Yuan et al. *Journal of Inequalities and Applications* (2019) 2019:247
https://doi.org/10.1186/s13660-019-2192-6

Journal of Inequalities and Applications
a SpringerOpen Journal

RESEARCH

Open Access

A conjugate gradient algorithm and its application in large-scale optimization problems and image restoration

Gonglin Yuan[1], Tingting Li[1] and Wujie Hu[1*]

*Correspondence:
hwj@st.gxu.edu.cn
[1]College of Mathematics and Information Science, Guangxi University, Nanning, P.R. China

Abstract

To solve large-scale unconstrained optimization problems, a modified PRP conjugate gradient algorithm is proposed and is found to be interesting because it combines the steepest descent algorithm with the conjugate gradient method and successfully fully utilizes their excellent properties. For smooth functions, the objective algorithm sufficiently utilizes information about the gradient function and the previous direction to determine the next search direction. For nonsmooth functions, a Moreau–Yosida regularization is introduced into the proposed algorithm, which simplifies the process in addressing complex problems. The proposed algorithm has the following characteristics: (i) a sufficient descent feature as well as a trust region trait; (ii) the ability to achieve global convergence; (iii) numerical results for large-scale smooth/nonsmooth functions prove that the proposed algorithm is outstanding compared to other similar optimization methods; (iv) image restoration problems are done to turn out that the given algorithm is successful.

MSC: 90C26

Keywords: Conjugate gradient; Nonconvex and nonsmooth; Descent property; Global convergence

1 Introduction

The concerned problem is given by

$$\min\{f(x) \mid x \in \Re^n\}, \tag{1.1}$$

where the function $f : \Re^n \to \Re$ and $f \in C^2$. The above model is quite typical but a difficult mathematic model and is seen throughout daily life, work, and scientific research, thus being the focus of a great variety of careers. Experts and scholars have conducted numerous in-depth studies and achieved a series of fruitful results (see, e.g., [2, 5, 12, 22, 29–31, 40, 50, 52, 53]). It is quite noticeable that the steepest descent method is simple, and the computational and memory requirements are low. In the negative gradient direction, the function's value decreases rapidly, which makes it easy to think that this is a suitable search direction, although the convergence rate of the gradient method is not always fast. Later, experts and scholars modified this method and presented an efficient conjugate gradient method, which provides a simple form but high performance.

There are two aspects of optimization problems: the step length and the search direction. In general, the mathematical formula for (1.1) is

$$x_{k+1} = x_k + \alpha_k d_k, \quad k \in \{0, 1, 2, \dots\}, \tag{1.2}$$

where x_k is the current iteration point, α_k is called the step length, and d_k is the kth search direction. The formula for d_k is often defined by

$$d_{k+1} = \begin{cases} -g_{k+1} + \beta_k d_k, & \text{if } k \geq 1, \\ -g_{k+1}, & \text{if } k = 0, \end{cases} \tag{1.3}$$

where $\beta_k \in \Re$. In addition, increasingly more efficient and successful conjugate gradient algorithms have been proposed using a variety of expression for β_k as well as d_k (see, e.g., [9, 11, 27, 38, 42–44, 47, 49]). The well-known PRP algorithm [26, 27] is of the following form:

$$\beta_k^{\mathrm{PRP}} = \frac{g_{k+1}^T (g_{k+1} - g_k)}{\|g_{(x_k)}\|^2}, \tag{1.4}$$

where g_k, g_{k+1} and f_k denote $g(x_k)$, $g(x_{k+1})$ and $f(x_k)$, respectively. $g(x_{k+1}) = g_{k+1} = \nabla f(x_{k+1})$ is the gradient function of the objective function f at x_{k+1}. It is remarkable that the PRP conjugate algorithm is extremely effective for large-scale optimization problems. It is regrettable that it fails to achieve global convergence when addressing nonconvex function problems under the so-called weak Wolfe–Powell (WWP) line search technique. Its formula is as follows:

$$g(x_k + \alpha_k d_k)^T d_k \geq \rho g_k^T d_k \tag{1.5}$$

and

$$f(x_k + \alpha_k d_k) \leq f_k + \varphi \alpha_k g_k^T d_k, \tag{1.6}$$

where $\varphi \in (0, 1/2)$, $\alpha_k > 0$ and $\rho \in (\varphi, 1)$. To address the above exchanging problem, Yuan, Wei, and Lu [48] developed the following innovative formula for the normal WWP line search technique (called Yuan, Wei, and Lu line search (YWL)) and obtained numerous rich theoretical results:

$$f(x_k + \alpha_k d_k) \leq f_k + \varphi \alpha_k g_k^T d_k + \alpha_k \min\left[-\varphi_1 g_k^T d_k, \delta \frac{\alpha_k}{2} \|d_k\|^2\right] \tag{1.7}$$

and

$$g(x_k + \alpha_k d_k)^T d_k \geq \rho g_k^T d_k + \min\left[-\varphi_1 g_k^T d_k, \delta \alpha_k \|d_k\| \|d_k\|\right], \tag{1.8}$$

where $\varphi \in (0, 1/2)$, $\rho \in (\varphi, 1)$ and $\varphi_1 \in (0, \varphi)$. Further study can be found in [45]. Based on the innovation of the YWL line search technique, Yuan et al. [41] focused on the usual

Armijo line search technique and proposed a modified Armijo line search technique as follows:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \lambda \alpha_k g_k^T d_k + \alpha_k \min\left[-\lambda_1 g_k^T d_k, \lambda \frac{\alpha_k}{2}\|d_k\|^2\right], \tag{1.9}$$

where $\lambda, \gamma \in (0,1)$, $\lambda_1 \in (0,\lambda)$, and α_k is the largest number of $\{\gamma^k \mid k = 0,1,2,\ldots\}$. It is interesting that some scholars not only focus on the expression of the coefficient β_k but also attempt to modify the formula of the search direction d_{k+1}. Nonlinear conjugate gradient methods are increasingly more interesting to scholars because of their simplicity and lower equipment requirements for the calculation environment. Thus, HS (see [13, 16, 33]) and PRP algorithms (see [35, 51]) are widely used to solve complex problems in various fields. Currently, some experts focus on the three-term conjugate gradient because its search direction sometimes has the descent and automatic trust region properties. Motivated by the above discussion, a new modified three-term conjugate gradient algorithm based on the modified Armijo line search technique is proposed. The algorithm has the following properties:

- The search direction has a sufficient decrease and a trust region property.
- For general functions, the proposed algorithm under mild assumptions possesses global convergence.
- The new algorithm combines the deepest descent method with the conjugate gradient algorithm through the size of the coefficients, and the numerical results demonstrate the method's good performance compared with established algorithms.
- The corresponding numerical results prove that the discussed method is efficient as well as successful at solving general problems.
- The paper successfully combines the mathematic theory with real-world application. On the one hand, the proposed algorithm has a good performance in solving the large-scale optimization problems, on the other hand, it is introduced in the image restoration, which has wild application in biological engineering, medical sciences and other areas of science and engineering.

The remainder of this paper is organized as follows: The next section presents the motivation and the content of the algorithm to solve large-scale smooth problems includes the important mathematical characters; the similar optimization algorithm was presented to solve large-scale non-smooth optimization problems; the Sect. 4 presents the application of the Sect. 3 in the problem of the image restoration; the paper's conclusion and algorithm's characters was listed in Sect. 5. Without loss of generality, $f(x_k)$ and $f(x_{k+1})$ are replaced by f_k and f_{k+1}, and $\|\cdot\|$ is the Euclidean norm.

2 New three-term conjugate gradient algorithm for smooth problems

The three-term conjugate gradient algorithm has seen extensive study and obtained extremely good theoretical results. In the light of the work by Toouati-Ahmed, Storey [34], Al-Baali [1], Gilbert, and Nocedal [17] on conjugate gradient methods, the sufficient descent condition is crucial for the global convergence. From this, a famous formula for the search direction d_{k+1} emerges. Zhang [51] proposed the following formula:

$$d_{k+1} = \begin{cases} -g_{k+1} + \frac{g_{k+1}^T y_k d_k - d_k^T g_{k+1} y_k}{g_k^T g_k} & \text{if } k \geq 1, \\ -g_{k+1}, & \text{if } k = 0, \end{cases} \tag{2.1}$$

where $y_k = g_{k+1} - g_k$. It is notable that the three-term conjugate gradient algorithm was firstly introduced in solving optimization problems and the numeral results proves it is competitive than similar methods, thus this paper choose it as the compared algorithm in Sects. 2.3 and 3.2. In [23], Nazareth proposed another variety of formula,

$$d_{k+1} = -y_k + \frac{y_k^T y_k}{y_k^T d_k} d_k + \frac{y_{k-1}^T y_k}{y_{k-1}^T d_{k-1}} d_{k-1}, \tag{2.2}$$

where $y_k = g_{k+1} - g_k$, g_k is the gradient function value at the point x_k, and $d_0 = d_{-1} = 0$. In [14], Deng and Zhong expressed a new three-term conjugate gradient formula as follows:

$$d_{k+1} = -g_{k+1} - \left(\left(1 - \frac{y_k^T y_k}{y_k^T s_k} \right) \frac{s_k^T g_{k+1}}{y_k^T s_k} - \frac{y_k^T g_{k+1}}{y_k^T s_k} \right) s_k - \frac{s_k^T g_{k+1}}{y_k^T s_k} y_k, \tag{2.3}$$

where $s_k = x_{k+1} - x_k$. Based on the above discussion, we express the new three-term algorithm under the modified Armijo line search technique (1.9) as follows:

$$d_{k+1} = \begin{cases} -g_{k+1} + \frac{g_{k+1}^T y_k^* d_k - d_k^T g_{k+1} y_k^*}{\max\{\xi_2 \|d_k^T\| \|y_k^*\|, \min(\xi_3 \|g_k\|^2, \xi_4 d_k^T d_k)\}} & \text{if } k \geq 1, \\ -g_{k+1} & \text{if } k = 0, \end{cases} \tag{2.4}$$

where $\xi_2, \xi_3, \xi_4 > 0$. To gather more information about the objective function, we address the corresponding gradient function as well as the initial point, let $y_k^* = y_k + \varphi_k (x_{k+1} - x_k)$, where $y_k = g_{k+1} - g_k$, $\varphi_k = \max\{0, B_k\}$, and $B_k = \frac{(g_{k+1}+g_k)^T s_k + 2(f_k - f_{k+1})}{\|x_{k+1}-x_k\|^2}$. This plays an important role in theory and numerical performance [46]. From the above discussion, we introduce a new PRP algorithm (Algorithm 2.1).

2.1 Algorithm steps

Algorithm 2.1

Step 1: (Initiation) Choose an initial point x_0, $\gamma \in (0,1)$, ξ_2, ξ_3, $\xi_4 > 0$, and positive constants $\varepsilon \in (0,1)$. Let $k = 0$, $d_0 = -g_0$.

Step 2: If $\|g_k\| \leq \varepsilon$, then stop.

Step 3: Find the step length, where the calculation $\alpha_k = \max\{\gamma^k \mid k = 0, 1, 2, \ldots\}$ stems from (1.9).

Step 4: Set the new iteration point of $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Update the search direction by (2.4).

Step 6: If $\|g_{k+1}\| \leq \varepsilon$ holds, the algorithm stops. Otherwise, go to next step.

Step 7: Let $k := k + 1$ and go to Step 3.

2.2 Algorithm characteristics

This section states the properties of the sufficient descent, trust region as well as global convergence of Algorithm 2.1.

Lemma 2.1 *If the search direction d_k is generated by* (2.4), *then*

$$g_k^T d_k = -\|g_k\| \|g_k\| \tag{2.5}$$

and

$$\|d_k\| \le (1 + 2/\xi_2)\|g_k\|, \tag{2.6}$$

where σ are positive constants.

Proof On the one hand, it is true that (2.5) and (2.6) are correct if $k = 0$.
On the other hand, from (2.4),

$$\begin{aligned} g_{k+1}^T d_{k+1} &= g_{k+1}^T \left[-g_{k+1} + \frac{g_{k+1}^T y_k^* d_k - d_k^T g_{k+1} y_k^*}{\max\{\xi_2 \|d_k^T\| \|y_k^*\|, \min(\xi_3 \|g_k\|^2, \xi_4 d_k^T d_k)\}} \right] \\ &= -\|g_{k+1}\|^2 + \frac{g_{k+1}^T g_{k+1}^T y_k^* d_k - g_{k+1}^T d_k^T g_{k+1} y_k^*}{\max\{\xi_2 \|d_k^T\| \|y_k^*\|, \min(\xi_3 \|g_k\|^2, \xi_4 d_k^T d_k)\}} \\ &= -\|g_{k+1}\|^2 \end{aligned}$$

and

$$\begin{aligned} \|d_{k+1}\| &= \left\| g_{k+1} + \frac{g_{k+1}^T y_k^* d_k - d_k^T g_{k+1} y_k^*}{\max\{\xi_2 \|d_k^T\| \|y_k^*\|, \min(\xi_3 \|g_k\|^2, \xi_4 d_k^T d_k)\}} \right\| \\ &\le \|g_{k+1}\| + \left\| \frac{g_{k+1}^T y_k^* d_k - d_k^T g_{k+1} y_k^*}{\max\{\xi_2 \|d_k^T\| \|y_k^*\|, \min(\xi_3 \|g_k\|^2, \xi_4 d_k^T d_k)\}} \right\| \\ &\le \|g_{k+1}\| + \frac{\|g_{k+1}\| \|y_k^*\| \|d_k\| + \|d_k\| \|g_{k+1}\| \|y_k^*\|}{\xi_2 \|d_k^T\| \|y_k^*\|} \\ &\le \|g_{k+1}\| + 2\frac{\|g_{k+1}\| \|y_k^*\| \|d_k\|}{\xi_2 \|d_k^T\| \|y_k^*\|} \\ &= (1 + 2/\xi_2)\|g_{k+1}\|. \end{aligned} \tag{2.7}$$

It is true that (2.5) and (2.6) demonstrate that the search direction has a sufficient descent trait and a trust region property, respectively. □

Aiming at achieving global convergence, we propose the following mild assumptions.

Assumption (i) The level set of $\Omega = \{x \mid f(x) \le f(x_0)\}$ is bounded.

Assumption (ii) The objective function $f(x) \in C^2$ is bounded from below, and its gradient function $g(x)$ is Lipschitz continuous, i.e., there exists a positive constant τ such that

$$\|g(x) - g(y)\| \le \tau \|x - y\|, \quad x, y \in R^n. \tag{2.8}$$

Based on the above discussion and established conclusion concerning the modified Armijo line search of being reasonable and necessary (see [48]), the global convergence algorithm is established as follows.

Theorem 2.1 *If assumptions (i)–(ii) are true and the corresponding sequences of $\{x_k\}$, $\{d_k\}$, $\{g_k\}$, $\{\alpha_k\}$ are generated by Algorithm 2.1, then we arrive at the conclusion that*

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{2.9}$$

Proof Suppose that the conclusion of the above theorem is incorrect, i.e., there exist a positive constant σ_3 and index number k' such that

$$\|g_k\| \geq \sigma_3, \quad \forall k \geq k'. \tag{2.10}$$

Based on (1.9) and (2.5),

$$\begin{aligned} f(x_k+\alpha_k d_k) &\leq f(x_k) + \lambda\alpha_k g_k^T d_k + \alpha_k \min\left[-\lambda_1 g_k^T d_k, \lambda\frac{\alpha_k}{2}\|d_k\|^2\right] \\ &\leq f(x_k) + \alpha_k(\lambda-\lambda_1)g_k^T d_k. \end{aligned}$$

Then with the above formulas with $k = 0$ from ∞ and combining with Assumption (ii), we obtain

$$\sum_{k=0}^{\infty} \alpha_k(\lambda-\lambda_1)g_k^T d_k \leq f_0 - f_\infty < \infty. \tag{2.11}$$

Based on the convergence theorem of sequences,

$$\lim_{k\to\infty}(\lambda-\lambda_1)\alpha_k g_k^T d_k = 0. \tag{2.12}$$

Then we have

$$\lim_{k\to\infty}\alpha_k g_k^T d_k = 0, \tag{2.13}$$

from the formula of (2.5), then

$$\lim_{k\to\infty}\alpha_k\|g_k\|^2 = 0. \tag{2.14}$$

This means that $\{\alpha_k\} \to 0$, $k \to \infty$ or $\{\|g_k\|\} \to 0$, $k \to \infty$. We then state two cases:

(i) If $\{\alpha_k\} \to 0$, $k \to \infty$, consider the line search method, for every suitable parameter α_k,

$$f(x_k + \alpha_k/\gamma d_k) > f(x_k) + \lambda\alpha_k/\gamma g_k^T d_k + \alpha_k/\gamma \min\left(-\lambda_1 g_k^T d_k, \lambda\alpha_k\|d_k\|^2/(2\gamma)\right),$$

there thus exists a positive constant $\lambda^* \leq \lambda_1$, such that

$$f(x_k+\alpha_k d_k/\gamma) - f(x_k) \geq -\left(\lambda-\lambda^*\right)\alpha_k/\gamma\,\|g_k\|^2.$$

Using (2.5), Assumption (ii) and the continuity of $f(x)$ and $g(x)$, we have

$$\begin{aligned} f(x_k+\alpha_k d_k/\gamma) - f(x_k) &= \alpha_k/\gamma g(x_k+\eta_k\alpha_k/\gamma d_k)^T d_k \\ &= \alpha_k/\gamma g(x_k)^T d_k + \left[\alpha_k/\gamma g(x_k+\eta_k\alpha_k/\gamma d_k) - g(x_k)\right]^T d_k \\ &\leq \alpha_k/\gamma g(x_k)^T d_k + \eta_k\tau\alpha_k^2/\gamma^2\|d_k\|^2 \\ &= -\alpha_k/\gamma\left\|g(x_k)\right\|^2 + \eta_k\tau\alpha_k^2/\gamma^2\|d_k\|^2, \end{aligned}$$

where $\eta_k \in (0,1)$. Comparing the above two expressions, we have

$$-\alpha_k/\gamma \|g_{x_k}\|^2 + \eta_k \tau \alpha_k^2/\gamma^2 \|d_k\|^2 \geq -(\lambda - \lambda^*)/\gamma \alpha_k \|g_{x_k}\|^2$$

i.e.,

$$\|g_{x_k}\|^2 - \eta_k \tau \alpha_k/\gamma \left\|d_k^2\right\| \leq \left(\lambda - \lambda^*\right) \left\|g(x_k)\right\|^2.$$

Thus,

$$\alpha_k \geq \gamma\left(1 - \lambda + \lambda^*\right)/\left(\eta_k \tau \sigma^2\right) > 0,$$

where $\sigma \in (1 + \frac{2}{\xi_2}, \infty)$. This contradicts the assumption of case (i).

(ii) Clearly, $\{g_k\} \to 0$ if α_k is a positive finite constant when k is a sufficiently large constant from the formula of (2.14). This conclusion does not satisfy the assumption of (2.10); this completes the proof. □

2.3 Numerical results

Related content is presented in this section and consists of two parts: test problems and corresponding numerical results. To measure the algorithm's efficiency, we compare Algorithm 2.1 with Algorithm 1 in [51] in terms of NI, NFG, and CPU on the test problems listed in Table 2 of Appendix 1, which are from [3], where NI, NFG, and CPU indicate the number of iterations, the sum of the calculation's frequency of the objective function and gradient function, and the calculation time needed to solve various test problems (in seconds), respectively. Algorithm 1 is different from the objective algorithm in the formula of d_{k+1} that was determined by (2.1), and the remainder of Algorithm 1 is identical to Algorithm 2.1.

Stopping rule: If $|f(x_k)| > e_1$, let $stop1 = \frac{|f(x_k)-f(x_{k+1})|}{|f(x_k)|}$ or $stop1 = |f(x_k) - f(x_{k+1})|$. If the condition $\|g(x)\| < \epsilon$ or $stop1 < e_2$ is satisfied, the algorithm stops, where $e_1 = e_2 = 10^{-4}$, $\epsilon = 10^{-4}$. On the one hand, based on the virtual case, the proposed algorithm also stops if the number of iterations is greater than 10,000 and the iteration number of α_k is greater than 5. On the other hand, 'NO' and 'problem' in Table 2 indicate the number of the tested problem and the name of the problem, respectively.

Initiation: $\lambda = 0.9$, $\lambda_1 = 0.4$, $\xi_3 = 300$, $\xi_2 = \xi_4 = 0.01$, $\gamma = 0.01$.

Dimension: 30,000, 90,000, 150,000, 210,000.

Calculation environment: The calculation environment is a computer with 2 GB of memory, a Pentium (R) Dual-Core CPU E5800@3.20 GHz, and the 64-bit Windows 7 operating system.

The algorithms' numerical results are listed in Table 3 of Appendix 1 with their corresponding NI, NFG and CPU. Then, based on the technique in [15], plots of the corresponding figures are presented for the proposed algorithm. Some of the test problems are especially complex in that the above algorithms fail to solve them. Thus, a list of the numerical results with the corresponding problem index is given in Table 3, where 'NO' and 'Dim' are the index of the problem and the dimension of the variable, respectively. Clearly, the proposed algorithm (Algorithm 2.1) is effective from the above figures because the point's value on the algorithm's curve is larger than for the other algorithms. In Fig. 1, the

Figure 1 Performance profiles of these methods (NI)

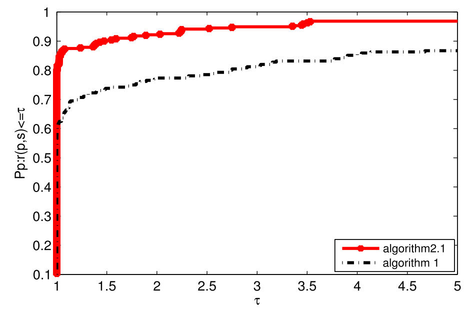


Figure 2 Performance profiles of these methods (NFG)

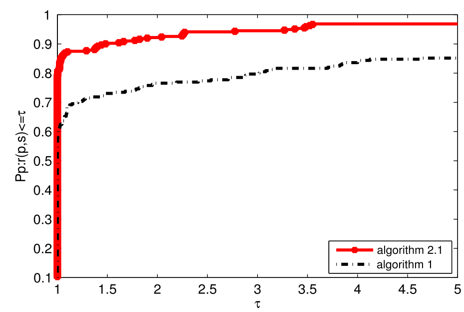
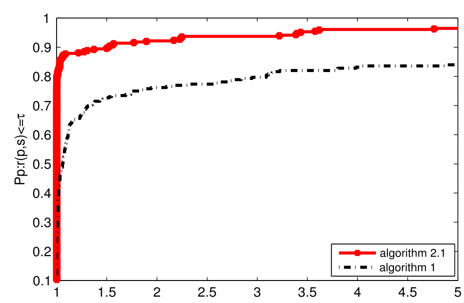


Figure 3 Performance profiles of these methods (CPU time)



red curve's value of the initial point is close to 0.9, while Algorithm 1 only arrives at a value of 0.6. This means that Algorithm 2.1 addresses complex problems fewer iterations. In Fig. 2, the red curve is above the left curve because the calculation number of the objective function is less than the left one when addressing practical problems, which is an important aspect for measuring the performance of an algorithm. It is well know that the calculation time (CPU time) is the most essential metric of an algorithm because the speed of the algorithm is the most basic feature. The objective algorithm not only is well defined because the curve seems more feasible and smooth but also can address most complex problems because the largest value of the point on the curve is close to 0.98, which indicates that the proposed algorithm is highly effective. Overall, the proposed algorithm cannot only solve smooth problems but it also enriches the knowledge of optimization and lays the foundation for further in-depth studies.

3 Algorithm for nonsmooth problems

From the previous section, the proposed algorithm is trustworthy and has good potential based on the fundamental numerical results. Thus, this section attempts to apply the proposed method to nonsmooth problems. It is interesting that the vast majority of practical conditions are harsh; therefore, Newton's series of methods are often unsatisfactory for solving such problems because they require information about the gradient function [37, 39, 46]. Currently, most experts and scholars focus on bundled methods, which are successful solutions to small-scale problems (see [18, 19, 24, 36]) but fail to solve large-scale practical problems. With the development of science and technology, it is becoming an urgent need to design a simple but effective algorithm to solve large-scale nonsmooth problems. Based on the simplicity of the conjugate gradient method, some experts and scholars have proposed relevant algorithms and made numerous fruitful theoretical achievements (see [20, 28]).

Consider the following problem:

$$\min \theta(x), \tag{3.1}$$

where $\theta(x)$ sometimes is nonsmooth and $x \in R^n$. A famous technique is called 'Moreau–Yosida' regularization, which calculates the equivalent solution of the previous problem through a modified object function with the formula

$$\min_{t\in R^n}\left\{\theta(t) + \frac{\|t-x\|^2}{2\chi}\right\}, \tag{3.2}$$

where χ and $\|\cdot\|$ denote a positive constant and the Euclidean norm, respectively. Without loss of generality, we denote $\theta^M(x)$ as in (3.1) 'Moreau–Yosida' regularization, i.e.,

$$\theta^M(x) = \min_{t\in R^n}\left\{\theta(t) + \frac{\|t-x\|^2}{2\chi}\right\}. \tag{3.3}$$

The 'Moreau–Yosida' regularization technique was introduced because of its outstanding properties such as differentiability (see [21, 25]). Assume that the function (3.1) is convex and that its 'Moreau–Yosida' regularization obtains the best solution of $\omega(x) = \operatorname{argmin} \theta^M(t)$. Based on the mathematic knowledge and the established conclusion, we then have

$$\nabla\theta^M(x) = \frac{\omega(x) - x}{\chi}. \tag{3.4}$$

The most surprising result is that the function $\theta^M(x)$ is instantly smooth and that its gradient function is Lipschitz continuous, which is not true for the original function. It is worth noting that (3.1) and (3.3) are equivalent to each other because they have the same solution. In the remainder of this discussion, we attempt to further study the goal of (3.3) because it is clear and brief to a large extent, and we introduce relevant components of the objective algorithm simultaneously. Some necessary properties of sufficient descent and trust region will be listed in the next section. Then we provide relevant numerical results to show the performance of the proposed algorithm and draw a conclusion with regard of the whole paper.

3.1 New algorithm and its necessary properties

We start with the following formula for d_{k+1}, which is an important component of the proposed algorithm in addressing complex problems:

$$d_{k+1} = \begin{cases} -\nabla\theta^M(x_{k+1})^T + \frac{\nabla\theta^M(x_{k+1})y_k d_k - d_k^T\nabla\theta^M(x_{k+1})y_k}{\max\{\xi_2\|d_k\|\|y_k\|,\min\{\xi_3\|\nabla\theta^M(x_k)\|^2,\xi_4\|d_k\|^2\}\}}, & \text{if } k \geq 1, \\ -\nabla\theta^M(x_{k+1}), & \text{if } k = 0, \end{cases} \tag{3.5}$$

where $s_k = x_{k+1} - x_k$, $y_k = \nabla\theta^M(x_{k+1}) - \nabla\theta^M(x_k)$, and ξ_2, ξ_3, and ξ_4 are positive constants. The step length α_k is determined by

$$\begin{aligned} \theta^M(x_k + \alpha_k d_k) \leq{}& \theta^M(x_k) + \lambda\alpha_k\nabla\theta^M(x_k)^T d_k \\ &+ \alpha_k \min\bigl[-\lambda_1\nabla\theta^M(x_k)^T d_k, \lambda\alpha_k d_k^T d_k/2\bigr], \end{aligned} \tag{3.6}$$

where $\lambda, \gamma \in (0,1)$, $\lambda_1 \in (0,\lambda)$, and α_k is the largest number of $\{\gamma^k \mid k = 0,1,2,\ldots\}$. It is well known from the case of smooth functions that the new search direction has satisfactory descent and trust region properties; therefore, we merely list them without proof. We have

$$\nabla\theta^M(x_k)^T d_k = -\bigl\|\nabla\theta^M(x_k)\bigr\|^2, \tag{3.7}$$

$$\|d_k\| \leq \sigma\bigl\|\nabla\theta^M(x_k)\bigr\|, \tag{3.8}$$

where σ is the same as in (2.7). Now, manifesting specific algorithm steps, we express the cause of the existing $\alpha_k' \in \Re$ that satisfies the demands of the modified Armijo line search formula and provides the global convergence of the proposed algorithm.

Algorithm 3.1

Step 1: (Initiation) Choose an initial point x_0, $\gamma \in (0,1)$, ξ_2, ξ_3, and $\xi_4 > 0$ and positive constants $\varepsilon \in (0,1)$. Let $k = 0$, $d_0 = -\nabla\theta^M(x_0)$.

Step 2: If $\|\nabla\theta^M(x_k)\| \leq \varepsilon$, then stop.

Step 3: Find the step length, i.e., the calculation $\alpha_k = \max\{\gamma^k \mid k = 0,1,2,\ldots\}$ stemming from (3.6).

Step 4: Set the new iteration point $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Update the search direction by (3.5).

Step 6: If $\|\nabla\theta^M(x_{k+1})\| \leq \varepsilon$ holds, the algorithm stops; otherwise, go to the next step.

Step 7: Let $k := k + 1$ and go to Step 3.

To express the validity of the step length α_k in (3.6) and the global convergence of Algorithm 3.1, the following assumptions are necessary.

Assumption

(i) The level set $\pi = \{x \mid \theta^M(x) \leq \theta^M(x_0)\}$ is bounded.

(ii) The function $\theta^M(x) \in C^2$ is bounded from below.

From the 'Moreau–Yosida' regularization technique, the function $\theta^M(x)$ is Lipschitz continuous, i.e., there exists a positive constant κ subject to

$$\left\|\nabla\theta^M(x)-\nabla\theta^M(y)\right\|\le\kappa\|x-y\|. \tag{3.9}$$

Theorem 3.1 *If Assumptions* (i)–(ii) *are true, then there exists a constant* α_k *that satisfies the requirements of* (3.6).

Proof We introduce the following function:

$$\begin{aligned}\vartheta(\alpha)={}&\theta^M(x_k+\alpha d_k)-\theta^M(x_k)-\lambda\alpha\nabla\theta^M(x_k)^Td_k\\&-\alpha\min\left[-\lambda_1\nabla\theta^M(x_k)^dk,\lambda\alpha d_k^Td_k/2\right].\end{aligned} \tag{3.10}$$

Based on the established theorem, following the sufficient decrease of (3.7), for sufficiently small positive α, we have

$$\begin{aligned}\vartheta(\alpha)={}&\theta^M(x_k+\alpha d_k)-\theta^M(x_k)-\lambda\alpha\nabla\theta^M(x_k)^Td_k\\&-\alpha\min\left[-\lambda_1\nabla\theta^M(x_k)^Td_k,\lambda\alpha d_k^Td_k/2\right]\\={}&\alpha(1+\lambda_1-\lambda)\nabla\theta^M(x_k)^Td_k+o(\alpha)<0,\end{aligned}$$

where the latter inequality holds since the objective function is continuous. Thus, there exists a constant $0<\alpha_0<1$ such that $\vartheta(\alpha_0)<0$; on the other hand, $\vartheta(0)=0$, and based on the function's continuous property, there exists a constant α_1 such that

$$\begin{aligned}\vartheta(\alpha_1)={}&\theta^M(x_k+\alpha_1 d_k)-\theta^M(x_k)-\lambda\alpha_1\nabla\theta^M(x_k)^Td_k\\&-\alpha_1\min\left[-\lambda_1\nabla\theta^M(x_k)^dk,\lambda\alpha_1 d_k^Td_k/2\right]<0.\end{aligned}$$

Thus,

$$\begin{aligned}\theta^M(x_k+\alpha_1 d_k)<{}&\theta^M(x_k)+\lambda\alpha_1\nabla\theta^M(x_k)^Td_k\\&+\alpha_1\min\left[-\lambda_1\nabla\theta^M(x_k)^Td_k,\lambda\alpha_1 d_k^Td_k/2\right]\end{aligned}$$

is correct, and this means that the modified Armijo line search is well defined. From the above discussion, Algorithm 3.1 has the properties of the sufficient descent and a trust region, and we can now present the theorem of global convergence. □

Theorem 3.2 *If the above assumptions are satisfied and the relative sequences* $\{x_k\}$, $\{\alpha_k\}$, $\{d_k\}$, $\{\theta^M(x_k)\}$ *are generated by Algorithm* 3.1, *then we have* $\lim_{k\to\infty}\|\nabla\theta^M(x_k)\|=0$.

We neglect the proof because its proof is similar to that of Theorem 2.1.

3.2 Nonsmooth numerical experiment

Two algorithms are proposed and compared to the proposed algorithm because this section measures the objective algorithm's efficiency on the test problems listed in Table 4. In addition, the problems are only different from Algorithm 3.1 in the formula for d_{k+1}. The relevant numerical data are listed in Table 5 of Appendix 2, and we plot the corresponding graphs based on these data, where 'NI', 'NF', and 'CPU' are the iteration number, calculation number of the objective function and the algorithm's run time (in seconds). The first metric is determined by

$$d_{k+1} = \begin{cases} -\nabla\theta^M(x_{k+1}) + \frac{\nabla\theta^M(x_{k+1}))^T y_k d_k - d_k^T \nabla\theta^M(x_{k+1}) y_k}{\nabla\theta^M(x_k)^T \nabla\theta^M(x_k)}, & \text{if } k \geq 1, \\ -\nabla\theta^M(x_{k+1}), & \text{if } k = 0. \end{cases} \tag{3.11}$$

In [51], without loss of generality, calling Algorithm 2, the other algorithm in [4] is calculated as

$$d_{k+1} = \begin{cases} \frac{-y_k^T s_k \nabla\theta^M(x_{k+1}) + y_k^T \nabla\theta^M(x_{k+1}) s_k - s_k^T g_{k+1} y_k}{\|\nabla\theta^M(x_k)\|^2}, & \text{if } k \geq 1, \\ -\nabla\theta^M(x_{k+1}), & \text{if } k = 0, \end{cases} \tag{3.12}$$

denoted as Algorithm 3.

Dimension: 150,000, 180,000, 192,000, 210,000, 222,000, 231,000, 240,000, 252,000, 270,000.

Initiation: $\lambda = 0.9$, $\lambda_1 = 0.4$, $\xi_3 = 100$, $\xi_2 = \xi_4 = 0.01$, $\gamma = 0.5$.

Stopping rule: If NI is no greater than 10,000, $|f(x_{k+1}) - f(x_k)| < 1e-7$ and if the iteration number of α_k is no greater than 5, then the algorithm stops.

Calculation environment: The calculation environment is a computer with 2 GB of memory, a Pentium (R) Dual-Core CPU E5800@3.20 GHz and the 64-bit Windows 7 operating system.

From Figs. 4–6, the proposed algorithm is effective and successful to a large extent. First, the computational data of the algorithm fully address complex situations. Second, the algorithm in the design of the search direction carefully considers the corresponding function, gradient function and current direction. In Figs. 4 and 5, the curve of Algorithm 3.1 is above the other two curves because the number of iterations is much lower. Its initial point is close to 0.75, which is much larger than the other algorithms. Note that the proposed algorithm's computation time is the best of the three algorithms because the curve increases rapidly and is very smooth. In other words, its curve has a wonderful initiation point, which results in a high efficiency in addressing complex issues.

Figure 4 Performance profiles of these methods (NI)

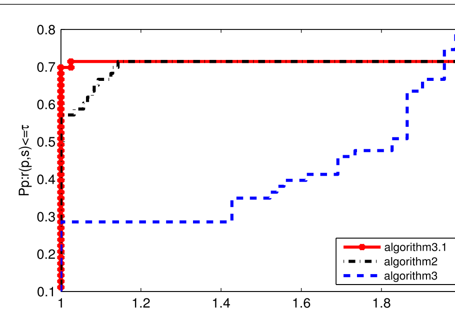


Figure 5 Performance profiles of these methods (NF)

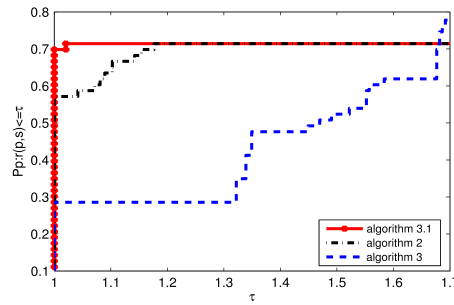
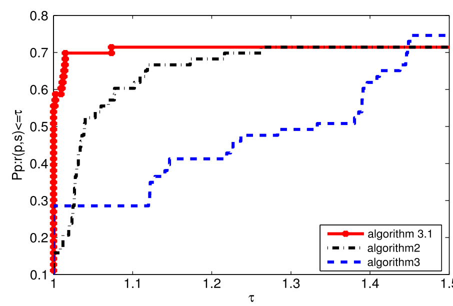


Figure 6 Performance profiles of these methods (CPU)



4 Applications of Algorithm 3.1 in image restoration

It is well known that many modern applications of optimization call for studying large-scale nonsmooth convex optimization problems, where the image restoration problem arising in image processing is an illustrating example. The image restoration problem plays an important role in biological engineering, medical sciences and other areas of science and engineering (see [6, 10, 32] etc.), which is to reconstruct an image of an unknown scene from an observed image. The most common image degradation model is defined by the following system:

$$b = Ax + \eta,$$

where $x \in \Re^n$ is the underlying images, $b \in \Re^m$ is the observed images, A is an $m \times n$ blurring matrix, and $\eta \in \Re^m$ denotes the noise. One way to get the unknown η is to solving the problem $\min_{x\in\Re^n} \|Ax + b\|^2$. This problem will not have a satisfactory solution since the system is very sensitive to lack of information and the noise. The regularized least square problem is often used to overcome the above shortcoming

$$\min_{x\in\Re^n} \|Ax + b\|^2 + \lambda\|Dx\|_1,$$

where $\|\cdot\|_1$ is the l^1 norm, λ is the regularization parameter controlling the trade-off between the regularization term and the data-fitting term, and D is a linear operator. It is easy to see that the above problem is a nonsmooth convex optimization problem and it is typically of large scale since the l^1 norm is nonsmooth.

4.1 Image restoration problem

The above section tells us that Algorithm 3.1 can be used for large-scale nonsmooth problems. Then we will use this algorithm to solve the above image noise problem, where the

Table 1 The CPU time of PRP algorithm and Algorithm 3.1 in seconds

	Lena	Cameraman	Barbara	Banoon	Man
20% noise					
PRP algorithm	1.796875	9.6875	1.421875	1.375	3.375
Algorithm 3.1	1.375	0.765625	1.46875	1.40625	3.328125
50% noise					
PRP algorithm	1.98437	1.000	2.296875	1.71875	5.1875
Algorithm 3.1	1.82812	0.921875	1.96875	1.8125	4.953125

parameters are the same as those of Sect. 3.2 different from $\xi_2 = \xi_3 = \xi_4 = 1$. All codes are written by MATLAB r2017a and run on a PC with an Intel Pentium(R) Xeon(R) E5507 CPU @2.27 GHz, 6.00 GB of RAM, and the Windows 7 operating system. The stopped condition is

$$\frac{|\theta_\alpha(v_{k+1}) - \theta_\alpha(v_k)|}{|\theta_\alpha(v_k)|} \leq 10^{-2}$$

or

$$\frac{\|v_{k+1} - v_k\|}{\|v_k\|} \leq 10^{-2},$$

where

$$\theta_\alpha(v_k) = \sum_{(i,j)\in N} \left\{ \sum_{(m,n)\in V_{i,j}\setminus N} \varphi_\alpha(v_{i,j} - y_{m,n}) + \frac{1}{2} \sum_{(m,n)\in V_{i,j}\cap N} \varphi_\alpha(v_{i,j} - v_{m,n}) \right\},$$

the noise candidate indices set $N := \{(i,j) \in A \mid \bar{y}_{i,j} \neq y_{i,j}, y_{i,j} = s_{\min} \text{ or } s_{\max}\}$, $s_{\max}$ is the maximum of the noisy pixel and $s_{\min}$ denotes the minimum of the noisy pixel, $A = \{1, 2, \ldots, M\} \times \{1, 2, 3, \ldots, N\}$, $V_{i,j} = \{(i, j-1), (i, j+1), (i-1, j), (i+1, j)\}$ is the neighborhood of (i,j), y denotes the observed noisy image of x corrupted by the salt-and-pepper noise, $\bar{y}$ is defined by the image obtained by applying the adaptive median filter method to the noisy image y in the first phase, x is the true image with M-by-N pixels, and $x_{i,j}$ denotes the gray level of x at pixel location (i,j). It is easy to see that the regularity of θ_α only depends on φ_α and there exist many properties as regards θ_α and φ_α that are studied by many scholars (see [7, 8] etc.). In the experiments, Lena (256×256), Cameraman (256×256), Barbara (512×512), Banoon (512×512), and Man (1024×1024) are the tested images. To compare Algorithm 3.1 with other similar algorithm, we also test the well-known PRP conjugate gradient algorithm, where the Step 5 of Algorithm 3.1 is replaced by the PRP formula. The tested performances of these two algorithms (Algorithm 3.1 and PRP algorithm) are listed and the spent time is stated in Table 1.

4.2 Results and discussion

Figures 7 and 8 show that Algorithm 3.1 and PRP algorithm have good performance to solve the image restoration and both of them can successfully do this problem. From the results of Table 1 it turns out that Algorithm 3.1 is competitive to PRP algorithm since it needs less CPU time to restoration of the most given images than those of the PRP algorithm.

Figure 7 Restoration of the images Lena, Cameraman, Barbara, Banoon, and Man by PRP algorithm and Algorithm 3.1. From left to right: the noisy image with 20% salt-and-pepper noise, the restorations obtained by minimizing z with PRP algorithm and Algorithm 3.1, respectively

5 Conclusion

This paper proposes a new PRP algorithm that combines the innovative formula of the search direction d_{k+1} with the modified Armijo line technique: (i) In the design of the proposed algorithm, the key information about the objective function, the gradient function and its current direction is collected and applied to complex problems, and the numerical results show that the proposed algorithm is efficient. (ii) For nonsmooth problems, the introduced 'Moreau–Yosida' regularization technique succeeds in enhancing the proposed

Figure 8 Restoration of the images Lena, Cameraman, Barbara, Banoon, and Man by PRP algorithm and Algorithm 3.1. From left to right: the noisy image with 50% salt-and-pepper noise, the restorations obtained by minimizing *z* with PRP algorithm and Algorithm 3.1, respectively

algorithm, and the numerical results prove the validity and simplicity of the discussed algorithm. (iii) Image restoration problems are done by Algorithm 3.1 and from the tested results it turns out that the given algorithm has better performance than those of the normal PRP algorithm. However, there are some problems with the optimization method that need to be studied such as how to better leverage the benefits of the steepest descent method while overcoming its shortcomings.

Appendix 1

Table 2 Test problems

No.	Problem	No.	Problem
1	Extended Freudenstein and Roth Function	33	TRIDIA Function (CUTE)
2	Extended Trigonometric Function	34	ARWHEAD Function (CUTE)
3	Extended Rosenbrock Function	35	NONDQUAR Function (CUTE)
4	Extended White and Holst Function	36	DQDRTIC Function (CUTE)
5	Extended Beale Function	37	EG2 Function (CUTE)
6	Raydan 1 Function	38	DIXMAANA Function (CUTE)
7	Raydan 2 Function	39	DIXMAANB Function (CUTE)
8	Diagonal 1 Function	40	DIXMAANC Function (CUTE)
9	Diagonal 2 Function	41	DIXMAANE Function (CUTE)
10	Hager Function	42	Broyden Tridiagonal Function
11	Generalized Tridiagonal 1 Function	43	Almost Perturbed Quadratic Function
12	Extended Tridiagonal 1 Function	44	Tridiagonal Perturbed Quadratic Function
13	Extended Three Exponential Terms Function	45	EDENSCH Function (CUTE)
14	Generalized Tridiagonal 2 Function	46	STAIRCASE S1 Function
15	Diagonal 4 Function	47	LIARWHD Function (CUTE)
16	Diagonal 5 Function	48	DIAGONAL 6 Function
17	Extended Himmelblau Function	49	DIXON3DQ Function (CUTE)
18	Generalized PSC1 Function	50	DIXMAANF Function (CUTE)
19	Extended PSC1 Function	51	DIXMAANG Function (CUTE)
20	Extended Powell Function	52	DIXMAANH Function (CUTE)
21	Extended Block Diagonal BD1 Function	53	DIXMAANJ Function (CUTE)
22	Extended Maratos Function	54	DIXMAANL Function (CUTE)
23	Extended Cliff Function	55	DIXMAAND Function (CUTE)
24	Quadratic Diagonal Perturbed Function	56	ENGVAL1 Function (CUTE)
25	Extended Wood Function	57	FLETCHCR Function (CUTE)
26	Extended Hiebert Function	58	COSINE Function (CUTE)
27	Quadratic Function QF1 Function	59	Extended DENSCHNB Function (CUTE)
28	Extended Quadratic Penalty QP1 Function	60	DENSCHNF Function (CUTE)
29	Extended Quadratic Penalty QP2 Function	61	SINQUAD Function (CUTE)
30	A Quadratic Function QF2 Function	62	BIGGSB1 Function (CUTE)
31	Extended EP1 Function	63	Partial Perturbed Quadratic PPQ2 Function
32	BDQRTIC (CUTE)	64	Scaled Quadratic SQ1 Function
		65	BDQRTIC Function (CUTE)

Table 3 Numerical results

NO	Dim	Algorithm 2.1			Algorithm 1		
		NI	NFG	CPU	NI	NFG	CPU
1	30,000	572	2832	17.170120	575	2844	17.689130
2	30,000	217	1082	10.122250	63	312	2.980625
3	30,000	43	170	0.886250	47	186	0.995250
4	30,000	1014	4385	23.856440	2007	8265	44.962060
5	30,000	312	1145	1.994063	5204	20,616	32.667130
6	30,000	623	3084	13.556880	607	3004	15.491130
7	30,000	195	584	1.625188	195	584	2.082937
8	30,000	114	567	2.542562	114	567	3.302094
9	30,000	78	297	0.970500	1799	3598	16.080500
10	30,000	276	1100	3.950344	256	1020	3.633312
11	30,000	116	448	3.119172	46	137	0.968500
12	30,000	214	640	2.761031	835	2460	10.498130
13	30,000	5	16	0.093313	78	233	1.184125
14	30,000	4909	19,634	285.120600	1464	5821	83.039530
15	30,000	551	2201	7.004469	1430	4822	15.256470
16	30,000	4	8	0.078563	4	8	0.061656
17	30,000	3916	15,660	56.706440	3725	14,896	53.351160
18	30,000	570	2260	45.941410	649	2496	50.545590

Table 3 (*Continued*)

NO	Dim	Algorithm 2.1			Algorithm 1		
		NI	NFG	CPU	NI	NFG	CPU
19	30,000	539	2096	22.635000	606	2297	25.423720
20	30,000	1485	5904	16.661230	1201	4573	13.690940
21	30,000	12	37	0.187125	424	1271	7.530531
22	30,000	54	214	0.796953	54	214	0.811281
23	30,000	7833	15,670	89.310080	7796	15,596	90.177780
24	30,000	747	3698	14.320530	2989	14,796	57.666090
25	30,000	828	3854	27.081220	1271	5619	40.056130
26	30,000	1088	5415	17.331190	24	114	0.383531
27	30,000	10,000	49,997	46.690160	3638	18,022	9.802813
28	30,000	766	3772	12.792700	766	3772	13.070120
29	30,000	890	4381	23.664730	890	4381	24.005910
30	30,000	1287	6398	38.251670	3898	19,316	118.245100
31	30,000	24	94	0.155719	24	94	0.151094
32	30,000	26	77	0.249469	26	77	0.251750
33	30,000	1470	7328	25.383470	3749	18,573	65.219060
34	30,000	80	397	4.352656	80	397	4.436313
35	30,000	723	3607	20.137970	1370	5627	34.018590
36	30,000	10,000	39,998	344.076300	4452	17,604	153.518100
37	30,000	98	486	1.090125	98	486	1.215531
38	30,000	666	2246	22.198280	651	2198	21.966780
39	30,000	2	7	0.062063	2	7	0.064031
40	30,000	2880	14,397	115.767600	2879	14,392	117.060600
41	30,000	510	1795	17.488940	546	1846	18.727130
42	30,000	34	135	0.125313	689	2707	2.399625
43	30,000	5492	27,391	15.740310	3729	18,476	10.305280
44	30,000	6489	32,372	123.601100	4540	22,492	81.385600
45	30,000	527	2092	20.201500	535	2115	20.517880
46	30,000	167	502	1.542719	1664	4911	15.354250
47	30,000	24	117	0.717750	24	117	0.716781
48	30,000	893	2678	5.958000	893	2678	6.079031
49	30,000	65	215	0.265969	646	1909	2.798000
50	30,000	2	7	0.062344	2	7	0.065219
51	30,000	1308	6537	52.088220	1310	6547	53.276810
52	30,000	7764	38,817	331.998400	7765	38,822	336.152700
53	30,000	509	1793	17.501380	539	1827	18.521470
54	30,000	2	7	0.064375	2	7	0.066031
55	30,000	2059	10,292	85.489190	2061	10,302	89.237690
56	30,000	1088	4350	3.479906	1088	4350	3.700406
57	30,000	2	6	0.014438	2	6	0.002281
58	30,000	19	56	0.295813	34	100	0.518375
59	30,000	501	1502	8.905937	501	1502	9.458844
60	30,000	962	3918	45.193690	1033	4129	49.081910
61	30,000	2	7	0.060375	2	7	0.062875
62	30,000	65	215	0.266781	646	1909	2.940375
63	30,000	1	215	5.836156	1	1909	5.956969
64	30,000	10,000	49,846	26.754220	7151	35,419	19.496940
65	30,000	10,000	49,863	26.738310	4461	22,096	12.329940
1	90,000	573	2836	51.941500	575	2844	52.787310
2	90,000	30	147	4.165094	5	22	0.633250
3	90,000	44	174	2.746031	47	186	3.082438
4	90,000	1551	6524	107.156900	2007	8265	137.040400
5	90,000	722	2796	15.007690	5626	22,298	114.106400
6	90,000	338	1687	23.915000	339	1692	24.350190
7	90,000	195	584	4.992938	195	584	5.162625
8	90,000	13	31	0.354344	2805	5610	68.010120
9	90,000	241	961	10.348970	226	901	9.954906
10	90,000	39	151	3.215312	46	137	3.037875
11	90,000	215	643	8.439844	1118	3291	44.010810
12	90,000	5	16	0.248438	78	233	3.493094
13	90,000	5188	20,750	909.355400	1498	5955	268.589000

Table 3 (*Continued*)

NO	Dim	Algorithm 2.1			Algorithm 1		
		NI	NFG	CPU	NI	NFG	CPU
14	90,000	854	3388	33.367970	1485	4987	50.862160
15	90,000	4	8	0.203781	4	8	0.201875
16	90,000	4097	16,384	181.834700	3938	15,748	178.470700
17	90,000	577	2282	139.602700	651	2501	151.183100
18	90,000	540	2099	68.454150	606	2297	76.485500
19	90,000	1912	7602	66.348030	1201	4573	42.566220
20	90,000	35	108	1.810375	450	1349	24.126810
21	90,000	54	214	2.466781	54	214	2.486906
22	90,000	7833	15,670	274.592900	7796	15,596	283.996600
23	90,000	93	460	5.444188	1085	5312	66.299620
24	90,000	961	4387	93.333880	1271	5619	120.455800
25	90,000	1088	5415	53.056840	24	114	1.183812
26	90,000	2051	10180	20.263250	3731	18,486	35.322310
27	90,000	748	3701	38.751530	748	3701	39.540090
28	90,000	891	4387	71.637530	891	4387	72.518130
29	90,000	1284	6402	105.456800	3898	19,320	332.476700
30	90,000	24	94	0.467781	24	94	0.483219
31	90,000	26	77	0.779031	26	77	0.777844
32	90,000	1133	5635	59.076220	3249	16,242	190.573400
33	90,000	24	117	3.867094	24	117	3.884219
34	90,000	714	3565	60.103160	1650	7029	125.869400
35	90,000	10,000	39,998	1042.793000	4473	17,688	463.743300
36	90,000	60	297	2.058625	60	297	2.087188
37	90,000	694	2330	69.748030	677	2276	68.414340
38	90,000	2	7	0.204938	2	7	0.195375
39	90,000	2880	14,397	350.003800	2879	14,392	352.487200
40	90,000	491	1737	51.027900	622	2060	63.507870
41	90,000	54	215	0.649813	693	2719	8.442843
42	90,000	1169	5823	11.180380	3730	18,481	35.198530
43	90,000	1167	5808	64.462680	4541	22,497	315.836300
44	90,000	529	2098	61.044750	535	2115	61.438470
45	90,000	222	665	6.377344	1588	4688	45.179410
46	90,000	8	37	0.669438	8	37	0.678719
47	90,000	948	2843	19.813470	948	2843	20.081030
48	90,000	65	215	0.858969	646	1909	7.911000
49	90,000	2	7	0.202594	2	7	0.196938
50	90,000	1308	6537	160.226500	1310	6547	159.038700
51	90,000	7765	38,822	1003.051000	7765	38,822	1012.967000
52	90,000	498	1761	51.683120	608	2018	62.156810
53	90,000	2	7	0.202531	2	7	0.199563
54	90,000	7434	37,167	930.166400	7434	37,167	938.409400
55	90,000	2059	10,292	258.693500	2062	10,307	262.954400
56	90,000	1088	4350	11.823500	1088	4350	12.275250
57	90,000	2	6	0.014281	2	6	0.017813
58	90,000	23	68	1.075656	34	100	1.572375
59	90,000	528	1583	28.627440	528	1583	28.943250
60	90,000	1039	4153	144.438500	1088	4349	151.380900
61	90,000	2	7	0.188031	2	7	0.192125
62	90,000	65	215	0.843531	646	1909	7.941813
63	90,000	1	215	52.368780	1	1909	52.762500
64	90,000	10,000	49,830	90.154590	7332	36,324	69.138880
65	90,000	9386	46,763	84.491620	4622	22,904	44.597250
1	150,000	573	2836	86.975000	575	2844	87.901440
2	150,000	8	37	1.761812	22	107	5.196187
3	150,000	44	174	4.586094	47	186	4.939875
4	150,000	1849	7707	211.646100	2007	8265	228.498600
5	150,000	791	3071	28.159620	6042	23,961	204.553600
6	150,000	1193	5950	109.430100	3752	18,593	340.449900
7	150,000	243	1212	28.998030	244	1217	29.267750
8	150,000	195	584	8.379937	195	584	8.431313

Table 3 (*Continued*)

NO	Dim	Algorithm 2.1			Algorithm 1		
		NI	NFG	CPU	NI	NFG	CPU
9	150,000	12	28	0.549281	3396	6792	140.188300
10	150,000	182	725	13.544060	178	709	13.124880
11	150,000	33	122	4.384094	46	137	4.963875
12	150,000	185	553	12.357280	1304	3837	85.495120
13	150,000	5	15	0.375813	78	233	5.665125
14	150,000	5317	21,266	1558.335000	1518	6035	435.598100
15	150,000	652	2597	43.007870	1510	5062	83.542500
16	150,000	4	8	0.327438	4	8	0.331625
17	150,000	4185	16,736	310.876300	4038	16,148	302.085700
18	150,000	577	2282	234.017300	651	2501	250.321700
19	150,000	543	2108	115.348700	606	2297	126.085400
20	150,000	1856	7388	108.733000	1201	4573	69.361500
21	150,000	37	114	3.196656	462	1385	41.271310
22	150,000	54	214	4.151344	54	214	4.162875
23	150,000	7833	15,670	463.476000	7796	15,596	470.208800
24	150,000	70	347	6.909438	697	3370	67.468870
25	150,000	911	4185	149.822700	1271	5619	200.791100
26	150,000	1088	5415	89.901440	24	114	1.983250
27	150,000	2215	11,021	35.738090	3730	18,481	62.137870
28	150,000	741	3672	64.568810	741	3672	65.072250
29	150,000	889	4377	121.323800	889	4377	121.201100
30	150,000	981	4890	125.251900	3772	18,694	85,912.53
31	150,000	24	94	0.809406	24	94	0.824999
32	150,000	26	77	1.310812	26	77	1.307996
33	150,000	14	67	3.713187	14	67	3.747922
34	150,000	703	3511	100.417200	1388	6145	186.073000
35	150,000	10,000	39,998	1754.543000	4503	17,809	787.514800
36	150,000	38	187	2.184250	38	187	2.263187
37	150,000	707	2369	118.873700	690	2315	118.594500
38	150,000	2	7	0.325656	2	7	0.332813
39	150,000	2880	14,397	587.451400	2879	14,392	606.360800
40	150,000	10,000	30,267	1747.903000	672	2196	113.973500
41	150,000	184	735	3.663625	694	2722	13.838310
42	150,000	1358	6771	22.041560	3702	18,337	202.568000
43	150,000	1210	6037	111.168500	4543	22,507	1077.162000
44	150,000	530	2101	102.349000	535	2115	109.978700
45	150,000	274	819	13.210880	1690	4988	98.434780
46	150,000	7	32	0.966938	7	32	1.057281
47	150,000	973	2918	34.427190	973	2918	38.767780
48	150,000	65	215	1.448500	646	1909	15.874250
49	150,000	2	7	0.331188	2	7	0.354250
50	150,000	1308	6537	263.783100	1311	6552	293.491800
51	150,000	7765	38,822	1684.177000	7766	38,827	1981.202000
52	150,000	10,000	30,268	1756.903000	644	2118	119.279600
53	150,000	2	7	0.324875	2	7	0.425313
54	150,000	7434	37,167	1561.060000	7434	37,167	1800.328000
55	150,000	2059	10,292	446.907300	2062	10,307	487.069700
56	150,000	1088	4350	20.671000	1088	4350	25.482500
57	150,000	2	6	0.030125	2	6	0.043125
58	150,000	28	83	2.166875	34	100	2.942250
59	150,000	540	1619	49.252810	541	1622	54.262940
60	150,000	1086	4341	253.299800	1114	4453	281.304400
61	150,000	2	7	0.314938	2	7	0.324813
62	150,000	65	215	1.469437	646	1909	15.412690
63	150,000	1	215	145.548700	1	1909	159.423600
64	150,000	10,000	49,809	196.063200	7233	35,833	189.318800
65	150,000	8676	43,204	140.774700	4592	22,753	93.901630
1	210,000	573	2836	122.895800	575	2844	123.117700
2	210,000	53	262	17.625310	56	277	19.078840
3	210,000	44	174	6.490313	47	186	6.942719

Table 3 (*Continued*)

NO	Dim	Algorithm 2.1			Algorithm 1		
		NI	NFG	CPU	NI	NFG	CPU
4	210,000	2053	8524	329.735100	2007	8265	363.818300
5	210,000	839	3263	42.105750	6047	23,976	318.768000
6	210,000	1385	6911	176.616000	3778	18,725	481.052400
7	210,000	195	972	32.574000	196	977	32.512590
8	210,000	195	584	11.812690	195	584	11.700880
9	210,000	10	25	0.636500	3828	7656	216.963400
10	210,000	151	601	15.601060	149	593	16.178340
11	210,000	35	133	7.425500	46	137	6.971656
12	210,000	159	475	14.729620	1457	4286	133.270000
13	210,000	7	21	0.690063	78	233	7.877844
14	210,000	5403	21,610	2218.623000	1531	6085	613.377000
15	210,000	1027	4086	95.329560	1527	5113	117.547900
16	210,000	4	8	0.469438	4	8	0.469063
17	210,000	4244	16,972	440.685700	4103	16,408	427.691300
18	210,000	578	2286	327.911200	651	2501	349.330900
19	210,000	543	2108	161.378500	606	2297	11,790.320000
20	210,000	2097	8332	171.760500	1201	4573	97.047560
21	210,000	41	129	5.103688	470	1409	58.391690
22	210,000	54	214	5.743125	54	214	5.820063
23	210,000	7833	15,670	654.266300	7796	15,596	652.518700
24	210,000	400	1997	55.551690	1261	6156	171.682500
25	210,000	1088	5415	125.671300	24	114	2.806562
26	210,000	2035	10,133	46.097440	3732	18,491	107.095900
27	210,000	686	3427	84.321690	686	3427	85.084120
28	210,000	872	4292	163.879400	872	4292	163.689200
29	210,000	1207	6011	219.317600	4992	24,796	874.306100
30	210,000	24	94	1.136250	24	94	1.203375
31	210,000	26	77	1.813437	26	77	1.918250
32	210,000	10	47	3.683844	10	47	3.776062
33	210,000	638	3187	127.252500	641	3202	127.119400
34	210,000	28	137	2.273625	28	137	2.449063
35	210,000	715	2393	169.418200	698	2339	164.307800
36	210,000	2	7	0.448313	2	7	0.455063
37	210,000	2880	14,397	825.603000	2879	14,392	829.216300
38	210,000	283	1131	7.781998	695	2725	20.372810
39	210,000	1240	6188	28.252000	3597	17,819	209.757800
40	210,000	1128	5627	145.184000	4559	22,590	1239.831000
41	210,000	371	1109	25.005980	1735	5125	127.223700
42	210,000	20	97	4.087996	19	92	4.170031
43	210,000	990	2969	49.404980	990	2969	49.341160
44	210,000	65	215	2.028015	646	1909	19.142310
45	210,000	2	7	0.452000	2	7	0.453688
46	210,000	1308	6537	368.925000	1311	6552	458.029900
47	210,000	7765	38,822	2364.621000	7766	38,827	3018.524000
48	210,000	10,000	30,268	2460.779000	676	2206	166.936300
49	210,000	2	7	0.467848	2	7	0.466844
50	210,000	7434	37,167	2191.087000	7434	37,167	3023.795000
51	210,000	2059	10,292	608.572100	2062	10,307	653.233600
52	210,000	1088	4350	28.642380	1088	4350	38.929310
53	210,000	2	6	0.030805	2	6	0.047719
54	210,000	34	102	3.696555	34	100	4.255938
55	210,000	548	1643	69.857270	549	1646	78.052220
56	210,000	1106	4421	360.469800	1130	4517	1023.525000
57	210,000	2	7	0.453117	2	7	0.492094
58	210,000	65	215	2.011992	646	1909	21.231910
59	210,000	1	215	285.122300	1	1909	318.237500
60	210,000	10,000	49,791	317.897200	7278	36,056	876.884300
61	210,000	8113	40,386	201.801700	4592	22,753	227.298400

Appendix 2

Table 4 Test problems

No.	Problem
1	Generalization of MAXQ (convex)
2	Chained LQ (convex)
3	Number of active faces (nonconvex)
4	Nonsmooth generalization of Brown function (nonconvex)
5	Chained Mifflin 2 (nonconvex)
6	Chained crescent (nonconvex)
7	Chained crescent 2 (nonconvex)

Table 5 Numerical results

NO	Dim	Algorithm 3.1			Algorithm 2			Algorithm 3		
		NI	NFG	CPU	NI	NFG	CPU	NI	NFG	CPU
1	150,000	268	5592	27.595310	268	5592	27.707090	3	33	0.185438
2	150,000	67	173	1.497563	67	173	1.576625	137	312	2.091156
3	150,000	104	1808	24.912940	114	1836	25.536130	3	33	0.436469
4	150,000	75	185	20.279250	82	204	21.684970	137	313	35.800970
5	150,000	67	173	1.838000	67	173	1.873375	151	337	2.839188
6	150,000	55	205	1.859063	55	205	1.917375	137	312	2.385719
7	150,000	4	12	0.108688	4	12	0.108406	137	312	2.370000
1	180,000	271	5655	33.589060	271	5655	33.772090	3	33	0.218094
2	180,000	81	201	2.009625	81	201	2.059219	137	312	2.465594
3	180,000	109	1859	30.826380	116	1878	31.326120	3	33	0.483875
4	180,000	88	210	27.703250	93	225	28.768280	137	313	42.884500
5	180,000	81	201	2.465125	81	201	2.494750	151	337	3.402688
6	180,000	69	233	2.461375	69	233	2.510656	137	312	2.760719
7	180,000	5	15	0.172438	5	15	0.169938	137	312	2.776000
1	192,000	272	5676	36.939880	272	5676	37.251780	3	33	0.218531
2	192,000	72	176	1.889156	78	194	2.215563	137	312	2.744531
3	192,000	106	1850	34.069270	116	1878	35.289880	3	33	0.577375
4	192,000	90	216	30.327910	93	225	31.282190	137	313	45.879250
5	192,000	81	201	2.651781	81	201	2.855937	151	338	3.744344
6	192,000	69	233	2.651438	69	233	2.855469	137	312	3.040250
7	192,000	5	15	0.201094	5	15	0.187375	137	312	3.027188
1	210,000	273	5697	39.558750	273	5697	40.656190	3	33	0.249844
2	210,000	81	201	2.367750	81	201	2.449625	137	312	2.885844
3	210,000	109	1877	36.393310	117	1899	37.347190	3	33	0.578000
4	210,000	89	213	32.668690	96	232	34.631900	137	313	50.043880
5	210,000	81	201	2.836188	81	201	2.947094	151	338	3.995281
6	210,000	69	233	2.871250	69	233	2.946250	137	312	3.275187
7	210,000	5	15	0.198250	5	15	0.219938	137	312	3.246750
1	222,000	274	5718	41.885560	274	5718	42.759780	3	33	0.278094
2	222,000	70	170	2.013531	80	200	2.541594	137	312	3.026187
3	222,000	109	1877	38.548250	117	1899	39.467090	3	33	0.637906
4	222,000	96	232	37.098370	96	232	36.658720	137	313	53.057190
5	222,000	81	201	3.026719	81	201	3.106844	151	337	4.210594
6	222,000	69	233	3.041375	69	233	3.121375	137	312	3.478031
7	222,000	5	15	0.216406	5	15	0.218281	137	312	3.449469
1	231,000	275	5739	43.525590	275	5739	44.507000	3	33	0.265125
2	231,000	81	201	2.573875	79	197	2.604687	137	312	3.183219
3	231,000	118	1920	41.074940	118	1920	41.665530	3	33	0.625156
4	231,000	85	201	34.211090	96	232	38.190850	137	313	55.084160
5	231,000	81	201	3.151156	81	201	3.228437	151	337	4.366219
6	231,000	70	236	3.196813	70	236	3.324875	137	312	3.587406
7	231,000	4	12	0.173500	4	12	0.171656	137	312	3.573469

Table 5 (*Continued*)

NO	Dim	Algorithm 3.1			Algorithm 2			Algorithm 3		
		NI	NFG	CPU	NI	NFG	CPU	NI	NFG	CPU
1	240,000	275	5739	45.549500	275	5739	47.158840	3	33	0.279813
2	240,000	81	201	2.702625	81	201	2.777250	137	312	3.307688
3	240,000	107	1889	41.918190	118	1920	43.150280	3	33	0.655719
4	240,000	96	232	40.166310	96	232	39.578840	137	313	57.328840
5	240,000	81	201	3.278250	81	201	3.354719	151	337	4.555719
6	240,000	70	236	3.319375	70	236	3.431250	137	312	3.745031
7	240,000	4	12	0.171063	4	12	0.188563	137	312	3.727937
1	252,000	276	5760	47.984220	276	5760	51.422370	3	33	0.295813
2	252,000	72	176	2.386094	81	201	2.901656	137	312	3.430656
3	252,000	118	1920	44.880910	118	1920	45.349560	3	33	0.703563
4	252,000	96	232	42.087090	96	232	41.544910	137	313	60.076440
5	252,000	81	201	3.433156	81	201	3.526937	151	338	4.771313
6	252,000	70	236	3.478938	70	236	3.603594	137	312	3.901781
7	252,000	4	12	0.185469	4	12	0.185125	137	312	3.899437
1	270,000	277	5781	51.697340	277	5781	52.729160	3	33	0.327406
2	270,000	75	185	2.761375	80	200	3.089156	137	312	3.681875
3	270,000	110	1916	47.813940	119	1941	49.108690	3	33	0.733344
4	270,000	96	232	45.132440	96	232	44.506310	137	313	64.317340
5	270,000	81	201	3.681625	81	201	3.789500	151	337	5.117938
6	270,000	70	236	3.760937	70	236	3.869437	137	312	4.212750
7	270,000	4	12	0.202219	4	12	0.200344	137	312	4.197813

Acknowledgements
The authors would like to thank for the support funds.

Funding
This work was supported by the National Natural Science Foundation of China (Grant No. 11661009), the Guangxi Natural Science Fund for Distinguished Young Scholars (No. 2015GXNSFGA139001), and the Guangxi Natural Science Key Fund (No. 2017GXNSFDA198046).

Competing interests
The authors declare to have no competing interests.

Authors' contributions
GY mainly analyzed the theory results and organized this paper, TL did the numerical experiments of smooth problems and WH focused on the nonsmooth problems and image problems. All authors read and approved the final manuscript.

Publisher's Note

Received: 30 January 2019 Accepted: 27 August 2019 Published online: 18 September 2019

References

1. Al-Baali, A., Albaali, M.: Descent property and global convergence of the Fletcher–Reeves method with inexact line search. IMA J. Numer. Anal. **5**(1), 121–124 (1985)
2. Al-Baali, M., Narushima, Y., Yabe, H.: A family of three-term conjugate gradient methods with sufficient descent property for unconstrained optimization. Comput. Optim. Appl. **60**(1), 89–110 (2015)
3. Andrei, N.: An unconstrained optimization test functions collection. Environ. Sci. Technol. **10**(1), 6552–6558 (2008)
4. Andrei, N.: On three-term conjugate gradient algorithms for unconstrained optimization. Appl. Math. Comput. **241**(11), 19–29 (2008)
5. Argyros, I.K., George, S.: Local convergence analysis of Jarratt-type schemes for solving equations. Appl. Set-Valued Anal. Optim. **1**, 53–62 (2019)
6. Banham, M.R., Katsaggelos, A.K.: Digital image restoration. IEEE Signal Process. Mag. **14**, 24–41 (1997)
7. Cai, J.F., Chan, R.H., Fiore, C.D.: Minimization of a detail-preserving regularization functional for impulse noise removal. J. Math. Imaging Vis. **29**, 79–91 (2007)
8. Cai, J.F., Chan, R.H., Morini, B.: Minimization of an edge-preserving regularization functional by conjugate gradient types methods. In: Image Processing Based on Partial Differential Equations: Prceedings of the International Conference on PDE-Based Image Processsing and Related Inverse Problems, CMA, Oslo, August 8–12, 2005, pp. 109–122. Springer, Berlin (2007)
9. Cardenas, S.: Efficient generalized conjugate gradient algorithms. I. Theory. J. Optim. Theory Appl. **69**(1), 129–137 (1991)
10. Chan, C.L., Katsaggelos, A.K., Sahakian, A.V.: Image sequence filtering in quantum-limited noise with applications to low-dose fluoroscopy. IEEE Trans. Med. Imaging **12**, 610–621 (1993)

11. Dai, Z.: A mixed HS–DY conjugate gradient methods. Math. Numer. Sin. (2005)
12. Dai, Z., Wen, F.: A generalized approach to sparse and stable portfolio optimization problem. J. Ind. Manag. Optim. **14**, 1651–1666 (2018)
13. Dai, Z.F.: Two modified HS type conjugate gradient methods for unconstrained optimization problems. Nonlinear Anal., Theory Methods Appl. **74**(3), 927–936 (2011)
14. Deng, S., Wan, Z.: A three-term conjugate gradient algorithm for large-scale unconstrained optimization problems. Appl. Numer. Math. **92**, 70–81 (2015)
15. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2001)
16. Du, X., Liu, J.: Global convergence of a spectral HS conjugate gradient method. Proc. Eng. **15**(4), 1487–1492 (2011)
17. Gilbert, J.C., Nocedal, J.: Global convergence properties of conjugate gradient methods for optimization. SIAM J. Optim. **2**(1), 21–42 (1990)
18. Haarala, M., Miettinen, K., Mäkelä, M.M.: New limited memory bundle method for large-scale nonsmooth optimization. Optim. Methods Softw. **19**(6), 673–692 (2004)
19. Haarala, N., Miettinen, K., Mäkelä, M.M.: Globally convergent limited memory bundle method for large-scale nonsmooth optimization. Math. Program. **109**(1), 181–205 (2007)
20. Lin, C.J., Weng, R.C., Keerthi, S.S.: Trust region Newton method for logistic regression. J. Mach. Learn. Res. **9**(2), 627–650 (2008)
21. Meng, F., Zhao, G.: On second-order properties of the Moreau–Yosida regularization for constrained nonsmooth convex programs. Numer. Funct. Anal. Optim. **25**(5–6), 515–529 (2004)
22. Narushima, Y., Yabe, H., Ford, J.A.: A three-term conjugate gradient method with sufficient descent property for unconstrained optimization. SIAM J. Optim. **21**(1), 212–230 (2016)
23. Nazareth, L.: A conjugate direction algorithm without line searches. J. Optim. Theory Appl. **23**(3), 373–387 (1977)
24. Oustry, F.: A second-order bundle method to minimize the maximum eigenvalue function. Math. Program. **89**(1), 1–33 (2000)
25. Pearson, J.W., Martin, S., Wathen, A.J.: Preconditioners for state-constrained optimal control problems with Moreau–Yosida penalty function. Numer. Linear Algebra Appl. **21**(1), 81–97 (2013)
26. Polak, E.: The conjugate gradient method in extreme problems. Comput. Math. Math. Phys. **9**, 94–112 (1969)
27. Polak, E., Ribière, G.: Note sur la convergence de directions conjugees. Rev. Fr. Inform. Rech. Opér. **3**, 35–43 (1969)
28. Schramm, H., Zowe, J.: A version of the bundle method for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. SIAM J. Optim. **2**(1), 121–152 (2006)
29. Sheng, Z., Yuan, G.: An effective adaptive trust region algorithm for nonsmooth minimization. Comput. Optim. Appl. **71**, 251–271 (2018)
30. Sheng, Z., Yuan, G., et al.: An adaptive trust region algorithm for large-residual nonsmooth least squares problems. J. Ind. Manag. Optim. **14**, 707–718 (2018)
31. Sheng, Z., Yuan, G., Cui, Z.: A new adaptive trust region algorithm for optimization problems. Acta Math. Sci. **38**(2), 479–496 (2018)
32. Slump, C.H.: Real-time image restoration in diagnostic X-ray imaging, the effects on quantum noise. In: Proceedings of the 11th IAPR International Conference on Pattern Recognition, Vol. II, Conference B: Pattern Recognition Methodology and Systems, pp. 693–696 (1992)
33. Sun, Q., Liu, Q.: Global convergence of modified HS conjugate gradient method. J. Appl. Math. Comput. **22**(3), 289–297 (2009)
34. Touati-Ahmed, D., Storey, C.: Efficient hybrid conjugate gradient techniques. J. Optim. Theory Appl. **64**(2), 379–397 (1990)
35. Wan, Z., Yang, Z.L. Wang, Y.L.: New spectral PRP conjugate gradient method for unconstrained optimization. Appl. Math. Lett. **24**(1), 16–22 (2011)
36. Wang, W., Qiao, X., Han, Y.: A proximal bundle method for nonsmooth and nonconvex constrained optimization. Comput. Stat. Data Anal. **34**(34), 3464–3485 (2011)
37. Wei, Z., Li, G., Qi, L.: New quasi-Newton methods for unconstrained optimization problems. Appl. Math. Comput. **175**(2), 1156–1188 (2006)
38. Wei, Z., Yao, S., Liu, L.: The convergence properties of some new conjugate gradient methods. Appl. Math. Comput. **183**(2), 1341–1350 (2006)
39. Wei, Z., Yu, G., Yuan, G., et al.: The superlinear convergence of a modified BFGS-type method for unconstrained optimization. Comput. Optim. Appl. **29**(3), 315–332 (2004)
40. Ying, L., Hai, Z., Fei, L.: A modified proximal gradient method for a family of nonsmooth convex optimization problems. J. Oper. Res. Soc. China **5**, 391–403 (2017)
41. Yuan, G., Hu, W., Wang, B.: A modified Armijo line search technique for large-scale nonconvex smooth and convex nonsmooth optimization problems. Preprint (2017)
42. Yuan, G., Li, Y., Li, Y.: A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimizations and nonlinear equations. J. Optim. Theory Appl. **168**(1), 129–152 (2016)
43. Yuan, G., Lu, X.: A modified PRP conjugate gradient method. Ann. Oper. Res. **166**(1), 73–90 (2009)
44. Yuan, G., Lu, X., Wei, Z.: A conjugate gradient method with descent direction for unconstrained optimization. J. Comput. Appl. Math. **233**(2), 519–530 (2009)
45. Yuan, G., Sheng, Z., Wang, P., Hu, W., Li, C.: The global convergence of a modified BFGS method for nonconvex functions. J. Comput. Appl. Math. **327**, 274–294 (2018)
46. Yuan, G., Wei, Z.: Convergence analysis of a modified BFGS method on convex minimizations. Comput. Optim. Appl. **47**(2), 237–255 (2010)
47. Yuan, G., Wei, Z., Li, G.: A modified Polak–Ribière–Polyak conjugate gradient algorithm for nonsmooth convex programs. J. Comput. Appl. Math. **255**, 86–96 (2014)
48. Yuan, G., Wei, Z., Lu, X.: Global convergence of BFGS and PRP methods under a modified weak Wolfe–Powell line search. Appl. Math. Model. **47**, 811–825 (2017)
49. Yuan, G., Zhang, M.: A three-terms Polak–Ribière–Polyak conjugate gradient algorithm for large-scale nonlinear equations. J. Comput. Appl. Math. **286**, 186–195 (2015)

50. Zaslavski, A.J.: Three convergence results for continuous descent methods with a convex objective function. J. Appl. Numer. Optim. **1**, 53–61 (2019)
51. Zhang, L., Zhou, W., Li, D.H.: A descent modified Polak–Ribière–Polyak conjugate gradient method and its global convergence. IMA J. Numer. Anal. **26**(4), 629–640 (2006)
52. Zhao, X., Ng, K.F., Li, C., Yao, J.C.: Linear regularity and linear convergence of projection-based methods for solving convex feasibility problems. Appl. Math. Optim. **78**, 613–641 (2018)
53. Zhou, W.: Some descent three-term conjugate gradient methods and their global convergence. Optim. Methods Softw. **22**(4), 697–711 (2007)