0.3662
F29	500	(1,…,1)	461	19621	0.2568	378	4864	0.1917
F30	10	(2.5,…,2.5)	132	1572	0.0061	35	261	7.87E−04
F30	50	(2.5,…,2.5)	141	2356	0.0123	36	200	2.35E−01
F31	4	(10,10,10,10)	239	5724	0.022	393	3887	0.0329
F31	4	(100,…,100)	325	8204	0.0209	371	3912	0.021
F32	2	(1,1)	269	269	0.0055	269	269	0.0067
F32	2	(20,20)	342	342	0.0067	342	342	0.0056
F33	4	(1.01,…,1.01)	239	5802	0.0134	60	630	0.0059
F33	4	(1.05,…,1.05)	267	6510	0.0136	54	436	0.0024
F34	10	(2.5,…,2.5)	151	1872	0.0074	131	1067	0.0055
F34	100	(2.5,…,2.5)	677	15030	0.0698	683	4526	0.0581
F35	50000	(1,…,1)	58	1847	1.0773	29	58	0.3573
F35	100000	(1,…,1)	63	2447	2.4575	30	60	0.6896
F36	1000	(−1,…,−1)	404	10324	0.1742	733	7743	0.2627
F36	5000	(−1,…,−1)	855	26517	1.4923	2773	41336	4.8455
F37	10000	(−1,…,−1)	86	601	1.075	24	147	0.2807
F37	100000	(−1,…,−1)	91	660	8.7063	34	251	3.2634
F38	5000	(100,…,100)	167	3485	0.3386	49	369	0.0935
F38	50000	(100,…,100)	180	3727	3.6781	49	380	0.7706
F39	5000	(1.02,…,1.02)	–	–	–	34	211	0.0731
F39	50000	(1.02,…,1.02)	–	–	–	28	168	0.5191
F40	50000	(−1,…,−1)	–	–	–	39	340	0.803
F40	100000	(−1,…,−1)	–	–	–	39	331	1.4508
F41	50	(1,…,1)	–	–	–	2546	26486	0.14
F41	500	(1,…,1)	–	–	–	6631	85547	1.8229
F42	100	(2.5,…,2.5)	254	8876	0.0498	1519	32237	0.1569
F42	1000	(2.5,…,2.5)	264	8837	0.1669	1155	25694	0.6078
F43	10	(1,…,1)	117	1215	0.0056	36	257	0.0027
F43	50	(1,…,1)	120	1228	0.0108	36	251	0.0032
F44	4	(8,…,8)	163	1845	0.0066	50	443	0.0023
F44	8	(8,…,8)	126	1576	0.0056	70	680	0.0031
F45	5000	(2.5,…,2.5)	911	28172	1.7493	3566	54740	5.7253
F45	10000	(2.5,…,2.5)	1319	43903	7.0819	1523	15798	6.044
F46	50000	(2,…,2)	23	2486	20.6921	1	2	0.0436
F46	100000	(2,…,2)	23	2500	39.4022	1	2	0.0713
F47	100	(4,…,4)	116	2750	0.0263	171	2152	0.0273
F47	500	(4,…,4)	123	2900	0.0584	140	1711	0.0594
F48	10	(1.1,…,1.1)	82	1035	0.0067	30	262	0.0025
F48	50	(1.1,…,1.1)	79	1114	0.0077	23	155	9.23E−04
F49	2	(1,1)	51	1534	0.0037	26	51	0.0016
F49	2	(8,8)	56	1873	0.0043	26	52	6.77E−04
F50	2	(1,1)	–	–	–	–	–	–
F50	2	(−1,−1)	–	–	–	–	–	–
F51	2	(2.5,2.5)	135	2608	0.0077	30	152	0.0019
F51	2	(8,8)	156	3088	0.0148	34	184	0.0019
F52	2	(4,4)	137	2918	0.0102	487	5529	0.0249
F52	2	(1.5,1.5)	92	1244	0.0046	2570	41676	0.0856

Table 3 (*continued*)

Func.	Dim.	IP	NMPRP			NMLS		
			NOI	NOF	CPU Time	NOI	NOF	CPU Time
F53	2	(1,1)	58	2880	0.0063	26	151	0.0017
F53	2	(−1,−1)	37	516	0.002	24	238	0.0012
F54	500	(0.4,. . .,0.4)	117	2579	0.0506	245	2992	0.0759
F54	5000	(0.4,. . .,0.4)	133	3089	0.2593	242	2559	0.382
F55	100	(1,2,. . .,100)	120	1531	0.0217	30	232	0.005
F55	500	(1,2,. . .,500)	125	2301	0.0566	43	429	0.032
F56	100	(1,. . .,1)	163	2736	0.031	109	1009	0.022
F56	1000	(1,. . .,1)	441	12977	0.4352	176	1756	0.0878

Appendix B. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.jksus.2022.101923.

References

Abubakar, A.B., Kumam, P., Malik, M., Chaipunya, P., Ibrahim, A.H., 2021. A hybrid FR-DY conjugate gradient algorithm for unconstrained optimization with application in portfolio selection. AIMS Math. 6 (6), 6506–6527.

Abubakar, A.B., Kumam, P., Malik, M., Ibrahim, A.H., 2021. A hybrid conjugate gradient based approach for solving unconstrained optimization and motion control problems. Math. Comput. Simul.

Al-Baali, M., 1985. Descent property and global convergence of the Fletcher-Reeves method with inexact line search. IMA J. Numer. Anal. 5 (1), 121–124.

Aminifard, Z., Saman, B-K., 2019. A modified descent Polak–Ribiére–Polyak conjugate gradient method with global convergence property for nonconvex functions. Calcolo 56 (2), 16.

Andrei, N., 2011. A modified Polak–Ribière–Polyak conjugate gradient algorithm for unconstrained optimization. Optimization 60(12), 1457–1471..

Andrei, N., 2017. Accelerated adaptive perry conjugate gradient algorithms based on the self-scaling memoryless bfgs update. J. Comput. Appl. Math. 325, 149–164.

Dai, Y., Liao, L., 2001. New conjugacy conditions and related nonlinear conjugate gradient methods. Appl. Math. Optim. 43 (1), 87–101.

Dai, Y., Han, J., Liu, G., Sun, D., Yin, H., Yuan, Y., 2000. Convergence properties of nonlinear conjugate gradient methods. SIAM J. Optim. 10 (2), 345–358.

Ding, J., Liu, Y., Ding, F., 2010. Iterative solutions to matrix equations of the form aixbi=fi. Comput. Math. Appl. 59 (11), 3500–3507.

Dolan, E.D., Moré, J.J., 2002. Benchmarking optimization software with performance profiles. Math. Program. 91 (2), 201–213.

Dong, X., Han, D., Ghanbari, R., Li, X., Dai, Z., 2017. Some new three-term Hestenes-Stiefel conjugate gradient methods with affine combination. Optimization 66 (5), 759–776.

Gilbert, J.C., Nocedal, J., 1992. Global convergence properties of conjugate gradient methods for optimization. SIAM J. Optim. 2 (1), 21–42.

Grippo, L., Lucidi, S.A., 1997. globally convergent version of the polak-ribière conjugate gradient method. Math. Program. 78 (3), 375–391.

Hager, W., Zhang, H., 2005. A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J. Optim. 16 (1), 170–192.

Hager, W., Zhang, H.A., 2006. survey of nonlinear conjugate gradient methods. Pacific J. Optim. 2 (1), 35–58.

Hager, W., Zhang, H., 2013. The limited memory conjugate gradient method. SIAM J. Optim. 23 (4), 2150–2168.

Hestenes, M.R., Stiefel, E., 1952. Methods of conjugate gradients for solving linear systems. J. Res. Natl. Bureau Standards 49 (6), 409–436.

Jamil, M., Yang, X-S., 2013. A literature survey of benchmark functions for global optimisation problems. Int. J. Math. Model. Numer. Optim. 4 (2), 150–194.

Li, M., Feng, H., 2011. A sufficient descent LS conjugate gradient method for unconstrained optimization problems. Appl. Math. Comput. 218 (5), 1577–1586.

Liang, Dong X., Liu, H., He, Y., Yang, X., 2015. A modified Hestenes-Stiefel conjugate gradient method with sufficient descent condition and conjugacy condition. J. Comput. Appl. Math. 281, 239–249.

Liu, J., Feng, Y., 2011. Global convergence of a modified LS nonlinear conjugate gradient method. Proc. Eng. 15, 4357–4361.

Liu, Y., Storey, C., 1991. Efficient generalized conjugate gradient algorithms, part 1: theory. J. Optim. Theory Appl. 69 (1), 129–137.

Malik, M., Abubakar, A.B., Sulaiman, I.M., Mamat, M., Abas, S.S., Sukono. 2021. A new three-term conjugate gradient method for unconstrained optimization with applications in portfolio selection and robotic motion control. Int. J. Appl. Math. 51(3)..

Polak, E., Ribiere, G., 1969. Note sur la convergence de méthodes de directions conjuguées. Revue française d'informatique et de recherche opérationnelle, 3. Série rouge, pp. 35–43.

Polyak, B.T., 1969. The conjugate gradient method in extremal problems. USSR Computational Mathematics and Mathematical Physics 9 (4), 94–112.

Saman, B.-K., Ghanbari, R., 2014. A descent extension of the Polak–Ribière–Polyak conjugate gradient method. Comput. Math. Appl. 68(12), 2005–2011..

Shi, Z., Shen, J., 2007. Convergence of Liu-Storey conjugate gradient method. Eur. J. Oper. Res. 182 (2), 552–560.

Sun, W., Yuan, Y. Optimization theory and methods: Nonlinear programming. Springer (992)..

Sun, M., Liu, J., Wang, Y., 2020. Two improved conjugate gradient methods with application in compressive sensing and motion control. Math. Problems Eng. 2020.

Tang, C.M., Wei, Z., Li, Li, 2007. G.A new version of the Liu-Storey conjugate gradient method. Appl. Math. Comput. 189 (1), 302–313.

Xie, Li, Ding, Jie, Ding, Feng, 2009. Gradient based iterative solutions for general linear matrix equations. Comput. Math. Appl. 58 (7), 1441–1448.

Xie, Li, Liu, Yanjun, Yang, Huizhong, 2010. Gradient based and least squares based iterative algorithms for matrix equations axb+cxtd=f. Appl. Math. Comput. 217 (5), 2191–2199.

Yuan, G., Wei, Z., Yang, Y., 2019. The global convergence of the Polak–Ribière–Polyak conjugate gradient algorithm under inexact line search for nonconvex functions. J. Comput. Appl. Math. 362, 262–275.

Yuan, G., Lu, J., Wang, Z., 2020. The PRP conjugate gradient algorithm with a modified wwp line search and its application in the image restoration problems. Appl. Numer. Math. 152, 1–11.

Yuan, G., Lu, J., Wang, Z., 2021. The modified PRP conjugate gradient algorithm under a non-descent line search and its application in the muskingum model and image restoration problems. Soft. Comput. 25 (8), 5867–5879.

Yu, G., Guan, L., Li, G., 2008. Global convergence of modified Polak-Ribière-Polyak conjugate gradient methods with sufficient descent property. J. Ind. Manage. Optim. 4 (3), 565.

Zhang L., 2009. A new Liu–Storey type nonlinear conjugate gradient method for unconstrained optimization problems. J. Comput. Appl. Math. 225(1), 146–157..

Zhang, L., Zhou, W., Li, D., 2006. A descent modified Polak–Ribière–Polyak conjugate gradient method and its global convergence. IMA J. Numer. Anal. 26 (4), 629–640.

Zhang, L., Zhou, W., Li, D., 2007. Some descent three-term conjugate gradient methods and their global convergence. Optim. Methods Software 22 (4), 697–711.

Zhang, Y., He, L., Hu, C., Guo, J., Li, J., Shi, Y., 2019. General four-step discrete-time zeroing and derivative dynamics applied to time-varying nonlinear optimization. J. Comput. Appl. Math. 347, 314–329.