AIMS Mathematics, 9(10): 27535–27556.
DOI: 10.3934/math.20241337
Received: 29 July 2024
Revised: 07 September 2024
Accepted: 13 September 2024
Published: 23 September 2024

https://www.aimspress.com/journal/Math

Research article

A new hybrid conjugate gradient method close to the memoryless BFGS quasi-Newton method and its application in image restoration and machine learning

Xiyuan Zhang and Yueting Yang*

School of Mathematics and Statistics, Beihua University, Jilin, Jilin, 132013, China

* **Correspondence:** Email: yyt2858@163.com.

Abstract: A new hybrid conjugate gradient algorithm for solving the unconstrained optimization problem was presented. The algorithm could be considered as a modification of the memoryless Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method. Based on a normalized gradient difference, we introduced a new combining conjugate gradient direction close to the direction of the memoryless BFGS quasi-Newton direction. It was shown that the search direction satisfied the sufficient descent property independent of the line search. For general nonlinear functions, the global convergence of the algorithm was proved under standard assumptions. Numerical experiments indicated a potential performance of the new algorithm, especially for solving the large-scale problems. In addition, the proposed method was used in practical application problems for image restoration and machine learning.

Keywords: conjugate gradient; memoryless BFGS; wolfe line search; global convergence
image restoration
Mathematics Subject Classification: 90C06, 90C30, 65K05

1. Introduction

In this work, we consider the following unconstrained optimization problem:

$$\min f(x), \ x \in \mathbb{R}^n, \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below. Conjugate gradient methods are highly significant for solving large-scale optimization due to their lower storage requirements and simple computation. They are widely applied in various fields, including information technology [1], energy engineering [2, 3], material science [4, 5], and more. The conjugate gradient method for

solving (1.1) generates an iterate sequence $\{x_k\}$ using the formula

$$x_{k+1} = x_k + \alpha_k d_k,\ k = 0, 1, 2...,$$

where the scalar $\alpha_k > 0$ is the step size and satisfies the Wolfe line search conditions

$$f(x_k + \alpha d_k) - f(x_k) \leq \rho\alpha g_k^T d_k, \tag{1.2}$$

$$g(x_k + \alpha d_k)^T d_k \geq \sigma g_k^T d_k, \tag{1.3}$$

where $g_k = \nabla f(x_k)$, $0 < \rho \leq \sigma \leq 1$. The search direction d_k is generally given by

$$d_k = \begin{cases} -g_k, & \text{if } k = 1, \\ -g_k + \beta_k d_{k-1}, & \text{if } k \geq 2, \end{cases} \tag{1.4}$$

where $\beta_k \in \mathbb{R}^n$ is the conjugate parameter.

Some of the famous conjugate gradient methods are the Fletcher-Reeves (FR) method [6], the Polak-Ribiere-Polyak (PRP) method [7, 8], the Hestenes-Stiefel (HS) method [9], and the Dai-Yuan (DY) method [10], the Liu–Storey (LS) method [11], and their parameters β_k are, respectively,

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

where $y_{k-1} = g_k - g_{k-1}$. $\|\cdot\|$ represents the Euclidean norm.

Different conjugate parameter choices lead to variants of the conjugate gradient method, each with its own unique strengths and differences in theoretical convergence and numerical performance. In recent years, the modified conjugate gradient method [12–15] and hybrid conjugate gradient method [16,17] have received much attention. Their researches exhibited better convergence properties and broader applications and illustrated the significance of our in-depth study of the conjugate gradient method.

In 2006, Wei et al. [18] proposed a modification of the PRP conjugate gradient method, denoted as the Wei-Yao-Liu (WYL) method by substituting y_{k-1} with $y_{k-1}^* = g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1}$ in β_k^{PRP}, and introduced the new conjugate parameter

$$\beta_k^{WYL} = \frac{g_k^T y_{k-1}^*}{\|g_{k-1}\|^2}. \tag{1.5}$$

When the normalized gradient difference y_{k-1}^* tended to zero, leading that the scalar β_k^{WYL} also approached to zero, which in turn caused the search direction to be close to the steepest descent direction, it implied that the WYL method possessed the automatic restart feature, similar to the PRP method. Furthermore, when the gradients g_k and g_{k-1} were significantly different, the smaller one between g_k and g_{k-1} was almost negligible in the computation of y_{k-1}. The fact $\|\frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1}\| = \|g_k\|$ guaranteed that the vector y_{k-1}^* could utilize the available information both of g_k and g_{k-1} and enhanced the numerical stability. More researches on y_{k-1}^* could be found in [12–14, 16, 19].

In recent years, the memoryless BFGS method has been commonly employed in constructing the hybrid conjugate gradient method. In 2018, Li [15] rewrote the memeoryless BFGS update d_k^{BFGS} in [20] as

$$d_k^{BFGS} := -g_k + \left(\beta_k^{HS} - \frac{\|y_{k-1}\|^2 g_k^T d_{k-1}}{(d_{k-1}^T y_{k-1})^2}\right) d_{k-1} + \frac{g_k^T d_{k-1}}{d_{k-1}^T y_{k-1}} (y_{k-1} - s_{k-1}), \tag{1.6}$$

and proposed a new conjugate direction by closing to the memoryless BFGS quasi-Newton method

$$d_k^{THS} := -g_k + \left(\beta_k^{HS} - \frac{\|y_{k-1}\|^2 g_k^T d_{k-1}}{(d_{k-1}^T y_{k-1})^2}\right) d_{k-1} + t_k \frac{g_k^T d_{k-1}}{d_{k-1}^T y_{k-1}} y_{k-1}, \tag{1.7}$$

where t_k was established by minimizing the distance between $(y_{k-1} - s_{k-1})$ and $t_k y_{k-1}$. Moreover, such a constructed conjugate gradient method was considered as a three-term conjugate gradient method determined by vectors g_k, d_{k-1}, and y_{k-1}. The method retains the advantages of the HS method and memoryless BFGS method, and it was also suitable for large-scale optimization problems.

In 2023, Kumam [17] presented a hybrid conjugate gradient method for solving (1.1). The direction combines the three-term PRP, HS, and LS directions.

$$d_k^{\text{HTTHSLS}} = -g_k + \left(\frac{g_k^T y_{k-1}}{w_k} - \frac{\|y_{k-1}\|^2 g_k^T d_{k-1}}{w_k^2}\right) d_{k-1} + t_k \frac{g_k^T d_{k-1}}{w_k} y_{k-1}, \quad k \geq 1, \tag{1.8}$$

where

$$w_k := \max\{\mu \|d_{k-1}\| \|y_{k-1}\|, d_{k-1}^T y_{k-1}, -d_{k-1}^T g_{k-1}\}, \quad \mu > 0,$$

and t_k is as well as that of (1.7). Furthermore, the direction was close to the memoryless BFGS quasi-Newton method and had the descent and trust region properties.

In addition, conjugate gradient methods have been employed to deal with the image restoration efficiently. In 2020, Luo et. al [21] proposed a conjugate gradient algorithm based on double parameter scaled BFGS and applied it to solve image restoration problems. In 2023, Jiang et. al [22] proposed a family of hybrid conjugate gradient methods with restart procedure for unconstrained optimizations and image restorations. In 2024, Jiang et. al [23] put forward a family of spectral conjugate gradient methods with strong convergence and its applications in image restoration and machine learning. More related researches are detailed in the references [21, 24–27].

Inspired by the above works, we present a hybrid conjugate gradient method by constructing new combined conjugate parameters for solving (1.1). The main contributions of this paper are stated as follows.

• The new method can be viewed as a three-term conjugate gradient method in which the search direction is a linear combination of g_k, d_{k-1}, y_{k-1}^*. Their coefficients are defined as a combination of PRP, HS, LS, and WYL conjugate paremeters. The new direction is close to the memoryless BFGS quasi-Newton direction.

• The search directions generated by the proposed method satisfy the sufficient descent condition independent of any line search. Under usual assumptions and the Wolfe line search, the global convergence of our algorithm is proved.

• Numerical experiments of the proposed method are carried out for solving unconstrained test problems, and we apply the new method in the image restoration problems and machine learning problems.

The rest of this paper is organized as follows. In the next section, we put forward a new hybrid three-term conjugate gradient method and provide a framework of the algorithm. The sufficient descent property and global convergence are demonstrated in Section 2. In Section 3, some numerical experiments are implemented for solving some unconstrained test problems. In Section 4, we use the new algorithm to solve image restoration problems. In Section 5, a conclusion for this work is made.

2. New algorithm and convergence analysis

In this work, motivated by the three-term conjugate gradient direction defined in (1.8) and taking the advantage of the β_k^{WYL} and y_{k-1}^* in [18], we define a new three-term hybrid conjugate gradient method with the following search direction:

$$d_k^{HTTWYL} := -g_k + \beta_k d_{k-1} + \gamma_k y_{k-1}^*, \quad k \geq 1, \tag{2.1}$$

where

$$y_{k-1}^* = g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1},$$

$$\beta_k := \frac{g_k^T y_{k-1}^*}{\eta_k} - \frac{\|y_{k-1}^*\|^2 g_k^T d_{k-1}}{\eta_k^2}, \ \gamma_k := t_k \frac{g_k^T d_{k-1}}{\eta_k}, \tag{2.2}$$

and

$$\eta_k := \max\{\mu \|d_{k-1}\| \|y_{k-1}\|, \mu \|d_{k-1}\| \|y_{k-1}^*\|, d_{k-1}^T y_{k-1}, -d_{k-1}^T g_{k-1}, \|g_{k-1}\|^2\}, \quad \mu > 0.$$

When $\eta_k = \|g_{k-1}\|^2, \beta_k = \beta_k^{WYL} - \frac{\|y_{k-1}^*\|^2 g_k^T d_{k-1}}{\eta_k^2}$ could be regarded as a modification of β_k^{WYL}.

To obtain the parameter t_k, similar to the treatment of t_k in (1.7), we require the solution of the univariate minimal problem

$$\min_{t \in \mathbf{R}} \|(y_{k-1} - s_{k-1}) - t \cdot y_{k-1}^*\|^2.$$

Setting $M_k = (y_{k-1} - s_{k-1}) - t \cdot y_{k-1}^*$,

$$\begin{aligned} M_k M_k^T &= [(y_{k-1} - s_{k-1}) - t \cdot y_{k-1}^*][(y_{k-1} - s_{k-1}) - t \cdot y_{k-1}^*]^T \\ &= [(y_{k-1} - s_{k-1}) - t \cdot y_{k-1}^*][(y_{k-1} - s_{k-1})^T - t \cdot y_{k-1}^{*T}] \\ &= t^2 y_{k-1}^* y_{k-1}^{*T} - t[(y_{k-1} - s_{k-1}) y_{k-1}^{*T} + y_{k-1}^* (y_{k-1} - s_{k-1})^T] + (y_{k-1} - s_{k-1})(y_{k-1} - s_{k-1})^T. \end{aligned}$$

Setting $A_k = y_{k-1} - s_{k-1}$,

$$M_k M_k^T = t^2 y_{k-1}^* y_{k-1}^{*T} - t[A_k y_{k-1}^{*T} + y_{k-1}^* A_k^T] + A_k A_k^T,$$

and

$$\begin{aligned} tr(M_k M_k^T) &= t^2 \|y_{k-1}^*\|^2 - t[tr(A_k y_{k-1}^{*T}) + tr(y_{k-1}^* A_k^T)] + \|A_k\|^2 \\ &= t^2 \|y_{k-1}^*\|^2 - 2t y_{k-1}^{*T} A_k + \|A_k\|^2. \end{aligned}$$

Taking the derivative of the above equation and setting it equal to zero, we obtain

$$2t \left\|y_{k-1}^*\right\|^2 - 2 y_{k-1}^{*T} A_k = 0,$$

which implies

$$t = \frac{y_{k-1}^{*T}(y_{k-1} - s_{k-1})}{\|y_{k-1}^*\|^2}. \tag{2.3}$$

Therefore, we choose t_k as

$$t_k := \min\left\{\bar{t}, \max\left\{0, \frac{y_{k-1}^{*T}(y_{k-1} - s_{k-1})}{\|y_{k-1}^*\|^2}\right\}\right\}, \tag{2.4}$$

which implies $0 \leq t_k \leq \bar{t} < 1$. Note that the direction defined by (2.1) is close to the direction of the memoryless BFGS method when t_k is chosen by (2.3).

Based on the above analysis, the new algorithm can be presented as follows.

Algorithm 1 (HTTWYL)

Step 0: Input $x_0 \in \mathbb{R}^n$, parameters $\varepsilon > 0$, $0 < \rho < \sigma < 1$. $d_1 = -g_1$, set $k := 1$;

Step 1: If $\|g_k\| \leq \varepsilon$, then stop.

Step 2: Determine a step-length α_k by the Wolfe line search (1.2) and (1.3);

Step 3: Let $x_{k+1} = x_k + \alpha_k d_k$, calculate g_{k+1}, $f(x_{k+1})$;

Step 4: Determine t_k, γ_k, and β_k as in (2.4) and (2.2), respectively. Compute the search direction as in (2.1).

Step 5: Set $k := k + 1$, and go to Step 1.

The following lemma shows that the d_k produced by (2.1) satisfies the property of sufficient descent without any line search.

Lemma 2.1. If $\{d_k\}$ is defined by (2.1), then we obtain

$$g_k^T d_k \leq -c_1\|g_k\|^2,$$

where $c_1 = \left(1 - \frac{(1+\bar{t})^2}{4}\right)$.

Proof.

$$\begin{aligned}
g_k^T d_k &= -\|g_k\|^2 + \frac{g_k^T y_{k-1}^*}{\eta_k} g_k^T d_{k-1} - \frac{\|y_{k-1}^*\|^2}{\eta_k^2}(g_k^T d_{k-1})^2 + t_k \frac{g_k^T y_{k-1}^*}{\eta_k} g_k^T d_{k-1} \\
&= -\|g_k\|^2 + (1 + t_k)\frac{g_k^T y_{k-1}^*}{\eta_k} g_k^T d_{k-1} - \frac{\|y_{k-1}^*\|^2}{\eta_k^2}(g_k^T d_{k-1})^2 \\
&= -\|g_k\|^2 + 2\left(\frac{(1 + t_k)}{2} g_k^T\right)\frac{y_{k-1}^*}{\eta_k} g_k^T d_{k-1} - \frac{\|y_{k-1}^*\|^2}{\eta_k^2}(g_k^T d_{k-1})^2 \\
&\leq -\|g_k\|^2 + \frac{(1 + t_k)^2}{4}\|g_k\|^2 + \frac{\|y_{k-1}^*\|^2}{\eta_k^2}(g_k^T d_{k-1})^2 - \frac{\|y_{k-1}^*\|^2}{\eta_k^2}(g_k^T d_{k-1})^2 \\
&= -\|g_k\|^2 + \frac{(1 + t_k)^2}{4}\|g_k\|^2 \\
&\leq -\left(1 - \frac{(1 + \bar{t})^2}{4}\right)\|g_k\|^2,
\end{aligned} \tag{2.5}$$

so the proof is complete. ■

Next, we will attempt to establish the convergence of the standard Wolfe line search conditions. We use the following assumptions.

Assumption 1. The level set $D = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded. It means

$$\|x\| \leq B.$$

Assumption 2. The gradient function $g(x)$ is Lipschitz continuous on the level set D, exists a constant L, s.t.,

$$\|g(x) - g(y)\| \leq L\|x - y\|, \quad x, y \in \mathbb{R}^n.$$

Based on the above assumptions, it can be shown that there exists a constant $\Gamma > 0$ such that

$$\|g(x)\| \leq \Gamma, \quad x \in D.$$

Theorem 2.1. Let the sequence $\{x_k\}$ be generated by Algorithm 1, and suppose the Assumptions 1 and 2 holds, If

$$\sum_{k=0}^{\infty} \frac{1}{\|d_k\|^2} = +\infty, \tag{2.6}$$

then

$$\liminf_{k\to\infty} \|g_k\| = 0. \tag{2.7}$$

Proof. By paradoxically assuming that (2.7) does not hold, then there exists a nonnegative constant ζ such that

$$\|g_k\| \geq \zeta. \tag{2.8}$$

From (2.2), we have

$$\begin{aligned} |\beta_k| &= \left| \frac{g_k^T y_{k-1}^*}{\eta_k} - \frac{\|y_{k-1}^*\|^2 g_k^T d_{k-1}}{\eta_k^2} \right| \\ &\leq \frac{\|g_k\|\|y_{k-1}^*\|}{\mu\|d_{k-1}\|\|y_{k-1}^*\|} + \frac{\|y_{k-1}^*\|^2\|g_k\|\|d_{k-1}\|}{(\mu\|d_{k-1}\|\|y_{k-1}^*\|)^2} \\ &= \left(\frac{1}{\mu} + \frac{1}{\mu^2}\right) \frac{\|g_k\|}{\|d_{k-1}\|}. \end{aligned} \tag{2.9}$$

Also,

$$\begin{aligned} |\gamma_k| &= \left| t_k \frac{g_k^T d_{k-1}}{\eta_k} \right| \\ &\leq \bar{t} \frac{\|g_k\|\|d_{k-1}\|}{\eta_k} \\ &\leq \bar{t} \frac{\|g_k\|\|d_{k-1}\|}{\mu\|d_{k-1}\|\|y_{k-1}^*\|} \end{aligned}$$

$$= \frac{\bar{t}}{\mu}\frac{\|g_k\|}{\|y_{k-1}^*\|}. \tag{2.10}$$

Then, from formula (2.1), (2.9), and (2.10), we obtain

$$\begin{aligned}\|d_k\| &= \| - g_k + \beta_k d_{k-1} + \gamma_k y_{k-1}^*\| \\ &\leq \|g_k\| + |\beta_k|\|d_{k-1}\| + |\gamma_k|\|y_{k-1}^*\| \\ &\leq \|g_k\| + \left(\frac{1}{\mu} + \frac{1}{\mu^2}\right)\frac{\|g_k\|}{\|d_{k-1}\|}\|d_{k-1}\| + \frac{\bar{t}}{\mu}\frac{\|g_k\|}{\|y_{k-1}^*\|}\|y_{k-1}^*\| \\ &= \left(1 + \frac{1+\bar{t}}{\mu} + \frac{1}{\mu^2}\right)\|g_k\| \\ &= \left(1 + \frac{1+\bar{t}}{\mu} + \frac{1}{\mu^2}\right)\Gamma. \end{aligned} \tag{2.11}$$

Letting $M_1 = \left(1 + \frac{1+\bar{t}}{\mu} + \frac{1}{\mu^2}\right)\Gamma$, we have

$$\|d_k\| \leq M_1.$$

From Lemma 2.1, the first Wolfe condition (1.2), and (2.8),

$$f(x_{k+1}) \leq f(x_k) + \rho\alpha_k g_k^T d_k \leq f(x_k) - \rho\alpha_k c_1\|g_k\|^2 < f(x_k) < f(x_{k-1}) < \cdots < f(x_0).$$

Conjugating with Assumption 1, the sequence $\{f(x_k)\}$ is convergent.

From the second Wolfe condition (1.3) and Assumption 2, we obtain

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

Using the above inequality and (1.2), we derive

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

Summing both sides of the above equation for $k = 0, 1, ..., K$, we get

$$\frac{\rho(1-\sigma)}{L}\sum_{k=0}^{K}\frac{(g_k^T d_k)^2}{\|d_k\|^2} \leq \sum_{k=0}^{K}(f(x_k) - f(x_{k+1})) = f(x_0) - f(x_{K+1}).$$

Let $K \to \infty$, and the above implies that

$$\sum_{k=0}^{\infty}\frac{(g_k^T d_k)^2}{\|d_k\|^2} < +\infty. \tag{2.12}$$

Now, combining (2.8) with (2.5), we get

$$g_k^T d_k \leq -\left(1 - \frac{(1+\bar{t})^2}{4}\right)\|g_k\|^2$$

$$\leq -\left(1 - \frac{(1+\bar{t})^2}{4}\right)\zeta^2. \tag{2.13}$$

Therefore, we have

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

this contradicts (2.6). Then, (2.7) holds. ■

3. Numerical experiments

In this section, we compare the computational performance of HTTWYL with the method proposed by Hager and Zhang [28] (HZ), the method proposed by MinLi [15] (NHS+), and the method by Kumam [17] (HTTHSLS) under standard Wolfe line search. All codes are written on Matlab R2022a and run on a PC with the 2200MHz central processing unit (CPU) processor, 16.00 GB RAM memory.

Most of the test functions were drawn from the Constrained and Unconstrained Testing Environment, revisited (CUTEr) library [29], and some test functions were suggested by Andrei [30] and Moré et al. [31]. Table 1 lists 49 test functions for 100 problems with dimensions from 10 to 1000000. The parameters for the four methods are selected in the following manner.

$$\varepsilon = 10^{-6}, \rho = 0.1, \sigma = 0.01, \bar{t} = 0.3.$$

Table 1. Test problems and dimensions.

No.	Problem	Dim	No.	Problem	Dim
1	Bdexp	10000	51	Diagonal3	30
2	Bdexp	50000	52	Diagonal8	5000
3	Bdexp	100000	53	Diagonal8	10000
4	Trid	1000	54	Hager	100
5	Trid	2500	55	Extended Beale	100
6	Trid	5000	56	Penalty-I	1000
7	Dqrtic	500	57	Himmelbg1	500
8	Dqrtic	1000	58	Himmelbg2	70000
9	Dqrtic	1200	59	Himmelbg2	200000
10	Ie	1000	60	Edensch	500
11	Ie	2000	61	Edensch	600
12	Raydan1	100	62	Dqdrtic	6000
13	Raydan1	150	63	Dqdrtic	10000
14	Raydan2	1000	64	Penalty	100
15	Raydan2	5000	65	Penalty	200
16	Raydan2	10000	66	Penalty	300
17	Chebyquad	10	67	Tridia	100
18	Chebyquad	20	68	Tridia	300
19	Broyden banded	10	69	Woods	1000
20	Broyden tridiagonal	1000	70	Woods	5000
21	Broyden tridiagonal	2000	71	Woods	10000
22	Broyden tridiagonal	7000	72	Woods	100000
23	Broyden tridiagonal	10000	73	Arwhead	10000
24	Separable cubic	1000	74	Arwhead	200000
25	Separable cubic	5000	75	Dimaana	6000
26	Separable cubic	10000	76	Dimaana	9000
27	Nearly separable	500	77	Dimaanb	30000
28	Nearly separable	1000	78	Dimaanb	600000
29	Nearly separable	1000	79	Dimaanc	6000
30	Rosex	300	80	Dimaanc	24000
31	Rosex	500	81	Dimaand	9000
32	Rosex	1000	82	Dimaand	12000
33	Schittkowski	20	83	Dimaane	3300
34	Schittkowski	30	84	Dimaane	6000
35	Quartic	1000	85	Dimaanf	12000
36	Quartic	2000	86	Dimaanf	15000
37	Quartic	10000	87	Dimaang	6000
38	Dicon3dq	50	88	Nonscomp	1000
39	Dicon3dq	70	89	Nonscomp	2000
40	Cube	1000	90	Generalized Rosebrock	100

Continued on next page

No.	Problem	Dim	No.	Problem	Dim
41	Cube	10000	91	Generalized Rosebrock	200
42	Extended Tridiagonal 1	5000	92	Biggsb1	100
43	Fletchcr	5000	93	Biggsb1	150
44	Fletchcr	10000	94	Sine	20
45	Generalized Quartic	500000	95	Extended Powell singular	1000
46	Generalized Quartic	1000000	96	Cosine	1000
47	Diagonal1	10	97	Cosine	2000
48	Diagonal1	20	98	Power1	50
49	Diagonal2	2000	99	Genquqrtic	200000
50	Diagonal2	6000	100	Genquqrtic	300000

All algorithms are terminated when it satisfies the condition $\|g_k\| \leq \epsilon$ or the number of iterations exceeds 2000. Table 2 lists the numerical results of the four algorithms for 100 test problems. If the Itr exceeds 2000 and the method never reaches the optimal value, the algorithm stops and we write it as "F". The detailed numerical results are listed in the form Itr, Nf, $\|g_*\|$, and Tcpu, where it denotes the number of iterations, function evaluations, the gradient value at the end of iteration, and the CPU time, respectively.

We use the performance profile introduced by Dolan and More [32] to analyze the computational performance of all methods. Let P be the collection of n_p test problems and S be the set of solvers used in the comparison. The measure $t_{p,s}$ is defined as Itr, Nf, or Tcpu required by solver s for problem p. The definition of the performance ratio is

$$r_{p,s} = \frac{t_{p,s}}{\min\left\{t_{p,s} : s \in S \text{ and } p \in P\right\}}.$$

It is obvious that $r_{p,s} \geq 1$ for all p and s. The performance profiles for each solver s are defined by

$$P(\tau) = \frac{size\{p \in P : r_{p,s} \leq \tau\}}{n_p},$$

where $size\{p \in P : r_{p,s} \leq \tau\}$ is the number of the elements in the set $\{p \in P : r_{p,s} \leq \tau\}$, then $P(\tau)$ is the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ is within a factor τ. The efficiency of a method is represented on the horizontal axis by the percentage τ of test problems for which it is the fastest, while the vertical axis indicates the success rate $P(\tau)$ of each method in solving the test problems.

Figure 1 illustrates the iteration performance profiles of the methods. It demonstrates that most of the positions of the curves of our proposed algorithm lie above the curves of the other three methods, and our algorithm can successfully solve about 99% of the test problems. The HZ method solves about 90%, the NHS+ method solves about 87%, and the HTTHSLS method solves about 87% of the test problems, respectively. Figure 2 shows the curve of the HTTWYL method is above those of the other methods. This indicates the HTTWYL method wins the match in the function evaluations. Similarly, Figure 3 shows the HTTWYL method wins the performance profiles on CPU time, and we can see that the HTTWYL method outperforms the other three methods for the given test set. In brief, by observing the overall trend in the performance graphs, the HTTWYL algorithm performs slightly better than other algorithms. We believe that the HTTWYL algorithm is more competitive.

Table 2. Numerical test report

	HTTWYL	HZ	NHS+	HTTHSLS
No.	Itr/Nf/$\|\|g_*\|\|$/Tcpu	Itr/Nf/$\|\|g_*\|\|$/Tcpu	Itr/Nf/$\|\|g_*\|\|$/Tcpu	Itr/Nf/$\|\|g_*\|\|$/Tcpu
1	2/5/1.1e-14/0	5/22/2.0e-07/0.1562	2/5/3.4e-07/0.0625	2/5/5.9e-08/0.1875
2	2/5/2.4e-14/0.0312	5/22/2.2e-11/0.0781	2/5/2.7e-15/0.0156	2/5/1.7e-13/0.0156
3	2/5/3.4e-14/0.0781	5/22/3.0e-11/0.0312	2/5/3.7e-15/0.0312	2/5/2.4e-13/0.0469
4	86/213/4.1e-07/0.6406	123/461/7.9e-07/1.0938	77/193/6.5e-07/0.4531	85/209/7.5e-07/0.7031
5	79/211/5.1e-07/1.75	88/255/9.6e-07/2.2656	88/229/3.3e-07/2.1719	100/278/5.6e-07/2.5156
6	70/168/9.9e-07/18.3438	88/298/7.2e-07/35.5	87/220/5.0e-07/24.5781	81/191/7.0e-07/26.75
7	27/222/3.3e-07/0.0312	35/155/1.0e-06/0.0156	37/199/7.4e-07/0.0469	26/125/4.2e-07/0.0469
8	30/239/2.2e-07/0.0469	36/238/8.9e-07/0.0781	31/211/5.4e-07/0.07813	22/140/1.8e-07/0.0469
9	28/310/7.9e-07/0.0625	34/253/5.6e-07/0.0469	F/F/F/F	35/312/3.0e-07/0.0469
10	6/13/1.6e-07/1.4688	7/15/9.3e-07/1.6719	7/15/3.2e-07/1.3125	6/13/6.0e-07/1.4375
11	6/13/2.3e-07/4.1719	8/17/2.2e-08/7.5312	7/15/4.5e-07/2.8438	6/13/8.5e-07/4.7188
12	68/132/7.0e-07/0	71/143/9.1e-07/0	70/131/7.9e-07/0.0469	63/122/9.7e-07/0.0156
13	72/140/9.0e-07/0	84/166/9.7e-07/0.0156	F/F/F/F	F/F/F/F
14	6/20/3.1e-07/0	5/9/4.6e-08/0.0156	7/13/8.6e-13/0.0625	8/14/7.9e-07/0
15	6/20/6.9e-07/0.0156	5/9/8.7e-08/0.0156	7/13/1.9e-12/0.0156	F/F/F/F
16	6/20/9.7e-07/0.0312	5/9/7.5e-08/0.0312	7/13/2.7e-12/0.0156	F/F/F/F
17	134/340/6.7e-07/0.0156	157/501/8.7e-07/0.0625	114/335/9.4e-07/0.0156	131/354/7.0e-07/0.0469
18	230/774/7.7e-07/0.0781	317/1332/9.2e-07/0.0312	F/F/F/F	210/661/7.9e-07/0.0156
19	48/159/5.7e-07/0	46/218/8.4e-07/0.0312	45/376/8.4e-07/0.0156	51/381/6.5e-07/0.0312
20	70/161/5.4e-07/0	103/347/6.8e-07/0.0156	84/209/9.3e-07/0.0469	78/188/9.0e-07/0
21	81/210/7.4e-07/0.0156	88/267/6.8e-07/0.0625	71/164/6.7e-07/0	85/198/7.3e-07/0.0312
22	71/188/6.4e-07/0.0625	125/461/8.2e-07/0.1094	74/272/9.1e-07/0.0938	77/188/7.7e-07/0.0469
23	77/230/4.8e-07/0.0938	97/357/8.5e-07/0.0781	84/228/6.8e-07/0.0938	83/241/8.6e-07/0.0938
24	18/29/3.4e-07/0.0781	F/F/F/F	F/F/F/F	19/55/7.3e-07/0.0625
25	18/27/6.3e-07/0.3438	16/124/6.3e-08/1.7656	18/124/5.3e-07/1.0469	20/106/6.2e-07/1.0312
26	19/28/2.5e-07/0.5	13/96/1.5e-07/5.3906	20/104/6.6e-07/2.0625	23/84/5.0e-07/2.2656
27	234/1396/8.3e-07/0.4375	214/1396/8.4e-07/0.3281	212/1420/7.3e-07/0.3281	166/855/7.8e-07/0.2188
28	F/F/F/F	330/2698/8.0e-07/1.3594	157/1260/7.2e-07/0.5156	133/831/9.3e-07/0.5156
29	180/1235/7.2e-07/1.5	322/2615/8.3e-07/5.3281	191/2332/8.3e-07/2.906	200/1578/3.2e-07/2.5156
30	64/356/9.4e-07/0.1875	111/946/8.8e-07/0.5625	40/215/9.7e-07/0.0938	25/145/8.8e-07/0.1094
31	68/353/7.4e-07/1.5625	41/308/7.7e-07/1.5	52/305/6.8e-07/0.7969	55/341/3.7e-07/1.5469
32	90/517/4.6e-07/1.2344	77/678/8.1e-07/1.7344	68/407/7.8e-07/1.2031	20/104/7.3e-07/0.2656
33	75/541/9.8e-07/0.0156	91/923/9.3e-07/0.0156	15/90/2.0e-07/0	21/161/7.2e-07/0.0469
34	83/883/9.3e-07/0	22/299/9.7e-07/0.0156	50/619/6.9e-07/0.0938	21/260/6.1e-07/0
35	4/17/1.6e-07/0	3/13/1.2e-07/0	3/16/5.2e-08/0.0156	5/19/4.1e-07/0.0156
36	4/17/2.3e-07/0.0156	3/13/1.7e-07/0.0156	3/16/7.4e-08/0.0625	5/19/5.8e-07/0.0156
37	4/17/5.1e-07/0.0156	3/13/3.8e-07/0.0469	3/16/1.6e-07/0.0469	6/30/9.3e-11/0.0469
38	476/881/5.8e-07/0.0625	526/998/8.7e-07/0.0312	554/1078/9.4e-07/0.0156	548/1077/8.8e-07/0.0156
39	710/1329/6.7e-07/0.0312	728/1424/9.3e-07/0.0469	751/1459/9.5e-07/0.0312	782/1522/9.0e-07/0.0469
40	80/517/9.9e-07/0.0156	50/408/7.2e-07/0	46/434/9.5e-07/0.0156	64/545/8.5e-07/0.01

Continued on next page

	HTTWYL	HZ	NHS+	HTTHSLS
No.	Itr/Nf/$\|\|g_*\|\|$/Tcpu	Itr/Nf/$\|\|g_*\|\|$/Tcpu	Itr/Nf/$\|\|g_*\|\|$/Tcpu	Itr/Nf/$\|\|g_*\|\|$/Tcpu
41	80/517/9.9e-07/0	50/408/7.2e-07/0.0156	46/434/9.5e-07/0.0156	64/545/8.5e-07/0.0312
42	29/70/4.8e-08/0.0938	6/23/1.3e-07/0.0469	21/80/1.4e-07/0.1094	21/56/2.5e-07/0.0469
43	38/210/5.9e-07/0	47/403/2.7e-07/0.0625	F/F/F/F	46/239/7.1e-07/0.0156
44	69/358/1.0e-06/0.0781	59/602/9.9e-07/0.1094	79/497/1.8e-07/0.1719	50/273/2.6e-07/0.125
45	5/11/5.6e-07/0.3281	11/27/2.2e-07/0.25	8/17/1.9e-08/0.2656	9/19/6.6e-07/0.3125
46	5/11/5.3e-07/0.2188	9/23/3.5e-08/0.1562	7/15/9.4e-07/0.3438	10/21/1.9e-07/1.5
47	24/52/8.0e-07/0	21/48/8.4e-07/0	24/50/6.7e-07/0.0156	24/52/5.8e-07/0.0156
48	35/74/5.8e-07/0.0156	35/69/7.2e-07/0.0156	34/73/5.1e-07/0	33/92/4.0e-07/0.0781
49	289/833/5.0e-07/0.2031	F/F/F/F	330/1181/9.2e-07/0.125	F/F/F/F
50	492/1565/8.8e-07/0.4375	568/1835/9.1e-07/0.4062	F/F/F/F	F/F/F/F
51	47/113/8.1e-07/0.0312	48/130/6.8e-07/0.0156	47/85/9.8e-07/0	48/97/9.1e-07/0.0469
52	5/10/1.1e-07/0.046875	6/12/6.8e-07/0.0156	5/10/4.3e-08/0.0156	5/10/7.9e-07/0.0156
53	5/10/3.3e-07/0.0156	F/F/F/F	5/10/3.3e-07/0.0938	F/F/F/F
54	24/56/5.2e-07/0.0156	25/83/7.2e-07/0.0156	24/56/7.1e-07/0.0312	23/59/6.6e-07/0.015625
55	26/143/7.3e-07/0.0156	23/77/9.9e-12/0	15/61/3.4e-07/0	19/169/7.1e-08/0.0312
56	129/568/1.5e-07/0	50/266/8.7e-07/0	31/272/1.1e-07/0.0156	60/565/4.7e-07/0.0156
57	7/18/1.8e-07/0.0156	7/22/3.3e-07/0	6/16/2.3e-07/0.0156	6/16/1.3e-07/0.0156
58	9/21/3.1e-07/0.0781	7/17/3.2e-07/0.0312	6/15/2.6e-07/0.0625	7/17/7.4e-12/0.0156
59	9/21/5.2e-07/0.3594	7/17/5.4e-07/0.1562	6/15/4.4e-07/0.0938	7/17/1.2e-11/0.2969
60	24/71/4.9e-07/0.0156	F/F/F/F	24/66/7.3e-07/0.0156	F/F/F/F
61	24/50/7.1e-07/0.0312	25/74/5.2e-07/0.0312	F/F/F/F	F/F/F/F
62	77/287/7.2e-07/0.0312	47/255/9.3e-07/0.0625	35/157/5.2e-07/0.0156	60/224/9.3e-07/0.0312
63	61/215/9.2e-07/0.0781	71/395/8.1e-07/0.1719	39/158/7.1e-07/0.0469	50/184/7.6e-07/0.0469
64	11/13/8.8e-07/0.0469	7/108/9.1e-10/0.0156	5/10/1.2e-10/0.0156	11/13/4.4e-07/0.0312
65	12/18/1.9e-07/0.0312	6/15/1.1e-10/0.03125	7/18/1.8e-08/0.0469	14/19/6.7e-07/0.0312
66	13/22/3.2e-07/0.0312	5/35/9.2e-10/0.01	8/21/6.2e-10/0.0156	F/F/F/F
67	331/1447/9.7e-07/0.0156	420/2750/8.8e-07/0.0469	408/2028/5.0e-07/0.0156	403/1809/9.9e-07/0.0156
68	723/4053/7.5e-07/0.0156	711/7762/8.0e-07/0.0469	774/4983/9.3e-07/0.0312	753/3856/9.3e-07/0.0469
69	134/611/9.5e-07/0.0156	F/F/F/F	126/827/7.5e-07/0.0312	147/720/8.7e-07/0.0156
70	144/748/7.6e-07/0.0625	F/F/F/F	177/906/8.0e-07/0.1094	163/788/6.7e-07/0.0781
71	179/799/6.7e-07/0.1094	F/F/F/F	226/1543/7.2e-07/0.0938	201/890/6.9e-07/0.0938
72	195/1020/6.0e-07/1.8281	F/F/F/F	F/F/F/F	158/829/6.0e-07/2
73	7/24/4.9e-07/0.09375	F/F/F/F	7/24/2.9e-08/0.0156	F/F/F/F
74	7/24/2.8e-07/0.1406	F/F/F/F	7/24/1.3e-07/0.1562	F/F/F/F
75	10/20/2.4e-07/0.0938	10/20/1.2e-07/0.0312	9/19/2.0e-08/0.1719	21/26/5.4e-07/0.0156
76	10/20/2.9e-07/0.1719	10/20/1.5e-07/0.0625	9/19/2.4e-08/0.0156	21/26/6.6e-07/0.0156
77	22/28/4.4e-07/0.1406	9/20/9.7e-07/0.0938	9/25/5.4e-07/0.2969	30/33/7.8e-07/0.1094
78	23/29/8.1e-07/3.1406	11/23/4.4e-08/2.7344	12/29/2.9e-07/2.9844	32/35/8.3e-07/4.7812

Continued on next page

No.	HTTWYL Itr/Nf/$\|\|g_*\|\|$/Tcpu	HZ Itr/Nf/$\|\|g_*\|\|$/Tcpu	NHS+ Itr/Nf/$\|\|g_*\|\|$/Tcpu	HTTHSLS Itr/Nf/$\|\|g_*\|\|$/Tcpu
79	18/64/7.4e-07/0.0312	11/26/7.7e-07/0.0156	15/30/6.0e-08/0.1094	19/34/4.7e-07/0.0469
80	23/149/1.3e-07/0.6094	13/29/1.2e-08/0.1875	17/29/9.7e-07/0.2969	25/36/5.6e-07/0.4219
81	30/93/9.6e-07/0.1562	12/30/4.7e-07/0.0781	14/37/8.4e-09/0.2188	45/99/7.7e-07/0.25
82	21/62/7.4e-07/0.1406	12/30/5.6e-07/0.0469	13/31/9.7e-07/0.0781	31/69/6.5e-07/0.1562
83	277/823/9.9e-07/0.0312	266/799/9.9e-07/0.25	286/846/9.4e-07/0.4219	292/866/9.6e-07/0.2969
84	366/1090/9.7e-07/1.25	351/1054/9.8e-07/0.8438	377/1119/9.7e-07/0.6875	385/1145/9.7e-07/1
85	434/1294/9.8e-07/2.0312	399/1194/9.9e-07/1.625	424/1258/9.9e-07/1.5625	442/1316/9.9e-07/1.4375
86	477/1423/9.9e-07/2.2812	440/1317/9.9e-07/3.3906	466/1386/9.9e-07/6.0156	487/1451/9.9e-07/3.6562
87	588/2224/9.5e-07/1.3125	259/790/9.7e-07/0.375	299/924/9.7e-07/0.5	480/1534/9.2e-07/0.4531
88	45/147/9.4e-07/0.0156	45/151/3.7e-07/0	48/236/8.5e-07/0.0156	61/165/3.4e-07/0
89	65/303/4.0e-07/0	74/357/6.2e-07/0.0156	69/283/9.3e-07/0	239/630/4.0e-07/0.0156
90	576/2328/8.6e-07/0.0312	732/4581/9.3e-07/0.0469	F/F/F/F	450/1661/7.7e-07/0
91	970/3649/8.2e-07/0.0312	1025/3968/9.6e-07/0.0312	F/F/F/F	F/F/F/F
92	602/1097/9.7e-07/0.0469	649/1272/7.8e-07/0.0312	684/1317/8.0e-07/0.0156	772/1489/9.7e-07/0.0156
93	682/1340/9.4e-07/0.0312	730/1429/8.0e-07/0.0469	1017/1977/9.4e-07/0.0312	986/1929/8.7e-07/0.0156
94	20/69/3.3e-07/0	16/68/5.5e-07/0.0625	F/F/F/F	13/62/4.2e-07/0
95	206/679/7.5e-07/0.0625	329/1349/9.9e-07/0.1875	237/770/9.0e-07/0.0781	113/337/8.5e-07/0.0781
96	9/20/8.0e-07/0	11/24/5.3e-07/0	F/F/F/F	F/F/F/F
97	9/20/9.9e-07/0.0156	11/28/7.9e-07/0.0156	F/F/F/F	9/20/9.3e-07/0.0156
98	509/2480/6.1e-07/0.0469	625/5133/8.6e-07/0.0156	541/2903/7.3e-07/0.0312	712/4746/7.9e-07/0.0312
99	17/104/5.1e-08/0.5938	13/37/7.4e-07/0.1406	20/109/1.9e-07/0.2656	25/76/3.8e-07/0.7188
100	21/95/1.9e-07/0.2344	12/35/2.0e-07/0.4219	22/96/3.4e-07/0.2812	22/119/5.9e-08/0.5

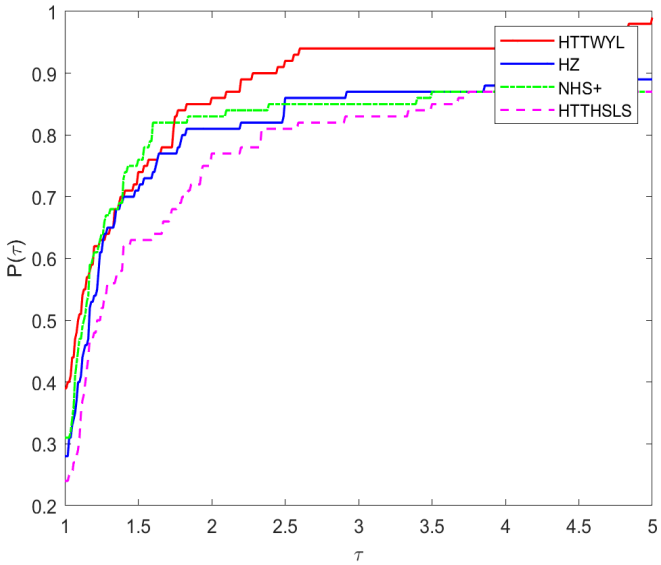

Figure 1. Performance profiles on Itr.

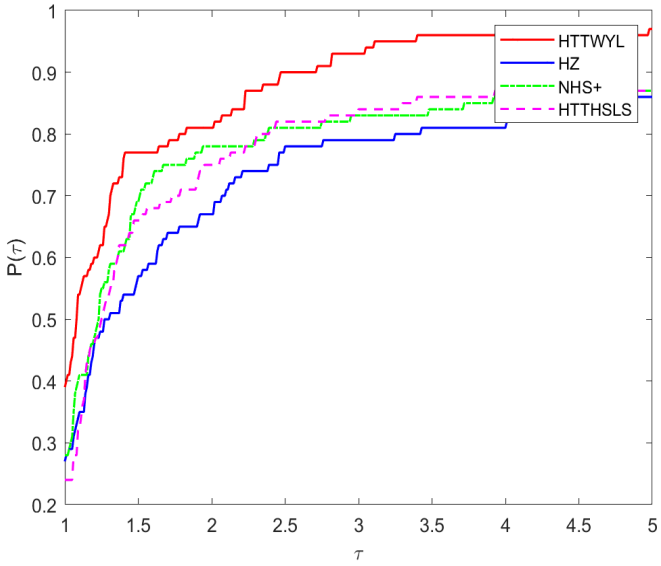

Figure 2. Performance profiles on Nf.

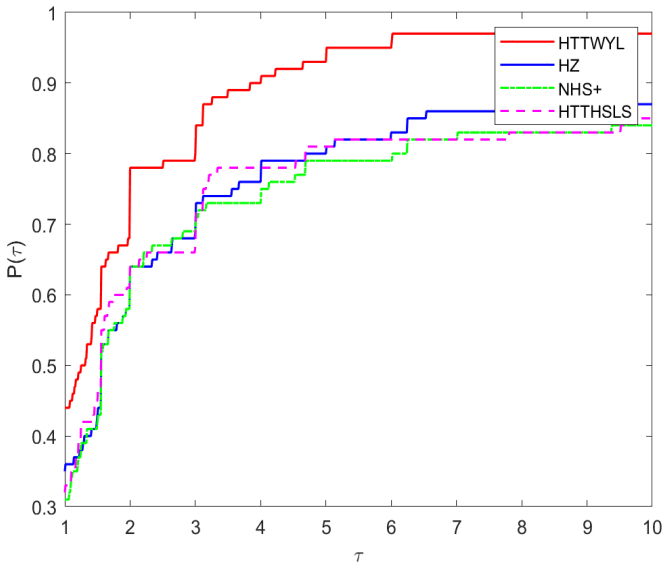

Figure 3. Performance profiles on Tcpu.

4. Application in image restoration

In this section, we use the HTTWYL algorithm to solve image restoration problems. Cai et al. [33] used the two-phase scheme to clean salt-and-pepper noise. The first phase is to detect noisy pixels by using the adaptive median filter [34]. The second phase involves clearing noisy pixels by solving the following smoothing minimization problem.

$$\min_{\mathbf{u}} F_\chi(\mathbf{u}) := \sum_{(i,j)\in\mathcal{N}} \left(2 \sum_{(m,n)\in\mathcal{V}_{i,j}\setminus\mathcal{N}} \varphi_\chi(u_{i,j} - y_{m,n}) + \sum_{(m,n)\in\mathcal{V}_{i,j}\cap\mathcal{N}} \varphi_\chi(u_{i,j} - u_{m,n}) \right). \tag{4.1}$$

Here, $\varphi_\chi(t) = \sqrt{t^2+\chi}$ is an edge-preserving function with parameter $\alpha > 0$. $\mathcal{N} \subset \mathcal{A}$ is the index set of noise pixels detected in the first phase, $\mathcal{A} = \{1,\ldots,M\} \times \{1,\ldots,N\}$ is the pixel pointer set of the original image of size $M \times N$, $\mathbf{u} = [u_{i,j}]_{(i,j)\in\mathcal{N}}$ is a column vector of length $|\mathcal{N}|$ ordered lexicographically, i.e., the number of elements in $\mathcal{N}$, and $y_{i,j}$ is the pixel value of the image at the point (i, j).

Now, our attention shifts to employing the two-phase strategy to eliminate salt-and-pepper noise, which is a specific instance of impulse noise. In the first phase, we employ an adaptive median filter [34] to identify noisy pixels. Obviously, the higher the noise ratio, the larger the scale of (4.1). Cai [33] discovered that the contaminated images can be restored efficiently by using the conjugate gradient methods to solve the above problem (4.1). Thus, in the second phase, we utilize the HTTWYL method to solve (4.1) and continue to compare it with the HZ, NHS+, and HTTHSLS methods. These methods all use the Wolfe line search (1.2) and (1.3) to compute the step-length α_k. Moreover, the related parameters for these methods are set as follows: $\rho = 0.1,\ \sigma = 0.01, \bar{t} = 0.3$.

The test images are Peppers(512×512), Hill(512×512), Man(512×512), and Boat(512×512). The stopping criterion for the involved methods is

$$\text{Itr} > 300 \text{ or } \frac{\left|F_\chi(u_k) - F_\chi(u_{k-1})\right|}{F_\chi(u_k)} \leq 10^{-4}. \tag{4.2}$$

In order to evaluate the restoration performance clearly, we use the peak signal to noise ratio (PSNR; see [35]) defined by

$$\text{PSNR} = 10\log_{10}\frac{255^2}{\frac{1}{MN}\sum_{i,j}\left(x^r_{i,j} - x^*_{i,j}\right)^2}, \tag{4.3}$$

where $x^r_{i,j}$ and $x^*_{i,j}$ represent the pixel values of the restored image and the original one, respectively.

We plot the original, noisy, and restored images for three algorithms when the salt-and-pepper noise ratio is 70% and 90%. For the corresponding results, see Figures 4 and 5. In Table 3, we report the number of iterations (Itr), the CPU time (Tcpu), and the PSNR values. From Table 3, we can intuitively perceive that the HTTWYL method usually has higher PSNR among the three methods under the same noise ratio, and it takes fewer iterations and CPU time than the other three methods in most cases. Thus, our algorithm seems to be capable to reconstruct the noisy pictures with more quality.

Table 3. Numerical results for image restoration problems.

Image	Noise ratio	HTTWYL Itr/PSNR/Tcpu	HZ Itr/PSNR/Tcpu	NHS+ Itr/PSNR/Tcpu	HTTHSLS Itr/PSNR/Tcpu
Peppers	30%	17/33.06/0.82	18/32.80/0.74	18/33.04/0.75	18/33.06/0.81
Peppers	50%	20/30.35/1.39	25/30.34/1.49	22/30.36/1.78	18/30.36/1.47
Peppers	70%	23/27.28/1.78	23/26.98/1.88	23/27.28/1.85	16/27.16/1.56
Peppers	90%	34/22.61/2.57	26/20.65/2.25	33/22.60/2.59	32/22.51/2.59
Hill	30%	17/34.97/3.12	16/34.95/4.38	15.34.94/7.01	14/34.94/6.93
Hill	50%	18/32.62/6.00	20/32.58/6.05	19/32.62/5.96	16/32.59/6.01
Hill	70%	18/29.64/7.62	29/29.77/20.68	20/29.72/16.72	19/29.67/16.15
Hill	90%	30/25.58/15.52	36/25.33/29.73	33/25.63/25.13	24/25.28/25.13
Man	30%	17/31.53/3.03	17/31.47/3.84	11/31.52/6.29	16/31.53/7.26
Man	50%	13/29.09/4.85	27/29.07/13.25	16/29.08/11.77	18/29.10/12.64
Man	70%	19/26.21/8.56	32/26.25/23.80	23/26.27/18.50	22/26.25/17.87
Man	90%	36/22.52/16.14	44/22.42/17.65	37/22.54/13.49	28/22.43/22.36
Boat	30%	16/33.67/3.09	17/33.62/4.43	16/33.67/7.20	13/33.61/6.80
Boat	50%	17/31.10/5.60	22/31.10/7.84	16/31.09/12.06	18/31.09/12.22
Boat	70%	22/28.24/8.74	26/28.26/13.16	22/28.24/7.84	21/28.23/10.33
Boat	90%	31/24.12/16.70	27/23.57/24.79	33/24.13/29.47	33/24.13/18.26

5. Application in machine learning problem

In this section, we test the numerical performance of the HTTWYL in solving the machine learning problem. We embed the HTTWYL algorithm into the stochastic recursive gradient algorithm (SARAH) [36] (denote as HTTWYL_RAH). We use the Wolfe line search criterion to calculate the learning rate (step size). In addition, we compare the HTTWYL_RAH method with the stochastic variance reduced gradient (SVRG) [37], SARAH, and stochastic gradient descent (SGD) [38] methods. All the experiments tested the following learning model:

Ridge regression (ridge)

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - x_i^T \boldsymbol{w} \right)^2 + \lambda \|\boldsymbol{w}\|_2^2, \tag{5.1}$$

where $y_i \in \{-1, +1\}$ is the target value of the ith sample and $x_i \in R^d$ is a feature vector of the ith sample. $\lambda > 0$ is a regularization parameter. The related parameters for these methods are set as follows: $\rho = 0.1$, $\sigma = 0.01, \bar{t} = 0.3$. All algorithms are executed on four large-scale datasets, including a8a, a9a, ijcnn1, and w8a. These datasets come from the LIBSVM Data website [*] , and the specific details of the datasets are provided in Table 4.

[*]All datasets are available at https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/.

Figure 4. The original images (first row), the noisy images with 70% salt-and-paper noise (second row) and the restored images by HTTWYL (third row), HZ (fourth row), NHS+ (fifth row), and HTTHSLS (last row).

Figure 5. The original images (first row), the noisy images with 90% salt-and-paper noise (second row) and the restored images by HTTWYL (third row), HZ (fourth row), NHS+ (fifth row), and HTTHSLS (last row).

Table 4. Description of datasets.

Data set	Sample size(n)	Dimension
a8a	9865	122
a9a	16281	122
ijcnn1	91701	22
w8a	49749	301

The vertical axis represents the function value of the loss value and the horizontal axis represents the number of internal iterations. The parameters in SARAH, SVRG, SGD are the same as their original papers. In this experiment, the maximum number of inner iterations is 30. Figure 6 represents the convergence of the loss function ($\lambda = 10^{-2}$) for six algorithms on four datasets.

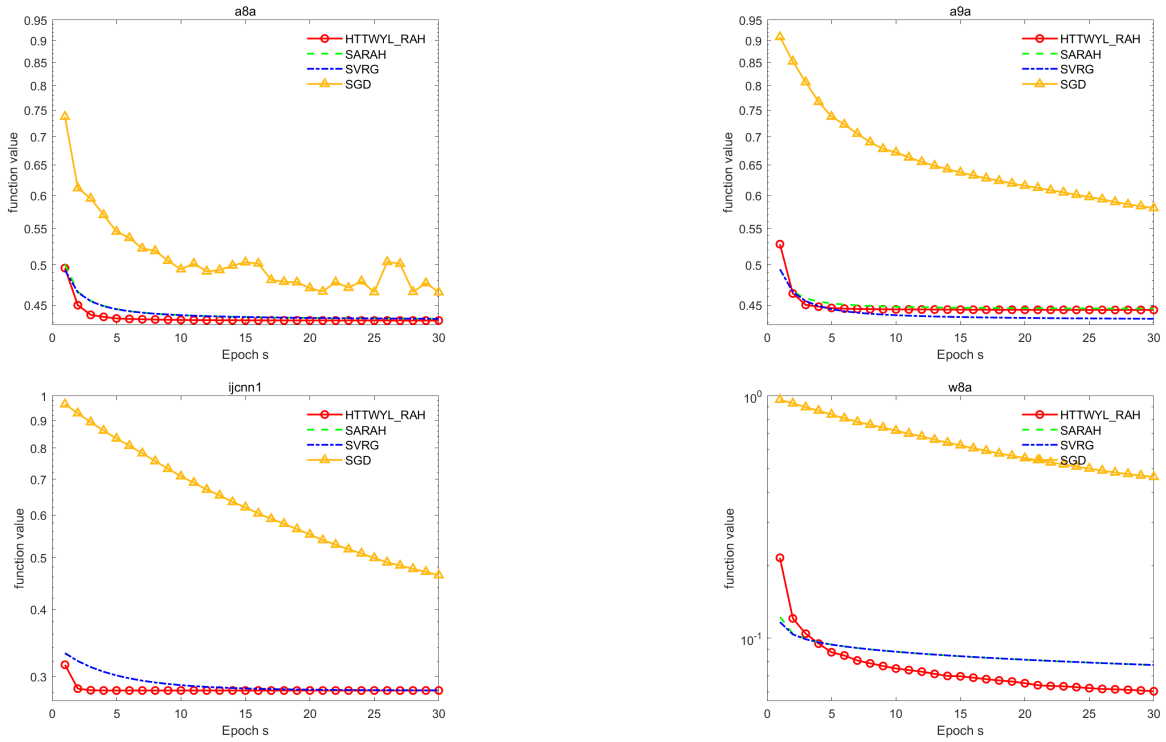

Figure 6. Comparison of the HTTWYL_RAH with other stochastic methods on four datasets.

In general, a lower curve of the algorithm indicates better convergence behavior. As can be seen in Figure 6, the HTTWYL_RAH method has the fastest convergence in most cases.

6. Conclusions

We present a hybrid three-term conjugate gradient method HTTWYL for solving unconstrained optimization problems, which combines features from WYL and other classical conjugate gradient methods. The global convergence of the method is established under certain conditions. We use it to deal with the unconstrained optimization test problems and apply it to solving image restoration problems and ridge regression in machine learning problems. The numerical results indicate that

compared to other methods, our proposed method is more effective and promising. Finally, we hope that the contributions in this paper will continue to be explored for applications in other areas.

Author contributions

Zhang and Yang: Writing-original draft and editing. All authors have read and approved the final version of the manuscript for publication.

Acknowledgments

This research is funded by the Science and Technology Development Planning Grant Program of Jilin Province (YDZJ202101ZYTS167, 20200403193SF) , and the graduate innovation project of Beihua University (2023035, 2022038).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. H. Adeli, S. L. Hung, An adaptive conjugate gradient learning algorithm for efficient training of neural networks, *Appl. Math. Comput.*, **62** (1994), 82–102. https://doi.org/10.1016/0096-3003(94)90134-1
2. S. Rong, W. X. Li, Z. M. Li, Y. Sun, T. Y. Zheng, Optimal allocation of Thermal-Electric decoupling systems based on the national economy by an improved conjugate gradient method, *Energies.*, **9** (2015), 17. https://doi.org/10.3390/en9010017
3. C. Decker, D. M. Falcao, E. Kaszkurewicz, Conjugate gradient methods for power system dynamic simulation on parallel computers, *EEE Trans. Power Syst.*, **11** (1996), 1218–1227. https://doi.org/10.1109/59.535593
4. P. Vivek, B. Addisu, M. G. A. Sayeed, J. K. Nand, K. S. Gyanendra, An application of conjugate gradient technique for determination of thermal conductivity as an inverse engineering problems, *Mater. Today Proc.*, **47** (2021), 3082–3087. https://doi.org/10.1016/j.matpr.2021.06.073
5. Z. C. Liu, S. P. Zhu, Y. Ge, F. Shan, L. P. Zeng, W. Liu, Geometry optimization of two-stage thermoelectric generators using simplified conjugate-gradient method, *Appl. Energ.*, **190** (2017), 540–552. https://doi.org/10.1016/j.apenergy.2017.01.002
6. R. Fletcher, C. Reeves, Function minimization by conjugate gradierent, *Comput. J.*, **7** (1964), 149–154. https://doi.org/10.1093/comjnl/7.2.149
7. E. Polak, G. Ribière, Note sur la convergence methodes de direction conjugées, *Math. Model. Numer. Anal.*, **3** (1969), 35–43. https://doi.org/10.1051/M2AN/196903R100351
8. B. T. Polyak, The conjugate gradient method in extreme problems, *USSR Comput. Math. Math. Phys.*, **9** (1969), 94–112. https://doi.org/10.1016/0041-5553(69)90035-4

9. M. R. Hestenes, E. Stiefel, Method of conjugate gradient for solving linear equations, *Res. Nat. Bur. Stand.*, **49** (1952), 409–436.
10. Y. Dai, Y. Yuan. A nonlinear conjugate gradient with a strong global convergence properties, *SIAM. Optim.*, **10** (2000), 177–182. https://doi.org/10.1137/S1052623497318992
11. Y. Liu, C. Storey. Efficient generalized conjugate gradient algorithms, part 1: Theory, *J. Optim. Theory Appl.*, **69** (1991), 129–137. https://doi.org/10.1007/BF00940464
12. L. Zhang, An improved Wei-Yao-Liu nonlinear conjugate gradient method for optimization computation, *Appl. Math. Comput.*, **215** (2009), 2269–2274. https://doi.org/10.1016/j.amc.2009.08.016
13. H. Huang, S. H. Lin, A modified Wei-Yao-Liu conjugate gradient method for unconstrained optimization, *Appl. Math. Comput.*, **231** (2014), 179–186. https://doi.org/10.1016/j.amc.2014.01.012
14. Y. P. Hu, Z. X. Wei, Wei-Yao-Liu conjugate gradient projection algorithm for nonlinear monotone equations with convex constraints, *J. Comput. Math.*, **92** (2014), 2261–2272. https://doi.org/10.1080/00207160.2014.977879
15. M. Li, A modified Hestense-Stiefel conjugate gradient method close to the memoryless BFGS quasi-Newton method, *Optim. Methods Softw.*, **22** (2018), 336–353. https://doi.org/10.1080/10556788.2017.1325885
16. S. W. Yao, B. Qin, A hybrid of DL and WYL nonlinear conjugate gradient methods, *Abstr. Appl. Anal.*, 2014, 279891. http://dx.doi.org/10.1155/2014/279891
17. P. Kumam, A. B. Abubakar, M. Malik, A. H. Ibrahim, A hybrid HS-LS conjugate gradient algorithm for unconstrained optimization with applications in motion control and image recovery, *J. Comput. Appl. Math.*, **433** (2023), 115304. https://doi.org/10.1016/j.cam.2023.115304
18. Z. X. Wei, S. W. Yao, L. Y. Liu, The convergence properties of some new conjugate gradient methods, *Appl. Math. Comput.*, **183** (2006), 1341–1350. https://doi.org/10.1016/j.amc.2006.05.150
19. H. Huang, Z. X. Wei, S. W. Yao, The proof of the sufficient descent condition of the Wei-Yao-Liu conjugate gradient method under the strong Wolfe-Powell line search, *Appl. Math. Comput.*, **189** (2007), 1241–1245. https://doi.org/10.1016/j.amc.2006.12.006
20. J. Nocedal, Updating quasi-Newton matrices with limited storage, *Math. Comp.*, **35** (1980), 773–782. https://doi.org/10.1090/S0025-5718-1980-0572855-7
21. D. Luo, Y. Li, J. Y. Lu, G. L. Yuan, A conjugate gradient algorithm based on double parameter scaled Broyden-Fletcher-Goldfarb-Shanno update for optimization problems and image restoration, *Neural Comput. Appl.*, **34** (2022), 535–553. https://doi.org/10.1007/s00521-021-06383-y
22. X. Z. Jiang, X. M. Ye, Z. F. Huang, M. X. Liu, A family of hybrid conjugate gradient method with restart procedure for unconstrained optimizations and image restorations, *Comput. Oper. Res.*, **159** (2023), 106341. https://doi.org/10.1016/j.cor.2023.106341
23. X. Z. Jiang, L. G. Pan, M. X. Liu, J. B. Jian, A family of spectral conjugate gradient methods with strong convergence and its applications in image restoration and machine learning, *J. Franklin Inst.*, **361** (2024), 107033. https://doi.org/10.1016/j.jfranklin.2024.107033

24. G. L. Yuan, T. T. Li, W. J. Hu, A conjugate gradient algorithm for large scale nonlinear equations and image restoration problems, *Appl. Numer. Math.*, **147** (2020), 129–141. https://doi.org/10.1016/j.apnum.2019.08.022
25. G. L. Yuan, J. Y. Lu, Z. Wang, The PRP conjugate gradient algorithm with a modified WWP line search and its application in the image restoration problems, *Appl. Numer. Math.*, **152** (2020), 1–11. https://doi.org/10.1016/j.apnum.2020.01.019
26. J. H. Yin, J. B. Jian, X. Z. Jiang, A generalized hybrid CGPM-based algorithm for solving large-scale convex constrained equations with applications to image restoration, *J. Comput. Appl. Math.*, **391** (2021), 113–423. https://doi.org/10.1016/j.cam.2021.113423
27. J. Y. Cao, J. Z. Wu, A conjugate gradient algorithm and its applications in image restoration, *Appl. Math. Comput.*, **183** (2020), 243–252. https://doi.org/10.1016/j.apnum.2019.12.002
28. W. W. Hager, H. C. Zhang, A survey of nonlinear conjugate gradient methods, *Pacific J. Optim.*, **2** (2006), 35–58. https://doi.org/10.1006/jsco.1995.1040
29. N. I. M. Gould, D. Orban, P. L. Toint, CUTEr and SifDec: A constrained and unconstrained testing environment, *ACM Trans. Math. Soft.*, **29** (2003), 373–394. https://doi.org/10.1145/962437.962439
30. N. Andrei, An unconstrained optimization test functions collection, *Adv. Model. Optim.*, **10** (2008), 147–161.
31. J. J. Moré, B. S. Garbow, K. E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Software.*, **7** (1981), 17–41. https://doi.org/10.1145/355934.35593
32. E. D. Dolan, J. J. More, Benchmarking optimization software with performance profiles, *Math. Program.*, **91** (2002), 201–213. https://doi.org/10.1007/s101070100263
33. J. F. Cai, R. Chan, B. Morini, Minimization of an edge-preserving regularization functional by conjugate gradient type methods, *Image Proc. Based Part. Diff. Equat.*, 2007, 109–122. https://doi.org/10.1007/978-3-540-33267-1
34. H. Hwang, R. A. Haddad, Adaptive median filters: New algorithms and results, *IEEE Trans. Image Process.*, **4** (1995), 499–502. https://doi.org/10.1109/83.370679
35. A. Bovik, Handbook of image and video processing, Academic Press, San Diego., 2005, https://doi.org/10.1016/B978-0-12-119792-6.X5062-1
36. L. M. Nguyen, J. Liu, K. Scheinberg, M. Takáč, SARAH: A novel method for machine learning problems using stochastic recursive gradient, *Int. Conf. Mach. Lear.*, 2017, 2613–2621. https://doi.org/10.48550/arXiv.1703.00102
37. R. Johnson, T. Zhang, Accelerating stochastic gradient descent using predictive variance reduction, *Adv. Neural Inf. Process. Syst.*, 2013, 315–323.
38. L. Bottou, Large-scale machine learning with stochastic gradient descent, *Proc. COMPSTAT.*, **12** (2010), 177–186. https://doi.org/10.1007/978-3-7908-2604-3_16

AIMS Press