Global Convergence of a Modified Two-Parameter Scaled BFGS Method with Yuan-Wei-Lu Line Search for Unconstrained Optimization

Pengyuan Li, Zhan Wang, Dan Luo, and Hongtruong Pham

1College of Mathematics and Information Science, Guangxi University, Nanning, Guangxi, China
2School of Mathematics and Statistics, Baise University, Baise, Guangxi, China
3Thai Nguyen University of Economics and Business Administration, Thai Nguyen, Vietnam

Correspondence should be addressed to Dan Luo; bsxyldan@163.com

Received 23 July 2020; Accepted 5 August 2020; Published 26 August 2020

The BFGS method is one of the most efficient quasi-Newton methods for solving small- and medium-size unconstrained optimization problems. For the sake of exploring its more interesting properties, a modified two-parameter scaled BFGS method is stated in this paper. The intention of the modified scaled BFGS method is to improve the eigenvalues structure of the BFGS update. In this method, the first two terms and the last term of the standard BFGS update formula are scaled with two different positive parameters, and the new value of $y_k$ is given. Meanwhile, Yuan-Wei-Lu line search is also proposed. Under the mentioned line search, the modified two-parameter scaled BFGS method is globally convergent for nonconvex functions. The extensive numerical experiments show that this form of the scaled BFGS method outperforms the standard BFGS method or some similar scaled methods.

1. Introduction

Consider

$$\min f(x),$$

(1)

where $x \in \mathbb{R}^n$, and $f: \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function bounded from below. The quasi-Newton methods are currently used in countless optimization software for solving unconstrained optimization problems [1–8]. The BFGS method, one of the most efficient quasi-Newton methods, for solving (1) is an iterative method of the following form:

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

(2)

where $k = 0, 1, 2, \ldots, \alpha_k$, obtained by some line search rule, is a step size, and $d_k$ is the BFGS search direction computed by the following equation:

$$B_k d_k = -g_k,$$

(3)

where $g_k = g(x_k)$ is the gradient of $f(x)$, and the matrix $B_k$ is the BFGS approximation to the Hessian $\nabla^2 f(x_k)$, which has the following update formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$

(4)

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. The problems related to the BFGS method have been analyzed and studied by many scholars, and satisfactory conclusions have been drawn [9–16]. In earlier year, Powell [17] first proved the global convergence of the standard BFGS method with inexact Wolfe line search for convex functions. Under the exact line search or some specific inexact line search, the BFGS method has the convergence property for convex minimization problems [18–21]. By contrast, for nonconvex problems, Mascaren [22] has presented an example to elaborate that the BFGS method and some Broyden-type methods may not be convergent under the exact line search. As such, with the Wolfe line searches, Dai [23] also proved that the BFGS method may fail to converge. To verify the global convergence of the BFGS method for general functions and to obtain a better Hessian approximation matrix of the objective function, Yuan and Wei [24] presented a modified quasi-Newton equation as follows:
where
\[ B_{k+1} s_k = \tilde{y}_k, \]  
(5)

with
\[ \tilde{y}_k = y_k + \max\{ C_k, 0 \} \frac{1}{s_k^T s_k} s_k, \]
\[ C_k = 2[ f(x_k) - f(x_k + \alpha_k d_k)] + (g(x_k + \alpha_k d_k) + g(x_k))^T s_k, \]
(6)

\[ B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{\tilde{y}_k \tilde{y}_k^T}{s_k^T s_k} \]  
(7)

In practice, the standard BFGS method has many qualities worth exploring and can effectively solve a class of unconstrained optimization problems.

Here, two excellent properties of the BFGS method are introduced. One is the self-correcting quality, scilicet; if the current Hessian approximate inverse matrix estimates the curvature of the function incorrectly, then Hessian approximation matrix \( H_k \) will correct itself within a few steps. The other interesting property is that small eigenvalues are better corrected than large ones [25]. Hence, one can see other interesting property is that small eigenvalues are better corrected than large ones [25].

Formula 1. The general one-parameter scaled BFGS updating formula is
\[ B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T s_k}, \]  
(8)

where \( y_k \) is a positive parameter, and it is diverse for the selection of the scaled factor \( \gamma_k \), which is listed as follows.

Choice A:
\[ \gamma_k = \frac{2}{y_k s_k} \left( f(x_k) - f(x_{k+1}) + s^T g_{k+1} \right), \]  
(9)

where the value of \( \gamma_k \) is given by Yuan [29], and with inexact line search, the global convergence of the scaled BFGS method with \( \gamma_k \) given by (9) is established for convex functions by Powell [30]. Alternatively, for general nonlinear functions, Yuan limited the value range of \( \gamma_k \) to [0.01, 100] to ensure the positivity of \( \gamma_k \) under the inexact line search and proved the global convergence of the scaled BFGS method in this form.

Choice B:
\[ \gamma_k = \frac{y_k^T s_k}{\| y_k \|}, \]  
(10)

which is obtained as a solution of the problem: \( \min \| s_k - y_k y_k^T \| \). The scaled BFGS method based on this value of \( \gamma_k \) was introduced by Barzilai and Borwein [31] and was deemed the spectral scaled BFGS method. Cheng and Li [32] proved that the spectral scaled BFGS method is globally convergent under Wolfe line search with assuming the convexity of the minimizing function.

Choice C:
\[ \gamma_k = \min \left\{ \frac{y_k^T s_k}{\| y_k \|^2 + \beta_k}, 1 \right\}, \]  
(11)

where \( \beta_k > 0 \) for \( k = 0, 1, \ldots \). Under the Wolfe line search (20) and (21), \( y_k^T s_k > 0 \) holds for \( k = 0, 1, \ldots \), which implies that \( \gamma_k \) computed by (11) is bounded away from zero, that is to say, \( 0 < \gamma_k \leq 1 \). Therefore, in this instance, the large eigenvalues of \( B_{k+1} \) given by (8) are shifted to the left [33].

Formula 2. Proposed by Oren and Luenberger [26], this scaled BFGS method was the single parameter scaled of the first two items of the BFGS update and was defined as
\[ B_{k+1} = \delta_k \left[ B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \right] + \frac{y_k y_k^T}{y_k^T s_k} \]  
(12)

where \( \delta_k \) is a positive parameter and is calculated as follows:
\[ \delta_k = \frac{y_k^T s_k}{s_k^T B_k s_k} \]  
(13)

The parameter \( \delta_k \) assigned by (13) can make the structure of eigenvalue to inverse Hessian approximation more easily analyzed. Consequently, it is regarded as one of the best factors.

Formula 3. In this method, the scaled parameters are selected to cluster the eigenvalues of the iteration matrix \( B_{k+1} \) and shift the large eigenvalues to the left. The update formula of the Hessian approximate matrix is computed as
\[ B_{k+1} = \delta_k \left[ B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \right] + \frac{y_k y_k^T}{y_k^T s_k} \]  
(14)

where both \( \delta_k \) and \( \gamma_k \) are positive parameters, and Andrei [34] preset them as the following values:
\[ \gamma_k = \min \left\{ \frac{y_k^T s_k}{\| y_k \| + \| s^T g_{k+1} \|^2}, 1 \right\}, \]  
(15)

\[ \delta_k = \frac{n - y_k \left( \| y_k \|^2 / y_k^T s_k \right)}{n - \left( \| B_k s_k \|^2 / s_k^T B_k s_k \right)}, \]  
(16)

If the scaled parameters are bounded and line search is inexact, then this scaled BFGS algorithm is globally convergent for general functions. A large number of numerical experiments show that the double parameter scaled BFGS
method with \(\delta_k\) and \(y_k\) given by (15) and (16) is more competitive than the standard BFGS method. In this paper, combining (7) and (14), we propose a new update formula of \(B_{k+1}\) listed as follows:

\[
B_{k+1} = \delta_k \left[ B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} \right] + \frac{\gamma_k y_k y_k^T}{\gamma_k y_k^T s_k},
\]

(17)

where \(\gamma_k\) is determined by formula (6),

\[
y_k = \frac{\gamma_k y_k}{\|y_k\|^2 + \|s_k\|^2},
\]

(18)

\[
\delta_k = \frac{n - y_k \left(\|y_k\|^2/\|s_k\|^2, \gamma_k s_k\right)}{n - \left(\|B_k s_k\|^2/\|s_k\|^2, B_k s_k\right)},
\]

(19)

Some interesting properties of the BFGS-type method are inseparable from the weak Wolfe–Powell (WWP) line search:

\[
f(x_k + \alpha_k d_k) \leq f_k + \xi \alpha_k g_k^T d_k,
\]

(20)

\[
g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k,
\]

(21)

where \(0 < \xi < \sigma < 1\). There are many research studies based on this line search [35–43]. To further develop the inexact line search, Yuan et al. present a new line search and call it Yuan-Wei-Lu (YWL) line search, which has the following form:

\[
f(x_k + \alpha_k d_k) \leq f_k + \xi \alpha_k g_k^T d_k
\]

\[
+ \alpha_k \min \left[ -\xi_1 g_k^T d_k - \xi_2 s_k^T / 2 \|d_k\|^2 \right],
\]

(22)

\[
g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k + \min \left[ -\xi_1 g_k^T d_k, \xi_0 \|d_k\|^2 \right],
\]

(23)

where \(\xi \in (0, (1/2)), \xi_1 \in (0, \xi), \) and \(\sigma \in (\xi^{-1}, 1).\) The main contribution of this paper is to verify the global convergence of the modified scaled BFGS update (17) with \(\gamma_k\) and \(\delta_k\) given by (18) and (19), respectively, under this line search. Abundant numerical results show that such a combination is appropriate for nonconvex functions.

Our paper is organized as follows. The motivation and algorithm are introduced in the next section. In Section 3, the convergence analysis of the modified two-parameter scaled BFGS method under Yuan-Wei-Lu line search is established. Section 4 is devoted to show the results of numerical experiments. Some conclusions are stated in the last section.

2. Motivation and Algorithm

Two crucial tools for analyzing properties of the BFGS method are the trace and the determinant of the \(B_{k+1}\) given by (4). Thus, the corresponding relations are enumerated as follows:

\[
\det(B_{k+1}) = \det \left( B_k \left( I - \frac{s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{B_k^{-1} y_k y_k^T}{\gamma_k y_k^T s_k} \right) \right),
\]

(24)

\[
\text{tr}(B_{k+1}) = \text{tr}(B_k) - \frac{\|y_k\|^2}{\gamma_k y_k^T s_k},
\]

(25)

Applying the following existing relation in the study of Sun and Yuan [44],

\[
\det(I + v_1 v_1^T + v_2 v_2^T) = (1 + v_1^T v_2) (1 + v_2^T v_1) - (v_1^T v_2)^2,
\]

(26)

where \(v_1 = -s_k, v_2 = (B_k s_k / s_k^T B_k s_k), v_3 = B_k^{-1} y_k, \) and \(v_4 = (y_k / y_k^T s_k);\) we obtain

\[
\det(B_{k+1}) = \det(B_k) \frac{y_k^T s_k}{y_k^T s_k} \frac{s_k^T B_k s_k}{s_k^T B_k s_k},
\]

(27)

Obviously, the efficiency of the BFGS method depends on the eigenvalues structure of the Hessian approximation matrix, and the BFGS method is actually more affected by large eigenvalues than by small eigenvalues [25, 45, 46]. It can be seen that the second item on the right side of the formula (25) is negative. Therefore, it produces a shift of the eigenvalues of \(B_{k+1}\) to the left. Thus, the BFGS method can modify large eigenvalues. Moreover, the third term on the right hand side of (25) being positive produces a shift of the eigenvalues of \(B_{k+1}\) to the right. If this term is large, \(B_{k+1}\) may have large eigenvalues too. Therefore, the eigenvalues of the \(B_{k+1}\) can be corrected by scaling the corresponding items in (25), which is the main motivation for us to use the scaling BFGS method. In this paper, we scale the first two terms and the last term of the standard BFGS update formula with two different positive parameters and propose a new \(\gamma_k\). In subsequent proof, we will propose some lemmas based on these two important tools to analyze the convergence of the modified scaled BFGS method. Then, an algorithm framework for solving the problem (1) will be built in Algorithm 1, which can be designed as

3. Convergence Analysis

In Section 3, the global convergence of Algorithm 1 will be established, and the following assumptions are useful in convergence analysis.

Assumption 1

(i) The level set \(Y = \{ x \in \mathbb{R}^n \mid f(x) \leq f(x_0) \}\) is bounded

(ii) The function \(f(x)\) is twice continuously differentiable and bounded from below

Lemma 1. If \(B_k\) is the positive definite, \(y_k > 0, \) and if \(\alpha_k\) is computed by (22) and (23), then \(B_{k+1}\) given by (17) is an equally positive definite for all \(k.\)
Proof. The inequality (22) and (23) indicates that \( y_k^T s_k > 0 \). Using the definition of \( y_k \), we obtain
\[
\gamma_k s_k = y_k^T s_k + \max \{ C_k, 0 \} \geq y_k^T s_k > 0. \tag{28}
\]
For any \( c \neq 0 \),
\[
c^T B_{k+1} c = \delta_k c^T B_k c - \delta_k \frac{c^T B_k s_k y_k^T c}{y_k^T s_k} + y_k c^T \gamma_k s_k \\
= \delta_k c^T B_k c - \delta_k \frac{(c^T B_k s_k)^2}{s_k^T B_k s_k} + y_k \left( \frac{c^T \gamma_k}{y_k^T s_k} \right)^2 \tag{29}
\]
\[
\geq \gamma_k \left( \frac{c^T \gamma_k}{y_k^T s_k} \right)^2 > 0,
\]
where the penultimate inequality follows, and
\[
(s_k^T B_k c)^2 \leq \left( y_k^T s_k \right) \left( c^T B_k c \right), \quad \text{for } z \neq 0, \tag{30}
\]
which is obtained by the Cauchy–Schwarz inequality. \( \square \)

**Lemma 2.** Let \( \delta_k \) be generated by (16) for \( k = 0, 1, \ldots \), then \( \delta_k > 0 \) and inlines to 1.

Proof. Observe the formula (19); after substituting \( \delta_k \), we can find that \( (\| y_k \|^2 / (\| y_k \| + \| s_k \|)) \) is close to 1. Owing to the symmetry, positive definiteness, and nonsingularity of \( B_{k+1} \), its eigenvalues \( \lambda_1, \ldots, \lambda_n \) is real and positive, and \( \text{tr} (B_k) = n \).

Hence, for \( i = 1, \ldots, n, \lambda_i > 0 \) and \( \sum_{i=1}^n \lambda_i = n \). Since
\[
\| B_0 s_0 \|^2 = s_0^T B_0 s_0, \quad 0 < \| B_k s_k \|^2 < 1, \quad \text{and} \quad 0 < s_k^T B_k s_k < 1
\]
for sufficiently large \( k \), \( \| B_k s_k \|^2 \) and \( s_k^T B_k s_k \) are roughly of the same order of magnitude, which shows that \( (\| B_k s_k \|^2 / s_k^T B_k s_k) \approx n \). To sum up, the relations \((\| y_k \|^2 / y_k^T s_k) \approx n \) and \((\| B_k s_k \|^2 / s_k^T B_k s_k) \approx n \) are valid, namely for \( k = 0, 1, \ldots \), \( \delta_k > 0 \) and inlines to 1. The proof is completed. \( \square \)

**Remark 1.** Based on the conclusion of lemma, we can infer that for any integer \( j \in [0, k] \), there exist two positive constants \( 0 < \delta < \Lambda \) satisfying \( \delta < \delta_k \delta_{k-1}, \ldots, \delta_j < \Lambda \).

**Lemma 3.** If \( B_{k+1} \) is updated by (14), where \( y_k \) and \( \delta_k \) are determined by (18) and (16), then
\[
\text{tr} (B_{k+1}) \leq \Lambda \text{tr} (B_0) + \Lambda k + \frac{1}{2} \tag{31}
\]
\[
\sum_{i=0}^k \frac{\| B_i s_i \|^2}{s_i^T B_i s_i} \leq \frac{\Lambda}{\delta} \left( \text{tr} (B_0) + k \right) + \frac{1}{2\delta} \tag{32}
\]
Proof. Considering (25), we have
\[
\text{tr} (B_{k+1}) = \delta_k \text{tr} (B_k) - \delta_k \frac{\| B_k s_k \|^2}{s_k^T B_k s_k} + \gamma_k \frac{y_k^T \gamma_k}{y_k^T s_k} \\
= \delta_k \left( \delta_k \text{tr} (B_{k-1}) - \delta_k \frac{\| B_{k-1} s_{k-1} \|^2}{s_{k-1}^T B_{k-1} s_{k-1}} + \gamma_{k-1} \frac{y_{k-1}^T \gamma_k}{y_{k-1}^T s_{k-1}} \right) - \delta_k \frac{\| B_k s_k \|^2}{s_k^T B_k s_k} + \gamma_k \frac{y_k^T \gamma_k}{y_k^T s_k} \\
= \ldots \\
= \delta_k \delta_{k-1} \ldots \delta_0 \text{tr} (B_0) - \delta_k \delta_{k-1} \ldots \delta_0 \frac{\| B_0 s_0 \|^2}{s_0^T B_0 s_0} + \delta_k \delta_{k-1} \ldots \delta_1 \frac{\| y_0 \|^2}{y_0^T s_0} \\
- \ldots - \delta_k \delta_{k-1} \ldots \delta_1 \frac{\| B_{k-1} s_{k-1} \|^2}{s_{k-1}^T B_{k-1} s_{k-1}} + \delta_k \delta_{k-1} \ldots \delta_1 \frac{y_{k-1}^T \gamma_k}{y_{k-1}^T s_{k-1}} \\
- \ldots - \delta_k \delta_{k-1} \ldots \delta_1 \frac{\| B_k s_k \|^2}{s_k^T B_k s_k} + \gamma_k \frac{y_k^T \gamma_k}{y_k^T s_k} \\
(33)
In addition,

\[ \frac{\|\mathbf{y}\|^2}{\|\mathbf{y}\|} = \frac{\|\mathbf{y}\|^2}{\|\mathbf{y}\|^2 + \|\mathbf{s}\|^2} \leq \frac{1}{2}. \] (34)

Therefore, by Remark 1 and the above inequality, the formula (33) is transformed into

\[ \text{tr}(B_{k+1}) \leq \Lambda \text{tr}(B_0) - \sum_{i=0}^{k} \delta_i \|B_{k+i}\| + \sum_{j=1}^{k} \Lambda + \frac{1}{2} \leq \Lambda \text{tr}(B_0) + \Lambda k + \frac{1}{2}. \] (35)

which implies (31). From the positive definiteness of \( B_{k+1} \), (32) also holds. The proof is completed. \( \square \)

**Lemma 4.** Consider \( \gamma_k \geq s \) and \( \delta_k \geq q \) for all \( k \), where \( s \) and \( t \) are constants. Then, there exists a positive constant \( h \) such that

\[ \prod_{i=0}^{k} \alpha_i \geq h^k, \] (36)

for all \( k \) sufficiently large.

**Proof.** Utilizing the identity (26) and taking the determinant on both sides of the formula (14) with \( \gamma_k \) and \( \delta_k \) computed as in (18) and (16), we have

\[ \det(B_{k+1}) = \det(\delta_k B_k) \left( 1 - \frac{s_k T B_k s_k}{s_k T B_k s_k} + \frac{\mathbf{y}_k B_k^{-1} \mathbf{y}_k^T}{\delta_k} \right) \]

\[ = \delta_k \det(B_k) \left( \frac{\mathbf{y}_k T \mathbf{y}_k}{\delta_k T \mathbf{y}_k} \right) \]

\[ = \delta_k \det(B_k) \left( \frac{\mathbf{y}_k T \mathbf{y}_k}{\delta_k T \mathbf{y}_k} \right) \]

\[ = \delta_k \det(B_k) \left( \frac{\mathbf{y}_k T \mathbf{y}_k}{\delta_k T \mathbf{y}_k} \right) \]

\[ \geq \det(B_0) \prod_{i=0}^{k} \delta_i \left( 1 - \frac{s_i T B_i s_i}{\alpha_i} \right) \]

\[ \geq \det(B_0) \prod_{i=0}^{k} \frac{s_i T B_i s_i}{\alpha_i} \]

where the penultimate inequality follows \( s_i T B_i s_i = -\alpha_i s_i T g_i \), \( \mathbf{y}_k T \mathbf{y}_k \geq - (1 - \sigma) s_i T g_i \), \( \gamma_k \geq s \), and \( \delta_i \geq q \) for all \( i \). Furthermore, by \( \det(B_{k+1}) \leq (1/n) \text{tr}(B_{k+1}) \) and Lemma 4, we obtain

\[ \det(B_{k+1}) \leq \left( \frac{1}{n} \left( \Lambda \text{tr}(B_0) + \Lambda k + \frac{1}{2} \right) \right)^n. \] (38)

Therefore,

\[ \prod_{i=0}^{k} \alpha_i \geq \frac{\det(B_0) q^{(n-1)(k+1)} s_k T B_k s_k}{(1/n) \left( \Lambda \text{tr}(B_0) + \Lambda k + (1/2) \right)^n}. \] (39)

Suppose \( k \) is sufficiently large, (39) implies (36). The proof is completed. \( \square \)

**Theorem 1.** If the sequence \( \{x_k\} \) is obtained by Algorithm 1, then

\[ \lim_{k \to \infty} \inf \|g_k\| = 0. \] (40)

**Proof.** The proof by contradiction is used to prove (40) holds. Suppose that \( \|g_k\| > F > 0 \). By Yuan-Wei-Liu line search (22) and \( f(x) \) bounded below, we obtain

\[ f_k - f(x_k + \alpha_k d_k) \geq -\xi_k g_k T d_k - \alpha_k \min \left( -\xi_k g_k T d_k, \frac{\xi_k}{2} \|d_k\|^2 \right) \]

\[ \geq -\xi \alpha_k g_k T d_k. \] (41)

Adding the abovementioned inequalities from \( k = 0 \) to \( \infty \) and utilizing Assumption 1 (ii), we have

\[ \sum_{k=0}^{\infty} (-s_k T g_k) < \infty. \] (42)

From Assumption 1 (ii) and (42), we have

\[ \alpha_k \sum_{k=0}^{\infty} (-s_k T B_k s_k) = \sum_{k=0}^{\infty} \alpha_k T B_k s_k \]

\[ = \sum_{k=0}^{\infty} \|g_k\| T B_k s_k \]

\[ = \sum_{k=0}^{\infty} \|g_k\| T B_k s_k \]

\[ = \sum_{k=0}^{\infty} \|g_k\| T B_k s_k \]

\[ \geq F^2 \sum_{k=0}^{\infty} \alpha_k T B_k s_k \]

\[ \geq F^2 \sum_{k=0}^{\infty} \alpha_k T B_k s_k \]

Based on this, given a constant \( \Phi > 0 \), there is a positive integer \( k_0 > 0 \) satisfying
Table 1: The test problems.

| No. | Test problem                                      |
|-----|--------------------------------------------------|
| 1   | Extended Freudenstein and Roth function          |
| 2   | Extended trigonometric function                  |
| 3   | Extended Rosenbrock function                     |
| 4   | Extended White and Holst function                |
| 5   | Extended Beale function                          |
| 6   | Extended penalty function                        |
| 7   | Perturbed quadratic function                      |
| 8   | Raydan 1 function                                 |
| 9   | Raydan 2 function                                 |
| 10  | Diagonal 1 function                               |
| 11  | Diagonal 2 function                               |
| 12  | Diagonal 3 function                               |
| 13  | Hager function                                   |
| 14  | Generalized tridiagonal 1 function                |
| 15  | Extended tridiagonal 1 function                   |
| 16  | Generalized tridiagonal 2 function                |
| 17  | Diagonal 4 function                               |
| 18  | Diagonal 5 function                               |
| 19  | Extended Himmelblau function                      |
| 20  | Generalized PSCI function                         |
| 21  | Extended PSCI function                            |
| 22  | Extended Powell function                          |
| 23  | Extended block diagonal BD1 function              |
| 24  | Extended Maratos function                         |
| 25  | Extended Cliff function                           |
| 26  | Quadratic diagonal perturbed function             |
| 27  | Extended Wood function                            |
| 28  | Extended Hiebert function                         |
| 29  | Quadratic function QP1 function                   |
| 30  | Quadratic function QP2 function                   |
| 31  | A quadratic function QF2 function                 |
| 32  | Extended EP1 function                             |
| 33  | Extended tridiagonal 2 function                   |
| 34  | BDQRTIC function (CUTE)                           |
| 35  | TRIDIA function (CUTE)                            |
| 36  | ARWHEAD function (CUTE)                           |
| 37  | NONDIA function (CUTE)                            |
| 38  | NONDQUAR function (CUTE)                          |
| 39  | DQDRTIC function (CUTE)                           |
| 40  | EG2 function (CUTE)                               |
| 41  | DIXMAANA function (CUTE)                          |
| 42  | DIXMAANB function (CUTE)                          |
| 43  | DIXMAANC function (CUTE)                          |
| 44  | DIXMAANE function (CUTE)                          |
| 45  | Partial perturbed quadratic function              |
| 46  | Broyden tridiagonal function                      |
| 47  | Almost perturbed quadratic function               |
| 48  | Tridiagonal perturbed quadratic function          |
| 49  | EDENSCH function (CUTE)                           |
| 50  | VARDIM function (CUTE)                            |
| 51  | STAIRCASE S1 function                             |
| 52  | LIARWHD function (CUTE)                           |
| 53  | DIAGONAL 6 function                               |
| 54  | DIXON3DQ function (CUTE)                          |
| 55  | DIXMAANF function (CUTE)                          |
| 56  | DIXMAANG function (CUTE)                          |
| 57  | DIXMAANH function (CUTE)                          |
| 58  | DIXMAANI function (CUTE)                          |

Table 1: Continued.

| No. | Test problem                                      |
|-----|--------------------------------------------------|
| 59  | DIXMAAND function (CUTE)                          |
| 60  | DIXMAANL function (CUTE)                          |
| 61  | DIXMAAN function (CUTE)                           |
| 62  | DIXMAANK function (CUTE)                          |
| 63  | DIXMAANL function (CUTE)                          |
| 64  | DIXMAAND function (CUTE)                          |
| 65  | ENGVAL1 function (CUTE)                           |
| 66  | FLETCHCR function (CUTE)                          |
| 67  | COSINE function (CUTE)                            |
| 68  | Extended DENSCHNB function (CUTE)                 |
| 69  | Extended DENSCHNF function (CUTE)                 |
| 70  | SINQUAD function (CUTE)                           |
| 71  | BIGGSB1 function (CUTE)                           |
| 72  | Partial perturbed quadratic PPQ2 function         |
| 73  | Scaled quadratic SQ1 function                     |
| 74  | Scaled quadratic SQ2 function                     |

\[
\begin{align*}
\mathbf{s} & = \mathbf{B}_k \mathbf{B}_k^T \mathbf{s}_k \\
\mathbf{s}^T \mathbf{B}_k \mathbf{s}_k & \leq \Phi,
\end{align*}
\]

where \( c \) is any positive integer, and the first inequality follows the geometric inequality. Moreover, by Lemma 4, we obtain

\[
\begin{align*}
\mathbf{s} & = \mathbf{B}_k \mathbf{B}_k^T \mathbf{s}_k \\
\mathbf{s}^T \mathbf{B}_k \mathbf{s}_k & \leq \frac{\Phi}{c} \sum_{k=0}^{\infty} \mathbf{s}^T \mathbf{B}_k \mathbf{s}_k^{(1/c)} \\
& \leq \frac{\Phi}{c^2} \left( \text{Attr}(\mathbf{B}_0) + \Lambda (\mathbf{B}_0 + \mathbf{c}) + \frac{1}{2} \right).
\end{align*}
\]

Considering \( c \to \infty \), the above formula and formula (39) are contradictory. Thus, (40) is valid. The proof is completed. \( \square \)

4. Numerical Results

In this section, numerical results of Algorithm 1 are reported, and the following methods were compared: (i) MTBSBFMS method (\( B_{k+1} \) is updated by (17) with \( y_k \) and \( \delta_k \) given by (18) and (19)). (ii) SBFGS method (\( B_{k+1} \) is updated by (14) with \( y_k \) and \( \delta_k \) given by (11) and (16)).

4.1. General Unconstrained Optimisation Problems

Tested problems: a total of 74 test questions, listed in Table 1 and derived from the studies by Bongartz et al. and More et al. [47, 48].

Parameters: Algorithm 1 runs with \( \delta = 0.2, \delta_1 = 0.15, \sigma = 0.85, \beta_k = 10^{-17}, \) and \( B_0 = I. \)

Dimensionality: the algorithm is tested in the following three dimensions: 300, 900, and 2700.
Figure 1: Performance profiles of these methods (CPU time).

Figure 2: Performance profiles of these methods (NI).

Figure 3: Performance profiles of these methods (NFG).
Himmelblau stop rule [49]: if $|f(x_k)| > e_1$, then set $S_1 = |f(x_k) - f(x_{k+1})|$ or $|f(x_k) - f(x_{k+1})|/|f(x_k)|$. The iterations are stopped if $|g(x)| < \varepsilon$ or $S_1 < e_2$ holds, where $e_1 = e_2 = 10^{-5}$ and $\varepsilon = 10^{-6}$.

Experiment environment: all programs are written in MATLAB R2014a and run on a PC with an Inter(R) Core(TM) i5-4210U CPU at 1.70GHz, 8.00 GB of RAM, and the Windows 10 operating system.

| No. | Dim | NI | NFG | CPU time | NI | NFG | CPU time |
|-----|-----|----|-----|----------|----|-----|----------|
| 1   | 300 | 29 | 63  | 0.3125   | 25 | 57  | 0.3125   |
| 1   | 900 | 24 | 56  | 5.859375 | 23 | 54  | 5.96875  |
| 1   | 2700| 22 | 48  | 86.9375  | 29 | 67  | 135.6875 |
| 2   | 300 | 50 | 114 | 0.59375  | 46 | 102 | 0.5      |
| 2   | 900 | 51 | 114 | 12.890625| 50 | 112 | 13.51625 |
| 2   | 2700| 53 | 120 | 217.90625| 53 | 122 | 254.75   |
| 3   | 300 | 50 | 137 | 0.515625 | 47 | 120 | 0.5      |
| 3   | 900 | 75 | 215 | 17.59375 | 63 | 167 | 16.75    |
| 3   | 2700| 43 | 135 | 184.5625 | 60 | 166 | 280.48375|
| 4   | 300 | 104| 370 | 1.25     | 62 | 176 | 0.75     |
| 4   | 900 | 87 | 258 | 22.453125| 38 | 109 | 9.1875   |
| 4   | 2700| 67 | 179 | 308.3125 | 50 | 149 | 236.98375|
| 5   | 300 | 18 | 48  | 0.21875  | 22 | 58  | 0.296875 |
| 5   | 900 | 18 | 45  | 4.5625   | 20 | 46  | 5.171875 |
| 5   | 2700| 20 | 45  | 90.25    | 23 | 58  | 106.98375|
| 6   | 300 | 68 | 152 | 0.828125 | 68 | 152 | 0.796875 |
| 6   | 900 | 69 | 158 | 18.375   | 69 | 158 | 18.40625 |
| 6   | 2700| 85 | 192 | 405.84375| 85 | 192 | 410.765625|
| 7   | 300 | 76 | 154 | 0.8125   | 76 | 154 | 0.875    |
| 7   | 900 | 133| 268 | 36.6875  | 133| 268 | 36.84375 |
| 7   | 2700| 232| 466 | 1158.734375| 231| 464 | 1151.625 |
| 8   | 300 | 22 | 49  | 0.203125 | 26 | 54  | 0.265625 |
| 8   | 900 | 25 | 55  | 6.515625 | 25 | 55  | 6.421875 |
| 8   | 2700| 25 | 55  | 118.640625| 25 | 55  | 116.859375|
| 9   | 300 | 7  | 16  | 0.0625   | 12 | 26  | 0.109375 |
| 9   | 900 | 7  | 16  | 1.546875 | 12 | 26  | 2.84375  |
| 9   | 2700| 8  | 18  | 31.96875 | 12 | 26  | 49.046875|
| 10  | 300 | 2  | 9   | 0        | 2  | 9   | 0        |
| 10  | 900 | 2  | 9   | 0.0625   | 2  | 9   | 0.0625   |
| 10  | 2700| 2  | 9   | 0.25     | 2  | 9   | 0.25     |
| 11  | 300 | 75 | 194 | 0.921875 | 46 | 94  | 0.59375  |
| 11  | 900 | 95 | 272 | 26.359375| 66 | 134 | 18.484375|
| 11  | 2700| 6  | 20  | 19.875   | 97 | 196 | 472.390625|
| 12  | 300 | 11 | 24  | 0.125    | 11 | 24  | 0.109375 |
| 12  | 900 | 13 | 28  | 3.493125 | 13 | 28  | 3.296875 |
| 12  | 2700| 13 | 28  | 60.34375 | 13 | 28  | 58.40625 |
| 13  | 300 | 11 | 25  | 0.125    | 10 | 23  | 0.125    |
| 13  | 900 | 8  | 23  | 1.921875 | 10 | 26  | 2.46875  |
| 13  | 2700| 19 | 94  | 88.65625 | 19 | 96  | 88.71875 |
| 14  | 300 | 8  | 20  | 0.125    | 8  | 20  | 0.15625  |
| 14  | 900 | 7  | 18  | 1.703125 | 7  | 18  | 1.703125 |
| 14  | 2700| 7  | 18  | 27.6875  | 7  | 18  | 27.421875|
| 15  | 300 | 19 | 43  | 0.359375 | 22 | 49  | 0.40625  |
| 15  | 900 | 25 | 57  | 7.125    | 39 | 82  | 11.078125|
| 15  | 2700| 25 | 55  | 119.40625| 41 | 86  | 198.203125|
| 16  | 300 | 9  | 21  | 0.109375 | 10 | 23  | 0.0625   |
| 16  | 900 | 8  | 18  | 1.921875 | 9  | 21  | 2.15625  |
| 16  | 2700| 11 | 24  | 49.359375| 10 | 22  | 43.046875|
| 17  | 300 | 33 | 73  | 0.421875 | 32 | 75  | 0.390625 |
| 17  | 900 | 23 | 49  | 5.984375 | 23 | 49  | 5.8125   |
| 17  | 2700| 25 | 53  | 115.890625| 25 | 53  | 112.984375|
Table 3: The numerical results for problems 18–34.

| No. | Dim | NI | NFG | CPU time | NI | NFG | CPU time |
|-----|-----|----|-----|----------|----|-----|----------|
| 18  | 300 | 3  | 10  | 0        | 3  | 10  | 0        |
| 18  | 900 | 3  | 10  | 0.53125  | 3  | 10  | 0.53125  |
| 18  | 2700| 3  | 10  | 9.09375  | 3  | 10  | 8.75     |
| 19  | 300 | 3  | 10  | 0        | 3  | 10  | 0.0625   |
| 19  | 900 | 3  | 10  | 0.53125  | 3  | 10  | 0.484375 |
| 19  | 2700| 3  | 10  | 8.578125 | 3  | 10  | 8.421875 |
| 20  | 300 | 33 | 71  | 0.34375  | 34 | 68  | 3.59375  |
| 20  | 900 | 12 | 31  | 2.234375 | 12 | 31  | 2.171875 |
| 20  | 2700| 12 | 35  | 44.546875| 43 | 93  | 191.3125 |
| 21  | 300 | 25 | 56  | 0.28125  | 34 | 74  | 0.390625 |
| 21  | 900 | 25 | 56  | 6.703125 | 35 | 76  | 9.265625 |
| 21  | 2700| 26 | 58  | 124.484375|36 | 78  | 170.34375|
| 22  | 300 | 8  | 30  | 0.109375 | 8  | 31  | 0.125    |
| 22  | 900 | 8  | 31  | 1.90625  | 8  | 31  | 2        |
| 22  | 2700| 8  | 31  | 33.515625| 8  | 31  | 33.59375 |
| 23  | 300 | 34 | 85  | 0.359375 | 52 | 109 | 0.5625   |
| 23  | 900 | 39 | 89  | 10.65625 | 51 | 117 | 13.6875  |
| 23  | 2700| 47 | 111 | 233.65625|47 | 112 | 223.71875|
| 24  | 300 | 33 | 162 | 0.296875 | 29 | 145 | 0.234375 |
| 24  | 900 | 14 | 111 | 1.03125  | 37 | 171 | 6.890625 |
| 24  | 2700| 14 | 111 | 15.34375 | 15 | 114 | 24.234375|
| 25  | 300 | 90 | 262 | 1.0625   | 127| 348 | 1.5      |
| 25  | 900 | 123| 346 | 33.765625|88 | 257 | 23.703125|
| 25  | 2700| 56 | 139 | 270.703125|97 | 284 | 463.609375|
| 26  | 300 | 56 | 134 | 0.640625 | 56 | 134 | 0.609375 |
| 26  | 900 | 65 | 152 | 17.359375|65 | 152 | 17.296875|
| 26  | 2700| 61 | 146 | 292.546875|61 | 146 | 287.25   |
| 27  | 300 | 6  | 16  | 0.0625   | 6  | 16  | 0        |
| 27  | 900 | 11 | 31  | 2.75     | 11 | 31  | 2.65625  |
| 27  | 2700| 16 | 45  | 75.3125  | 17 | 44  | 79.671875|
| 28  | 300 | 28 | 63  | 0.3125   | 27 | 61  | 0.296875 |
| 28  | 900 | 28 | 63  | 7.34375  | 27 | 62  | 6.921875 |
| 28  | 2700| 25 | 61  | 119.5    | 25 | 61  | 117.265625|
| 29  | 300 | 4  | 17  | 0.0625   | 4  | 15  | 0.046875 |
| 29  | 900 | 4  | 17  | 0.6875   | 4  | 17  | 0.828125 |
| 29  | 2700| 4  | 17  | 9.484375 | 4  | 17  | 9.53125  |
| 30  | 300 | 93 | 188 | 1.203125 | 93 | 188 | 1.046875 |
| 30  | 900 | 164| 330 | 45.96875 | 166| 334 | 46.890625|
| 30  | 2700| 295| 592 | 1476.90625|294| 590 | 1474.96875|
| 31  | 300 | 23 | 52  | 0.21875  | 23 | 52  | 0.21875  |
| 31  | 900 | 27 | 62  | 6.828125 | 27 | 62  | 6.84375  |
| 31  | 2700| 28 | 64  | 126.734375|28 | 64  | 126.421875|
| 32  | 300 | 22 | 46  | 0.28125  | 22 | 46  | 0.25     |
| 32  | 900 | 20 | 44  | 5.171875 | 20 | 44  | 5.03125  |
| 32  | 2700| 47 | 98  | 219.515625|47 | 98  | 218.78125|
| 33  | 300 | 5  | 11  | 0.0625   | 5  | 11  | 0.0625   |
| 33  | 900 | 4  | 9   | 0.625    | 4  | 9   | 0.59375  |
| 33  | 2700| 3  | 7   | 7.421875 | 3  | 7   | 7.34375  |
| 34  | 300 | 3  | 7   | 0.0625   | 3  | 7   | 0        |
| 34  | 900 | 3  | 7   | 0.53125  | 3  | 7   | 0.5      |
| 34  | 2700| 4  | 8   | 13.203125| 4  | 8   | 13.234375|

Symbol representation: No.: the test problem number. CPU time: the CPU time in seconds. NI: the number of iterations. NFG: the total number of function and gradient evaluations.

Image description: Figures 1–3 show the profiles for CPU time, NI, and NFG, and Tables 2–6 provide the detail numerical results. From these figures and tables, it is obvious that the MTPSBFGS method possesses...
better numerical performance between these two methods, that is, the proposed modified scaled BFGS method is reasonable and feasible. The specific reasons for good performance are stated as follows. The parameter scaling the first two terms of the standard BFGS update is determined to cluster the eigenvalues of

| No. | Dim | MTPSBFGS-YWL | SBFGS-WWP |
|-----|-----|--------------|-----------|
|     |     | NI | NFG | CPU time | NI | NFG | CPU time |
| 35  | 300 | 4  | 8  | 0.0    | 4  | 8  | 0       |
| 35  | 900 | 7  | 14 | 1.59375 | 7  | 14 | 1.59375 |
| 35  | 2700| 11 | 22 | 46.875  | 11 | 22 | 46.875  |
| 36  | 300 | 24 | 59 | 0.359375 | 26 | 64 | 0.48675  |
| 36  | 900 | 23 | 66 | 6.359375 | 20 | 58 | 5.9375   |
| 36  | 2700| 18 | 49 | 82.359375| 19 | 46 | 85.5625  |
| 37  | 300 | 136| 275| 1.65625 | 136| 275| 1.6875   |
| 37  | 900 | 235| 473| 67.340625| 235| 473| 67.359375|
| 37  | 2700| 441| 886| 2240.015625| 442| 888| 2234.09375|
| 38  | 300 | 12 | 27 | 0.125   | 12 | 27 | 0.0625   |
| 38  | 900 | 12 | 26 | 2.8125  | 12 | 26 | 2.921875 |
| 38  | 2700| 15 | 33 | 64.90625| 15 | 33 | 64.5625  |
| 39  | 300 | 37 | 80 | 0.40625 | 37 | 80 | 0.48675  |
| 39  | 900 | 43 | 89 | 10.890625| 43 | 91 | 11.265625|
| 39  | 2700| 26 | 52 | 116.703125| 26 | 52 | 116.078125|
| 40  | 300 | 532| 1329| 7.40625 | 958| 1925| 12.96875 |
| 40  | 900 | 546| 1364| 150.921875| 1000| 2008| 274.5625 |
| 40  | 2700| 644| 1605| 3124.65625| 1000| 2014| 4810.296875|
| 41  | 300 | 18 | 41 | 0.1875 | 20 | 45 | 0.1875   |
| 41  | 900 | 19 | 43 | 4.828125 | 19 | 43 | 4.734375 |
| 41  | 2700| 19 | 43 | 85.0625 | 19 | 43 | 84.875   |
| 42  | 300 | 19 | 65 | 0.15625 | 16 | 57 | 0.078125 |
| 42  | 900 | 4  | 21 | 0.046875 | 4  | 21 | 0.046875 |
| 42  | 2700| 4  | 21 | 0.578125 | 4  | 21 | 0.515625 |
| 43  | 300 | 22 | 48 | 0.265625 | 25 | 54 | 0.234375 |
| 43  | 900 | 23 | 50 | 5.84375 | 27 | 58 | 6.984375 |
| 43  | 2700| 25 | 54 | 114.328125| 29 | 62 | 133.78125|
| 44  | 300 | 38 | 80 | 0.453125 | 39 | 82 | 0.546875 |
| 44  | 900 | 36 | 76 | 9.703125 | 43 | 90 | 11.5     |
| 44  | 2700| 39 | 82 | 180.453125| 46 | 96 | 212.125  |
| 45  | 300 | 17 | 40 | 0.171875 | 42 | 42 | 0.28125  |
| 45  | 900 | 17 | 40 | 4.265625 | 18 | 42 | 4.65625  |
| 45  | 2700| 18 | 42 | 80.171875| 19 | 44 | 85.09375 |
| 46  | 300 | 104| 255| 1.5625 | 106| 256| 85.9375  |
| 46  | 900 | 141| 354| 41.203125| 87 | 180| 24.796875|
| 46  | 2700| 164| 418| 829.3125 | 116| 238| 586.296875|
| 47  | 300 | 37 | 80 | 0.734375 | 36 | 78 | 0.71875  |
| 47  | 900 | 44 | 95 | 14.640625| 44 | 95 | 14.78125 |
| 47  | 2700| 13 | 31 | 64.21875 | 13 | 31 | 63.9375  |
| 48  | 300 | 26 | 52 | 0.296875 | 26 | 52 | 0.3125   |
| 48  | 900 | 48 | 96 | 12.890625| 48 | 96 | 12.890625|
| 48  | 2700| 22 | 47 | 99.546875| 22 | 47 | 99.6875  |
| 49  | 300 | 76 | 154| 0.953125| 76 | 154| 0.859375 |
| 49  | 900 | 133| 268| 37.203125| 134| 270| 37.59375 |
| 49  | 2700| 232| 466| 1161.109375| 234| 470| 1176.921875|
| 50  | 300 | 76 | 154| 1.234375| 75 | 152| 1.328125 |
| 50  | 900 | 132| 266| 38.859375| 132| 266| 38.53125 |
| 50  | 2700| 231| 464| 1165.1875| 232| 466| 1170     |
| 51  | 300 | 23 | 48 | 0.296875 | 23 | 48 | 0.3125   |
| 51  | 900 | 23 | 48 | 5.890625 | 23 | 48 | 5.859375 |
| 51  | 2700| 23 | 48 | 102.484375| 23 | 48 | 103.453125|
this matrix, and the parameter scaling the third term is determined to reduce its large eigenvalues, thus obtaining a better distribution of them.

4.2. Muskingum Model in Engineering Problems. In this subsection, we present the Muskingum model, and it has the following form:

Table 5: The numerical results for problems 52–68.

| No. | Dim | MTPSBFGS-YWL | SBFGS-WWP |
|-----|-----|--------------|-----------|
|     |     | NI | NFG | CPU time | NI | NFG | CPU time |
| 52  | 300 | 87 | 200 | 0.875 | 87 | 200 | 1.0625 |
| 52  | 900 | 103 | 236 | 27.6875 | 103 | 236 | 27.875 |
| 52  | 2700 | 118 | 270 | 566.65625 | 118 | 270 | 567.82125 |
| 53  | 300 | 938 | 2375 | 12.09375 | 388 | 778 | 4.671875 |
| 53  | 900 | 1000 | 2537 | 282.375 | 1000 | 2002 | 280.203125 |
| 53  | 2700 | 1000 | 2539 | 5006.875 | 1000 | 2002 | 5001.984375 |
| 54  | 300 | 31 | 72 | 0.328125 | 35 | 88 | 0.390625 |
| 54  | 900 | 42 | 107 | 10.8125 | 24 | 65 | 6.09375 |
| 54  | 2700 | 23 | 67 | 102.75 | 27 | 69 | 121.75 |
| 55  | 300 | 9 | 20 | 0.125 | 18 | 42 | 0.265625 |
| 55  | 900 | 10 | 22 | 2.234375 | 20 | 42 | 4.921875 |
| 55  | 2700 | 11 | 24 | 46.3125 | 21 | 44 | 92.390625 |
| 56  | 300 | 1000 | 2607 | 13.03125 | 375 | 750 | 4.5625 |
| 56  | 900 | 1000 | 2540 | 268.3125 | 1000 | 2000 | 277.515625 |
| 56  | 2700 | 1000 | 2539 | 4518.96875 | 1000 | 2000 | 4637.546875 |
| 57  | 300 | 107 | 260 | 1.578125 | 59 | 124 | 0.796875 |
| 57  | 900 | 99 | 236 | 28.140625 | 85 | 176 | 24.15625 |
| 57  | 2700 | 63 | 149 | 313.140625 | 105 | 216 | 522.734375 |
| 58  | 300 | 97 | 233 | 1.3125 | 69 | 144 | 0.890625 |
| 58  | 900 | 120 | 292 | 33.84375 | 99 | 203 | 24.1875 |
| 58  | 2700 | 120 | 292 | 33.84375 | 99 | 204 | 27.82125 |
| 59  | 300 | 24 | 73 | 0.25 | 76 | 169 | 0.890625 |
| 59  | 900 | 27 | 72 | 7.296875 | 71 | 27 | 7.296875 |
| 59  | 2700 | 31 | 78 | 148.625 | 79 | 173 | 389.4375 |
| 60  | 300 | 109 | 260 | 1.484375 | 60 | 126 | 0.796875 |
| 60  | 900 | 132 | 338 | 37.484375 | 87 | 180 | 24.359375 |
| 60  | 2700 | 176 | 434 | 882.109375 | 116 | 238 | 581.935125 |
| 61  | 300 | 104 | 246 | 1.53125 | 59 | 124 | 0.796875 |
| 61  | 900 | 96 | 225 | 26.609375 | 85 | 176 | 24.1875 |
| 61  | 2700 | 66 | 154 | 324.140625 | 105 | 216 | 521.9375 |
| 62  | 300 | 104 | 259 | 1.390625 | 61 | 146 | 0.8125 |
| 62  | 900 | 89 | 232 | 25.265625 | 99 | 220 | 27.625 |
| 62  | 2700 | 126 | 314 | 623.578125 | 105 | 226 | 521.1875 |
| 63  | 300 | 143 | 362 | 2.046875 | 147 | 308 | 2.046875 |
| 63  | 900 | 97 | 259 | 27.59375 | 173 | 365 | 49.28125 |
| 63  | 2700 | 186 | 465 | 933.5 | 199 | 422 | 1005.09375 |
| 64  | 300 | 40 | 88 | 0.53125 | 45 | 98 | 0.578125 |
| 64  | 900 | 28 | 62 | 7.078125 | 32 | 70 | 8.296875 |
| 64  | 2700 | 30 | 66 | 135.65625 | 33 | 72 | 150.015625 |
| 65  | 300 | 22 | 48 | 0.359375 | 22 | 48 | 0.4375 |
| 65  | 900 | 19 | 45 | 5.046875 | 19 | 45 | 5.1875 |
| 65  | 2700 | 18 | 40 | 79.125 | 18 | 40 | 80.765625 |
| 66  | 300 | 611 | 1233 | 11.171875 | 618 | 1238 | 11.6875 |
| 66  | 900 | 1000 | 2003 | 284.8125 | 1000 | 2002 | 286.65625 |
| 66  | 2700 | 1000 | 2003 | 4583.015625 | 1000 | 2002 | 4587.296875 |
| 67  | 300 | 6 | 21 | 0 | 6 | 21 | 0 |
| 67  | 900 | 12 | 33 | 0.25 | 12 | 33 | 0.21875 |
| 67  | 2700 | 10 | 29 | 12.03125 | 13 | 59 | 1.421875 |
| 68  | 300 | 42 | 50 | 0.234375 | 29 | 60 | 0.321875 |
| 68  | 900 | 25 | 52 | 6.453125 | 31 | 64 | 8.375 |
| 68  | 2700 | 27 | 56 | 126.4375 | 33 | 68 | 156.328125 |
Muskingum model [50]:

\[
\min f(x_1, x_2, x_3) = \sum_{i=1}^{n-1} \left( 1 - \frac{\Delta t}{6} \right) x_1 (x_2 I_{i+1} + (1 - x_2) Q_{i+1})^{x_3} - \left( 1 - \frac{\Delta t}{6} \right) x_1 (x_2 I_i + (1 - x_2) Q_i)^{x_3} - \frac{\Delta t}{2} (I_i - Q_i) + \frac{\Delta t}{3} \left( I_{i+1} - Q_{i+1} \right)^2,
\]

\[(46)\]

Table 6: The numerical results for problems 69–74.

| No. | Dim | NI | NFG | CPU time | NI | NFG | CPU time |
|-----|-----|----|-----|----------|----|-----|----------|
| 69  | 300 | 26 | 56  | 0.265625 | 25 | 54  | 0.21875  |
| 69  | 900 | 28 | 60  | 7.546875 | 25 | 54  | 6.734375 |
| 70  | 2700| 29 | 62  | 136.421875 | 25 | 54  | 119.25   |
| 70  | 300 | 31 | 84  | 0.5     | 30 | 90  | 0.4375   |
| 70  | 900 | 40 | 103 | 11      | 32 | 88  | 8.46875  |
| 70  | 2700| 33 | 91  | 146.765625 | 53 | 122 | 245.71875 |
| 71  | 300 | 342| 894 | 4.171875 | 190| 381 | 2.203125 |
| 71  | 900 | 1000| 2572 | 617.34375 | 534| 1069| 145.265625 |
| 71  | 2700| 1000| 2591 | 750.1875 | 1000| 2001| 4617.703125 |
| 72  | 300 | 124 | 313 | 2.53125  | 109 | 284 | 2.265625 |
| 72  | 900 | 283 | 762 | 94.0625  | 311 | 781 | 107.890625 |
| 72  | 2700| 843 | 2128 | 4629.25  | 871 | 2163| 4782.0625 |
| 73  | 300 | 95  | 192 | 1.0625   | 95  | 192 | 1.125    |
| 73  | 900 | 168 | 338 | 46.984375 | 169 | 340 | 46.765625 |
| 73  | 2700| 296 | 594 | 1477.046875 | 292 | 586 | 1462.71875 |
| 74  | 300 | 37  | 76  | 0.375    | 36  | 74  | 0.34375  |
| 74  | 900 | 50  | 102 | 13.40625 | 50  | 102 | 13.484375 |
| 74  | 2700| 81  | 164 | 396.390625 | 81  | 164 | 399.5625 |

Table 7: Results of the three algorithms.

| Algorithm    | $x_1$    | $x_2$    | $x_3$    |
|--------------|----------|----------|----------|
| BFGS [52]    | 10.8156  | 0.9826   | 1.0625   |
| HIWO [50]    | 13.2813  | 0.8001   | 0.9933   |
| MTPSBFGS     | 11.1849  | 1.0000   | 0.9996   |

Figure 4: Performance of Algorithm 1 in 1960.

Figure 5: Performance of Algorithm 1 in 1961.
whose symbolic representation is as follows: $x_1$ is the storage time constant, $x_2$ is the weight coefficient, $x_3$ is an extra parameter, $I_i$ is the observed inflow discharge, $Q_i$ is the observed outflow discharge, $n$ is the total time, and $\Delta t$ is the time step at time $t_i$ ($i = 1, 2, \ldots, n$).

The observed data of the experiment are obtained from the process of flood runoff from Chenggouwan and Linqing of Nanyunhe in the Haihe Basin, Tianjin, China. Select the initial point $x = [0, 1, 1]^T$ and the time step $\Delta t = 12$ (h). The concrete values of $I_i$ and $Q_i$ for the years 1960, 1961, and 1964 are listed in [51]. The test results are presented in Table 7.

Figures 4–6 and Table 7 imply the following three conclusions: (i) based on the Muskingum model, the efficiency of the MTPSBFGS method is wonderful, and numerical performance of these three algorithms is fantastic. (ii) Compared to other similar methods, the final points ($x_1$, $x_2$, and $x_3$) of the MTPSBFGS method are competitive. (iii) Due to the endpoints of these three methods being different, the Muskingum model may have more approximation optimum points.

5. Conclusion

A modified two parameter scaled BFGS method and the Yuan-Wei-Lu line search technology are introduced in this paper. By scaling the first two terms and the third term of the standard BFGS method with different positive parameters, a new two parameter scaled BFGS method is proposed. In this method, the new value of $y_k$ is given to guarantee better properties of the new scaled BFGS method. With Yuan-Wei-Lu line search, the proposed BFGS method is globally convergent. Numerical results indicate that the modified two parameter scaled BFGS method outperforms the standard BFGS method and even the same type of the BFGS method. As for the longer-term work, there are several points to consider: (1) are there some new values of $y_k$, $\delta_k$, and $y_k$ that make the BFGS method based on the update formula (17) perform better? (2) Whether the new scaled method combined with other line search have also great theoretical results. (3) Some new engineering problems based on the BFGS-type method are worth studying.

Data Availability

The data used to support this study are included within this article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 11661009), the High Level Innovation Teams and Excellent Scholars Program in Guangxi Institutions of Higher Education (Grant no. (2019)52), the Guangxi Natural Science Key Fund (Grant no. 2017GXNSFDA198046), and the Guangxi Natural Science Foundation (Grant no. 2020GXNSFAA159069).

References

[1] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, “A stochastic quasi-Newton method for large-scale optimization,” SIAM Journal on Optimization, vol. 26, no. 2, pp. 1008–1031, 2016.
[2] A. S. Lewis and M. L. Overton, “Nonsmooth optimization via quasi-Newton methods,” Mathematical Programming, vol. 141, no. 1-2, pp. 135–163, 2013.
[3] M. S. Salim and A. I. Ahmed, “A family of Quasi-Newton methods for unconstrained optimization problems,” Optimization, vol. 67, no. 10, pp. 1717–1727, 2018.
[4] Z. Wei, G. Li, and L. Qi, “New quasi-Newton methods for unconstrained optimization problems,” Applied Mathematics and Computation, vol. 175, no. 2, pp. 1156–1188, 2006.
[5] Z. Wei, G. Yu, G. Yuan, and Z. Lian, “The superlinear convergence of a modified BFGS-type method for unconstrained optimization,” Computational Optimization and Applications, vol. 29, no. 3, pp. 315–332, 2004.
[6] G. Yuan, Z. Sheng, B. Wang, W. Hu, and C. Li, “The global convergence of a modified BFGS method for nonconvex functions,” Journal of Computational and Applied Mathematics, vol. 327, pp. 274–294, 2018.
[7] W. Zhou and L. Zhang, “Global convergence of the non-monotone MBFGS method for nonconvex unconstrained minimization,” Journal of Computational and Applied Mathematics, vol. 223, no. 1, pp. 40–47, 2009.
[8] W. Zhou and X. Chen, “Global convergence of a new hybrid Gauss-Newton structured BFGS method for nonlinear least squares problems,” SIAM Journal on Optimization, vol. 20, no. 5, pp. 2422–2441, 2010.
[9] D.-H. Li and M. Fukushima, “A modified BFGS method and its global convergence in nonconvex minimization,” Journal of Computational and Applied Mathematics, vol. 129, no. 1-2, pp. 15–35, 2001.
[10] D.-H. Li and M. Fukushima, “On the global convergence of the BFGS method for nonconvex unconstrained optimization problems,” SIAM Journal on Optimization, vol. 11, no. 4, pp. 1054–1064, 2001.
[11] L. Liu, Z. Wei, and X. Wu, “The convergence of a new modified BFGS method without line searches for unconstrained optimization or complexity systems,” Journal of Systems Science and Complexity, vol. 23, no. 4, pp. 861–872, 2010.

[12] Y. Xiao, Z. Wei, and Z. Wang, “A limited memory BFGS-type method for large-scale unconstrained optimization,” Computers & Mathematics with Applications, vol. 56, no. 4, pp. 1001–1009, 2008.

[13] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, “Algorithm 778: l-BFGS-B: fortran subroutines for large-scale bound-constrained optimization,” ACM Transactions on Mathematical Software, vol. 23, no. 4, pp. 550–560, 1997.

[14] W. Zhou, “A modified BFGS type quasi-Newton method with line search for symmetric nonlinear equations problems,” Journal of Computational and Applied Mathematics, vol. 367, Article ID 112454, 2020.

[15] L. Zhang and H. Tang, “A hybrid MBFGS and CBFGS method for nonconvex minimization with a global complexity bound,” Pacific Journal of Optimization, vol. 14, no. 4, pp. 693–702, 2018.

[16] W. Zhou and L. Zhang, “A modified Broyden-like quasi-Newton method for nonlinear equations,” Journal of Computational and Applied Mathematics, vol. 372, Article ID 112744, 2020.

[17] M. J. D. Powell, “Some global convergence properties of a variable metric algorithm for minimization without exact line searches,” SIAM-AMS Proceedings, vol. 9, pp. 53–72, 1976.

[18] R. H. Byrd, J. Nocedal, and Y.-X. Yuan, “Global convergence of a class of quasi-Newton methods on convex problems,” SIAM Journal on Numerical Analysis, vol. 24, no. 5, pp. 1171–1190, 1987.

[19] L. C. W. Dixon, “Variable metric algorithms: necessary and sufficient conditions for identical behavior of nonquadratic functions,” Journal of Optimization Theory and Applications, vol. 10, no. 1, pp. 34–40, 1972.

[20] A. Griewank, “The global convergence of partitioned BFGS problems with convex decompositions and Lipschitzian gradients,” Mathematical Programming, vol. 50, no. 1–3, pp. 141–175, 1991.

[21] M. J. D. Powell, “On the convergence of the variable metric algorithm,” IMA Journal of Applied Mathematics, vol. 7, no. 1, pp. 21–36, 1971.

[22] W. F. Masnarehas, “The BFGS method with exact line searches fails for non-convex objective functions,” Mathematical Programming, vol. 99, no. 1, pp. 49–61, 2004.

[23] Y.-H. Dai, “Convergence properties of the BFGS algorithm,” SIAM Journal on Optimization, vol. 13, no. 3, pp. 693–701, 2006.

[24] G. Yuan and Z. Wei, “Convergence analysis of a modified BFGS method on convex minimizations,” Computational Optimization and Applications, vol. 47, no. 2, pp. 237–255, 2010.

[25] J. Nocedal, “Theory of algorithms for unconstrained optimization,” Acta Numerica, vol. 1, pp. 199–242, 1992.

[26] S. S. Oren and D. G. Luenberger, “Self-scaling variable metric (SSVM) algorithms, part I: criteria and sufficient conditions for scaling a class of algorithms,” Management Science, vol. 20, no. 5, pp. 845–862, 1974.

[27] J. Nocedal and Y.-X. Yuan, “Analysis of self-scaling quasi-Newton method,” Mathematical Programming, vol. 61, no. 1–3, pp. 19–37, 1993.

[28] M. Al-Baali, “Analysis of a family of self-scaling quasi-Newton methods,” Technical report, Department of Mathematics and Computer Science, United Arab Emirates University, Al Ain, UAE, 1993.

[29] Y.-X. Yuan, “A modified BFGS algorithm for unconstrained optimization,” IMA Journal of Numerical Analysis, vol. 11, no. 3, pp. 325–332, 1991.

[30] M. J. D. Powell, “How bad are the BFGS and DFP methods when the objective function is quadratic?” Mathematical Programming, vol. 34, no. 1, pp. 34–47, 1986.

[31] J. Barzilai and J. M. Borwein, “Two-point step size gradient methods,” IMA Journal of Numerical Analysis, vol. 8, no. 1, pp. 141–148, 1988.

[32] W. Y. Cheng and D. H. Li, “Spectral scaling BFGS method,” Journal of Optimization Theory and Applications, vol. 146, no. 2, pp. 305–319, 2010.

[33] N. Andrei, “An adaptive scaled BFGS method for unconstrained optimization,” Numerical Algorithms, vol. 77, no. 2, pp. 413–432, 2017.

[34] N. Andrei, “A double parameter scaled BFGS method for unconstrained optimization,” Journal of Computational and Applied Mathematics, vol. 332, pp. 26–44, 2018.

[35] Y.-H. Dai and C.-X. Kou, “A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search,” SIAM Journal on Optimization, vol. 23, no. 1, pp. 296–320, 2013.

[36] Z. Dai, X. Dong, J. Kang, and L. Hong, “Forecasting stock market returns: new technical indicators and two-step economic constraint method,” The North American Journal of Economics and Finance, vol. 53, Article ID 101216, 2020.

[37] Z. Dai and H. Zhu, “A modified Hestenes-Stiefel-type derivative-free method for large-scale nonlinear monotone equations,” Mathematics, vol. 8, no. 2, p. 168, 2020.

[38] G. Yuan, X. Wang, and Z. Sheng, “Family weak conjugate gradient algorithms and their convergence analysis for nonconvex functions,” Numerical Algorithms, vol. 84, no. 3, pp. 935–956, 2020.

[39] G. Yuan, J. Lu, and Z. Wang, “The PRP conjugate gradient algorithm with a modified WWP line search and its application in the image restoration problems,” Applied Numerical Mathematics, vol. 152, pp. 1–11, 2020.

[40] G. Yuan, T. Li, and W. Hu, “A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems,” Applied Numerical Mathematics, vol. 147, pp. 129–141, 2020.

[41] G. Yuan, Z. Wei, and Y. Yang, “The global convergence of the Polak-Ribiére-Polyak conjugate gradient algorithm under inexact line search for nonconvex functions,” Journal of Computational and Applied Mathematics, vol. 362, pp. 262–275, 2019.

[42] L. Zhang, “A derivative-free conjugate residual method using secant condition for general large-scale nonlinear equations,” Numerical Algorithms, vol. 83, no. 4, pp. 1277–1293, 2020.

[43] W. Zhou, “A short note on the global convergence of the unmodified PRP method,” Optimization Letters, vol. 7, no. 6, pp. 1367–1372, 2013.

[44] W. Sun and Y. Yuan, Optimization Theory and Methods, Springer US, New York, NY, USA, 2006.

[45] R. H. Byrd, D. C. Liu, and J. Nocedal, “On the behavior of broyden’s class of quasi-Newton methods,” SIAM Journal on Optimization, vol. 2, no. 4, pp. 533–557, 1992.

[46] M. J. D. Powell, “Updating conjugate directions by the BFGS formula,” Mathematical Programming, vol. 38, no. 1, pp. 29–46, 1987.
Transactions on Mathematical Software, vol. 21, no. 1, pp. 123–160, 1995.

[48] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, “Testing unconstrained optimization software,” ACM Transactions on Mathematical Software (TOMS), vol. 7, no. 1, pp. 17–41, 1981.

[49] Y. Yuan and W. Sun, Theory and Methods of Optimization, Science Press of China, Beijing, China, 1999.

[50] A. Ouyang, L.-B. Liu, Z. Sheng, and F. Wu, “A class of parameter estimation methods for nonlinear Muskingum model using hybrid invasive weed optimization algorithm,” Mathematical Problems in Engineering, vol. 2015, Article ID 573894, 15 pages, 2015.

[51] A. Ouyang, Z. Tang, K. Li, A. Sallam, and E. Sha, “Estimating parameters of Muskingum model using an adaptive hybrid PSO algorithm,” International Journal of Pattern Recognition and Artificial Intelligence, vol. 28, pp. 1–29, 2014.

[52] Z. W. Geem, “Parameter estimation for the nonlinear Muskingum model using the BFGS technique,” Journal of Irrigation and Drainage Engineering, vol. 132, no. 5, pp. 474–478, 2006.