AIMS Mathematics

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

AIMS Mathematics, 9(9): 25232–25252.
DOI: 10.3934/math.20241231
Received: 15 May 2024
Revised: 22 July 2024
Accepted: 21 August 2024
Published: 29 August 2024

Research article

An efficient augmented memoryless quasi-Newton method for solving large-scale unconstrained optimization problems

Yulin Cheng and Jing Gao*

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

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

Abstract: In this paper, an augmented memoryless BFGS quasi-Newton method was proposed for solving unconstrained optimization problems. Based on a new modified secant equation, an augmented memoryless BFGS update formula and an efficient optimization algorithm were established. To improve the stability of the numerical experiment, we obtained the scaling parameter by minimizing the upper bound of the condition number. The global convergence of the algorithm was proved, and numerical experiments showed that the algorithm was efficient.

Keywords: unconstrained optimization; quasi-Newton method; BFGS update; secant equation; global convergence
Mathematics Subject Classification: 65K05, 90C31, 90C53

1. Introduction

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

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function and its gradient is denoted by $g(x) = \nabla f(x)$. Generally, (1.1) iterates along the following form:

$$x_{k+1} = x_k + s_k, \text{ for all } k \geq 0, \tag{1.2}$$

where $s_k = \alpha_k d_k$, d_k is the search direction, $\alpha_k > 0$ is a step length often obtained by a line search along d_k.

There are many methods to solve unconstrained optimization problem (1.1). Among these methods, The Newton method has a second-order convergence rate when the Hessian matrix $\nabla^2 f(x_k)$ is positive definite, but it cannot ensure that the direction chosen for the objective function at x_k is a descent while solving unconstrained optimization problems. In addition, every iterations of the Newton method requires the second-order gradient of the objective function, namely the Hessian matrix, which is computationally complex for large-scale problems. To address large-scale problems more effectively, the quasi-Newton method was proposed. The quasi-Newton approach updates the search direction using an approximation Hessian matrix instead of the true Hessian matrix used in the Newton method. This approach can reduce the computation of second-order derivatives, lower the computational complexity, and handle non-differentiable functions in some special cases [1].

The quasi-Newton method is recognized as one of the excellent iterative methods for solving large-scale unconstrained optimization problems (for relevant research, see [22–25]). The fundamental concept of the quasi-Newton technique is to substitute an approximate matrix B_k for the Hessian matrix $\nabla^2 f(x_k)$ in the Newton method [1]. In order to satisfy the secant equation, i.e.,

$$B_{k+1} s_k = y_k, \tag{1.3}$$

where $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$, the approximate matrix B_k of the Hessian matrix is constructed using the quasi-Newton update formula. The direction of the quasi-Newton method can be calculated directly by the following formula:

$$d_0 = -g_0, d_k = -H_k g_k, \text{ for all } k \geq 0, \tag{1.4}$$

where H_k is the approximation of $\nabla^2 f(x_k)^{-1}$ and satisfies secant equation, $H_{k+1} y_k = s_k$, for all $k \geq 0$.

The numerical performance of the quasi-Newton method is directly impacted by the updating of H_k. There are many updated formulas of H_k such as the DFP formula, the BFGS formula, and the SR1 formula. These methods, which are composed of different updating formulas, are particularly effective in solving unconstrained optimization problems and they have promising computational performance in practical problems.

The BFGS method, independently proposed by Broyden, Fletcher, Goldfarb, and Shanno [9–13], is one of the most widely used and successful quasi-Newton methods. B_k is always positive definite during the computation; hence, the BFGS method keeps the convergence and stability for optimization problems. Moreover, the BFGS method has emerged as the preferred option for many applications, including neural networks, image processing, and machine learning (see [26–28]). The BFGS formula is as follows:

$$B_{k+1}^{BFGS} = B_k + \frac{y_k y_k^T}{s_k^T y_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}, \tag{1.5}$$

$$H_{k+1}^{BFGS} = H_k - \frac{s_k y_k^T H_k + H_k y_k s_k^T}{s_k^T y_k} + \left(1 + \frac{y_k^T H_k y_k}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k}. \tag{1.6}$$

The BFGS formula satisfies the secant equation (1.3). Many researchers have improved the formula (1.3) to enhance numerical stability and the accuracy of the approximation Hessian matrix. Zhang et al. [2], Zhang and Xu [3] put forward a modified secant condition

$$B_{k+1} s_k = \check{y}_k, \tag{1.7}$$

$$\check{y}_k = y_k + \tau \frac{\check{\eta}_k}{s_k^T w_k} w_k, \tag{1.8}$$

$$\check{\eta}_k = 2(f_k - f_{k+1}) + s_k^T (g_k + g_{k+1}), \tag{1.9}$$

where $\tau > 0$ and w_k is a vector parameter satisfying $s_k^T w_k \neq 0$.

On the other hand, in order to reduce memory storage and improve the computational efficiency of the algorithm, a memoryless quasi-Newton method is proposed to solve large-scale unconstrained optimization problems, where the inner product of multiple vectors is employed to determine the search direction [4]. Memoryless technology has been employed in numerous works; for example, Babaie-Kafaki and Aminifard [5], Aminifard, Babaie-Kafaki and Ghafoori [6], Babaie-Kafaki, Aminifard and Ghafoori [7], Jourak, Nezhadhosein, and Rahpeymaii [8] have applied a memoryless technique to design and develop new quasi-Newton methods for solving large-scale unconstrained optimization problems, and Narushima, Nakayama, Takemura et al. [29] propose a memoryless quasi-Newton method in Riemannian manifolds.

Our goal in this research is to present an effective augmented memoryless BFGS method for solving unconstrained optimization problems. The major contributions of this paper have at least three aspects as follows:

(1) To establish an effective optimization algorithm for large-scale unconstrained optimization problems, a new augmented memoryless BFGS updating formula is provided that is based on a specific modified secant equation.

(2) To enhance the effectiveness of the experiment, the condition number may be minimized in order to obtain the scaling parameters.

(3) We prove the global convergence of the algorithm and numerical experiments that show the efficiency of the algorithm.

The organization of the article is as follows. In Section 2, we present an augmented memoryless BFGS technique along with the algorithm framework. The descent property and the global convergence of the proposed algorithm are demonstrated in Section 3. In Section 4, the numerical experiment demonstrates the effectiveness of our method in solving large-scale unconstrained optimization problems and nonlinear equations. A conclusion to this work is presented in Section 5.

2. An augmented memoryless BFGS method

Inspired by Zhang and Xu [3] and Aminifard, Babaie-Kafaki, and Ghafoori [6], to improve the precision of the solution, we select $w_k = y_k$ in Eqs (1.7)–(1.9), get a modified secant equation,

$$B_{k+1} s_k = \bar{y}_k, \tag{2.1}$$

where $\bar{y}_k = (1 + \tau_k) y_k$, $\tau_k = \tau \frac{\eta_k}{s_k^T y_k}$ and $\eta_k = \max\{0, \check{\eta}_k\}$.

We modify the BFGS iteration formula (1.5) by a rank-1 correction, which we refer to as the augmented BFGS (ABFGS) formula

$$B_{k+1}^{ABFGS} = B_{k+1}^{BFGS} + \tau_k \frac{y_k s_k^T}{s_k^T s_k}. \tag{2.2}$$

$B_{k+1}^{ABFGS} s_k = \bar{y}_k$ because $B_{k+1}^{BFGS} s_k = y_k$. Therefore, (2.2) satisfies the modified secant equation (2.1).

As is known by the quasi-Newton property, when steplength α_k satifies Wolfe line search,

$$f(x_k + \alpha_k d_k) - f(x_k) \le \delta \alpha_k g_k^T d_k, \tag{2.3}$$

$$\nabla f(x_k + \alpha_k d_k)^T d_k \ge \rho g_k^T d_k, \tag{2.4}$$

with $0 < \delta < \rho < 1$, $s_k^T y_k > 0$ is established. By Theorem 5.2.2 of [1], we know (1.5) is positive definite. Therefore, B_k^{ABFGS} is a positive definite matrix since $\tau_k \ge 0$. In other words, the positive definiteness is passed down to the ABFGS update formula. As a result, the search directions of ABFGS are descent.

The scaled memoryless BFGS (SMBFGS) method is considered an effective tool for solving large-scale unconstrained optimization problems. In order to obtain SMBFGS formula, B_k can be replaced by $\frac{1}{\vartheta_k} I$ in (1.5), namely,

$$B_{k+1}^{SMBFGS} = \frac{1}{\vartheta_k} I + \frac{y_k y_k^T}{s_k^T y_k} - \frac{1}{\vartheta_k} \frac{s_k s_k^T}{s_k^T s_k}, \tag{2.5}$$

where $\vartheta_k > 0$ is called the scaling parameter. Similarly, by replacing H_k with $\vartheta_k I$, we can get the SMBFGS iteration formula for the inverse of the Hessian matrix

$$H_{k+1}^{SMBFGS} = \vartheta_k I - \vartheta_k \frac{s_k y_k^T + y_k s_k^T}{s_k^T y_k} + \left(1 + \vartheta_k \frac{y_k^T y_k}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k}. \tag{2.6}$$

In the formula (2.2), we replace B_{k+1}^{BFGS} with B_{k+1}^{SMBFGS} to make a rank-1 correction, and then, using the Sheran-Morrison formula [1], we get the augmented memoryless BFGS formula (AMBFGS), i.e.,

$$B_{k+1}^{AMBFGS} = B_{k+1}^{SMBFGS} + \tau_k \frac{y_k s_k^T}{s_k^T s_k} = \frac{1}{\vartheta_k}\left(I - \frac{s_k s_k^T}{s_k^T s_k}\right) + \frac{y_k y_k^T}{s_k^T y_k} + \tau_k \frac{y_k s_k^T}{s_k^T s_k}, \tag{2.7}$$

$$H_{k+1}^{AMBFGS} = H_{k+1}^{SMBFGS} - \frac{\tau_k (s_k^T y_k s_k s_k^T - \vartheta_k s_k^T y_k s_k y_k^T + \vartheta_k y_k^T y_k s_k s_k^T)}{(1 + \tau_k) s_k^T y_k s_k^T y_k}. \tag{2.8}$$

For the selection of parameter ϑ_k, Babaie-Kafaki [4] came up with well-structured upper bounds for the condition numbers of the scaled memoryless quasi-Newton formulas based on eigenvalue analysis. It was then demonstrated that the scaling parameter suggested by Oren and Spedicato [14] is the only value that can be found as the lowest of the upper bound given for the condition number of the scaled memoryless BFGS update formula. On the other hand, according to Oren and Luenberger [15], the scaling parameter is the distinct lowest value of the provided upper bound on the condition number of the scaled memoryless DFP update algorithm.

According to [5], the choice of parameter ϑ_k is addressed by minimizing the given upper bound for the condition number of the formula (2.8). Since Wolfe line search (2.4) ensures $s_k^T y_k > 0$, we have $s_k \ne 0$ and $y_k \ne 0$. So a set of mutually orthogonal unit vectors ${q_k^{(i)}}_{i=1}^{n-2}$ exists for which $s_k^T q_k^{(i)} = y_k^T q_k^{(i)} = 0$, yielding $B_{k+1}^{AMBFGS} q_k^{(i)}$, for $i = 1, 2, ..., n-2$. Therefore, $(n-2)$ eigenvalues of the matrix $H_{k+1}^{AMBFGS} (B_{k+1}^{AMBFGS})$ are equivalent to $\vartheta_k (\frac{1}{\vartheta_k})$.

Using the relationship between the determinants and traces of the matrices H_{k+1}^{AMBFGS} and B_{k+1}^{AMBFGS}, other eigenvalues of B_{k+1} and H_{k+1} can be obtained. The relationships are as follows:

$$\rho_k^+ + \rho_k^- = \frac{1}{\vartheta_k} + \frac{\|y_k\|^2}{s_k^T y_k} + \frac{\tau_k s_k^T y_k}{\|s_k\|^2}, \tag{2.9}$$

$$\rho_k^+ \rho_k^- = \frac{\vartheta_k \|s_k\|^2}{(1+\tau_k) s_k^T y_k}. \tag{2.10}$$

Formulas of the two other eigenvalues of B_{k+1}^{AMBFGS}, namely, ρ_k^+ and ρ_k^-, can be obtained,

$$\rho_k^{\pm} = \frac{1}{2}\left(\frac{1}{\vartheta_k} + \frac{\|y_k\|^2}{s_k^T y_k} + \frac{\tau_k s_k^T y_k}{\|s_k\|^2}\right) \pm \frac{1}{2}\sqrt{\left(\frac{1}{\vartheta_k} + \frac{\|y_k\|^2}{s_k^T y_k} + \frac{\tau_k s_k^T y_k}{\|s_k\|^2}\right)^2 - 4\frac{\vartheta_k \|s_k\|^2}{(1+\tau_k) s_k^T y_k}} \tag{2.11}$$

for which $0 < \rho_k^- < \rho_k^+$. Additionally, since H_{k+1}^{AMBFGS} is positive definite, the search direction of AMBFGS is descent. Furthermore, we have $0 < \rho_k^- < \frac{1}{\vartheta_k} < \rho_k^+$ after performing a few algebraic operations. Therefore, $\|H_{k+1}^{AMBFGS}\| = \rho_k^+$ and $\|H_{k+1}^{AMBFGS^{-1}}\| = \rho_k^-$. Now, from (2.9) and (2.10) we can write

$$\begin{aligned} \kappa(H_{k+1}^{AMBFGS}) &= \frac{\rho_k^+}{\rho_k^-} = \frac{\rho_k^{+2}}{\rho_k^+ \rho_k^-} \le \frac{(\rho_k^+ + \rho_k^-)^2}{\rho_k^+ \rho_k^-} \\ &\le \frac{(s_k^T y_k \|s_k\|^2 + \vartheta_k \|s_k\|^2 \|y_k\|^2 + \tau_k \vartheta_k (s_k^T y_k)^2)^2}{(1+\tau_k)\vartheta_k (s_k^T y_k)^3 \|s_k\|^2}, \end{aligned} \tag{2.12}$$

where $\kappa(\cdot)$ expresses the spectral condition number.

It is generally acknowledged that decreasing the condition number in matrix-based computing can enhance the stability of numerical calculations [16]. From (2.12), we derive the minimizer of the upper bound of $\kappa(H_{k+1}^{AMBFGS})$ as follows:

$$\vartheta_k = \frac{s_k^T y_k \|s_k\|^2}{\tau_k (s_k^T y_k)^2 + \vartheta_k \|s_k\|^2 \|y_k\|^2}. \tag{2.13}$$

Now, we present a new augmented memoryless BFGS algorithm for solving unconstrained optimization problems.

Algorithm 2.1. The AMBFGS algorithm

Step 1. Choose an initial point x_0, choose parameters $0 < \delta < \rho < 1$, $\epsilon > 0$, $\epsilon_1 > 0$. Set $H_0 = I$, $d_0 = -H_0 g_0$ and $k := 0$.

Step 2. If $\|g_k\| < \epsilon$, stop; otherwise, go to Step 3.

Step 3. Compute the step size α_k, so that it satisfies Wolfe line search (2.3) and (2.4). Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 4. Compute ϑ_k, if $\vartheta_k < \epsilon_1$, let $\vartheta_k = \frac{s_k^T y_k}{\|y_k\|^2}$, otherwise, obtains ϑ_k by (2.13).

Step 5. Update H_{k+1} by (2.8) and compute the search direction d_k using (1.4).

Step 6. Set $k := k + 1$ and go to Step 2.

Remark 2.1. *Step 4 is set this way in order to avoid ϑ_k falling into a dilemma during the calculation of the algorithm.*

3. The descent property and global convergence

In this section, we demonstrate that the direction is sufficiently descent and the algorithm is globally convergent. Our analysis is based on the following assumptions.

Assumption 3.1. *For arbitrary $x_0 \in \mathbb{R}^n$, $\mathcal{L} = \{x \mid f(x) \leq f(x_0)\}$ is a bounded set and in some neighborhood $\mathcal{U}$ of $\mathcal{L}$, $\nabla f(x)$ is Lipschitz continuous, that is, there exists a constant $L > 0$ such that*

$$\|\nabla f(x) - \nabla f(\check{x})\| \leq L\|x - \check{x}\|, \forall x, \check{x} \in \mathcal{U}. \tag{3.1}$$

Based on Assumption 3.1, we know that there is a positive constant Φ exists such that

$$\|\nabla f(x)\| \leq \Phi, \forall x \in \mathcal{L}. \tag{3.2}$$

Since AMBFGS directions are descent, from (2.4), we have $\{x_k\} \subset \mathcal{L}$.

The boundedness of the parameter ϑ_k in (2.13) is important. We will prove $\vartheta_k \in [m, M]$ in Lemma 3.1.

Lemma 3.1. *Considering f is a uniformly convex function on a neighborhood $\mathcal{U}$ of $\mathcal{L}$, the scaling parameter ϑ_k of the AMBFGS algorithm in (2.13) is well defined and bounded.*

Proof. Let f be uniformly convex on $\mathcal{U}$, then, by Theorem 1.3.16 of [1], for any $x, y \in \mathcal{L}$, we have

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2}v\|y - x\|^2, \tag{3.3}$$

and

$$f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{1}{2}v\|x - y\|^2, \tag{3.4}$$

where $v > 0$ is a constant. Let $y = x_{k+1}$, $x = x_k$, adding (3.3) and (3.4), we can obtain

$$\langle \nabla f(x_{k+1}) - \nabla f(x_k), x_{k+1} - x_k \rangle \geq v\|x_{k+1} - x_k\|^2, \forall k \geq 0.$$

In this paper, $s_k = x_{k+1} - x_k, y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$, and then,

$$s_k^T y_k \geq v\|s_k\|^2, \forall k \geq 0. \tag{3.5}$$

Through (3.1) and (3.5), we can get

$$\frac{v}{L^2} \leq \frac{s_k^T y_k}{\|y_k\|^2} \leq \frac{\|s_k\|^2}{s_k^T y_k} \leq \frac{1}{v}. \tag{3.6}$$

By mean value theorem, (1.9) can be rewritten by

$$\begin{aligned} \check{\eta} &= 2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k \\ &= 2\nabla f(\theta_k)^T (x_k - x_{k+1}) + (g_k + g_{k+1})^T s_k \\ &= -2\nabla f(\theta_k)^T s_k + (g_k + g_{k+1})^T s_k \\ &= (g_k - g(\theta_k) + g_{k+1} - g(\theta_k))^T s_k, \end{aligned} \tag{3.7}$$

where $\theta_k = hx_k + (1-h)x_{k+1}, h \in (0,1)$. Hence, by (3.1) we have that

$$\begin{aligned} |\check{\eta}| &\le (\|g_k - g(\theta_k)\| + \|g_{k+1} - g(\theta_k)\|)\|s_k\| \\ &\le (L\|x_k - \theta_k\| + L\|x_{k+1} - \theta_k\|)\|s_k\| \\ &= L\|s_k\|^2. \end{aligned} \tag{3.8}$$

So, we can get $0 < \tau_k \le \frac{\tau L}{v}$. In this case, using (3.6) and the definition of ϑ_k in (2.13), we have

$$\left|\frac{1}{\vartheta_k}\right| = \left|\frac{\tau_k\|s_k^T y_k\|^2 + \vartheta_k\|s_k\|^2\|y_k\|^2}{s_k^T y_k\|s_k\|^2}\right| \le \frac{\tau_k s_k^T y_k}{\|s_k\|^2} + \frac{\|y_k\|^2}{s_k^T y_k} \le \tau L + \frac{L^2}{v} = \frac{1}{m}. \tag{3.9}$$

Moreover,

$$\left|\frac{1}{\vartheta_k}\right| = \left|\frac{\tau_k\|s_k^T y_k\|^2 + \vartheta_k\|s_k\|^2\|y_k\|^2}{s_k^T y_k\|s_k\|^2}\right| \ge \frac{\tau_k s_k^T y_k}{\|s_k\|^2} \ge \tau_k v = \frac{1}{M}. \tag{3.10}$$

From (3.9) and (3.10), we have

$$\vartheta_k \in [m, M], \tag{3.11}$$

which shows the boundedness of ϑ_k.

Remark 3.1. *In Step 4 of the algorithm, when $\vartheta_k < \epsilon_1$, we set $\vartheta_k = \frac{s_k^T y_k}{\|y_k\|^2}$. Then, by applying Eq (3.5), it can be deduced that ϑ_k is bounded.*

The next lemma states an effective property of the direction (1.4).

Lemma 3.2. *Let f be uniformly convex on the neighborhood $\mathcal{U}$ of $\mathcal{L}$, then search direction $\{d_k\}$ produced by Algorithm 2.1 is sufficient descent, that is*

$$d_k^T g_k \le -\zeta\|g_k\|^2, \forall k > 0. \tag{3.12}$$

Proof. By carefully studying the proof of Lemma 3.6 of [17], we can show that $tr(B_{k+1}^{AMBFGS})$ is bounded.

By Lemma 3.1, we get $s_k^T y_k \ge v\|s_k\|^2, \forall k \ge 0$ and $|\check{\eta}| \le L\|s_k\|^2$. So, considering (2.7), (3.1), (3.2), (3.5) and (3.11), we have

$$\begin{aligned} tr(B_{k+1}^{AMBFGS}) &= tr\left(\frac{1}{\vartheta_k}I + \frac{y_k y_k^T}{s_k^T y_k} - \frac{1}{\vartheta_k}\frac{s_k s_k^T}{s_k^T s_k} + \tau_k \frac{y_k s_k^T}{s_k^T s_k}\right) \\ &= \frac{n}{\vartheta_k} + \frac{y_k^T y_k}{s_k^T y_k} - \frac{1}{\vartheta_k}\frac{s_k^T s_k}{s_k^T s_k} + \tau_k \frac{s_k^T y_k}{s_k^T s_k} \\ &\le \frac{n-1}{m} + \frac{L^2}{v} + \tau L \\ &= \frac{(n-1)v + mL^2 + \tau L m v}{mv}. \end{aligned} \tag{3.13}$$

So, from (1.4) and (3.13), we have

$$g_0^T d_0 = -\|g_0\|^2 \tag{3.14}$$

and

$$g_{k+1}^T d_{k+1} = -g_{k+1}^T H_{k+1} g_{k+1} \leq -\frac{1}{tr(B_{k+1}^{AMBFGS})} \|g_{k+1}\|^2 \leq -\frac{mv}{(n-1)v + mL^2 + \tau Lmv} \|g_{k+1}\|^2. \quad (3.15)$$

Finally, according to (3.14) and (3.15), let

$$\zeta = \min\left\{1, \frac{mv}{(n-1)v + mL^2 + \tau Lmv}\right\}, \quad (3.16)$$

then (3.12) is established and the proof is complete.

We next consider the convergence of AMBFGS algorithm. For this purpose, we make the following additional lemma.

Lemma 3.3. *Suppose that Assumption 3.1 holds. Consider iterative form $x_{k+1} = x_k + \alpha_k d_k$, where α_k satisfies the Wolfe conditions (2.3) and (2.4) and d_k satisfies the sufficient descent condition (3.12). If*

$$\sum_{k=0}^{\infty} \frac{1}{\|d_k\|^2} = \infty, \quad (3.17)$$

then,

$$\lim_{k\to\infty} \inf \|g_k\| = 0. \quad (3.18)$$

Proof. Since d_k is sufficiently descent by (3.12) and α_k satisfies the Wolfe conditions (2.3) and (2.4), the Zoutendijk condition [20]

$$\sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty \quad (3.19)$$

holds (see Theorem 3.2 of [18]). To prove this lemma by contradiction, we suppose that there exists a positive constant χ such that

$$\|g_k\| > \chi, \forall k > 0. \quad (3.20)$$

Inequalities (3.12) and (3.20) yield $g_k^T d_k \leq -\zeta\|g_k\|^2 \leq -\zeta\chi^2$, which implies $\frac{\zeta^2\chi^4}{\chi^4} \leq \frac{(g_k^T d_k)^2}{\|d_k\|^2}$. It follows from the above inequality and (3.19) that

$$\sum_{k=0}^{\infty} \frac{\zeta^2\chi^4}{\|d_k\|^2} \leq \sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} = \infty. \quad (3.21)$$

Since this contradicts the Zoutendijk condition (3.19), the proof is complete.

Theorem 3.1. *Suppose f is uniformly convex on the neighborhood $\mathcal{U}$ of $\mathcal{L}$, then the Algorithm 2.1 converges in the sense that (3.18) holds.*

Proof. Lemma 3.2 shows that $d_k \neq 0, \forall k > 0$, therefore, considering Lemma 3.3, it suffices to prove that $\|d_{k+1}\|$ is bounded.

From (2.8), (3.1), (3.2), (3.5)–(3.7) and (3.11), we can get

$$\begin{aligned}
\|H_{k+1}^{AMBFGS}\| &= \left\| H_{k+1}^{SMBFGS} - \frac{\tau_k(s_k^T y_k s_k^T s_k s_k s_k^T - \vartheta_k s_k^T y_k s_k^T s_k s_k y_k^T + \vartheta_k s_k^T s_k y_k^T y_k s_k s_k^T)}{(1+\tau_k) s_k^T y_k s_k^T y_k s_k^T s_k} \right\| \\
&\leq v_k + 2v_k \frac{\|s_k\|\|y_k\|}{s_k^T y_k} + (1 + v_k \frac{\|y_k\|^2}{s_k^T y_k}) \frac{\|s_k\|^2}{s_k^T y_k} + \frac{\tau_k \|s_k\|^2}{(1+\tau_k) s_k^T y_k} + \frac{\vartheta_k \|s_k\|^2 \|y_k\|^2}{(1+\tau_k)\|s_k^T y_k\|^2} + \frac{\vartheta_k}{1+\tau_k} \quad (3.22) \\
&\leq 2M + 2M\frac{L}{v} + \frac{2}{v} + 2M\frac{L^2}{v^2} \\
&= \Lambda.
\end{aligned}$$

Hence, from (1.4) and (3.2), we get

$$\|d_{k+1}\| \leq \|H_{k+1}^{\text{AMBFGS}}\| \|g_{k+1}\| \leq \Lambda\Phi. \tag{3.23}$$

Inequality (3.23) suggests that d_k is bounded. Thus, by Lemma 3.3, we can conclude that the Algorithm 2.1 is convergent.

4. Numerical experiments

In this section, we compare the computational efficiency of SMABFGS (provided by Aminifard et al. [6]), AMBFGS-OS (provided by Algorithm 2.1 and ϑ_k adopts the parameters in [14]) with AMBFGS (provided by Algorithm 2.1). All codes are written in Matlab 2017a and run on a Dell PC with 2.50 GHz CPU processor and 16 GB RAM memory as well as Windows 11 operation system.

We employ the effective Wolfe conditions with parameters $\rho = 0.99$ and $\delta = 10^{-4}$ in the implementations, as detailed in (2.3) and (2.4). When either $k > 10000$ or $\|g_k\| < 10^{-6}$, all algorithms come to an end. The selection of $\tau = 1, \epsilon_1 = 10^{-6}$ is made for the AMBFGS parameters, the selection of $\tau = 1$ and $\vartheta_k = \frac{s_k^T y_k}{\|y_k\|^2}$ is made for the AMBFGS-OS parameters. Additionally, for SMABFGS, we set $p = 1$, $\tau = 1$, and $C = 0.001$, if $\|g_k\| \geq 1$, otherwise, $p = 3$.

4.1. Experiment I: test for unconstrained optimization problems

For experiment I, the 71 unconstrained problems are tested and compared, in which the 1–32 problems are taken from the CUTE library [21], and the others come from the unconstrained problem collections [30, 31]. The number of iterations (Itr), the total number of gradient evaluations (Ng), CPU time (Tcpu), and the gradient value g_k at the end of iteration are also reported in Table 1. The performance of these algorithms is visually described in terms of Tcpu, Itr, and Ng in Figures 1–3, respectively, using the performance profiles suggested by Dolan and Moré [19] (see [19] for further information). In general, the top curve indicates that the applicable approach is the winner for the interpretation of the performance profiles.

Table 1. Numerical results.

		SMABFGS	AMBFGS-OS	AMBFGS
Problems	*n*	$Itr/Ng/Tcpu/\|\|g_k\|\|$	$Itr/Ng/Tcpu/\|\|g_k\|\|$	$Itr/Ng/Tcpu/\|\|g_k\|\|$
cosine	30	121/764/0.028/9.37e-07	1006/3482/0.095/9.79e-07	163/402/0.012/9.32e-07
dixmaana	6000	190/821/63.073/7.68e-07	213/1050/95.180/3.35e-07	262/1506/118.343/9.93e-08
dixmaanb	1500	211/1055/6.160/9.89e-07	127/736/4.796/8.66e-07	202/820/7.594/8.81e-07
dixmaanb	6000	148/592/51.969/2.63e-07	224/987/98.010/4.90e-07	226/1205/99.291/8.95e-07
dixmaanc	2700	192/804/15.977/6.93e-07	148/667/15.514/9.33e-07	131/580/13.602/9.91e-07
dixmaanc	5400	191/751/53.219/2.28e-07	124/537/43.716/8.44e-07	105/312/36.888/2.25e-07
dixmaand	3000	183/767/18.534/7.16e-07	184/570/22.726/9.81e-07	136/237/16.804/4.81e-07
dixmaane	2400	981/1133/61.848/9.53e-07	1050/1219/83.691/6.97e-07	792/966/64.245/8.84e-07
dixmaanf	6000	1385/1652/470.789/6.93e-07	1817/2280/841.260/9.32e-07	1580/1869/712.104/9.99e-07
dixmaang	900	480/608/6.013/8.85e-07	772/1106/11.630/9.59e-07	468/614/7.269/8.28e-07
dixmaanh	1500	912/1084/23.925/9.97e-07	577/704/19.216/8.73e-07	574/679/20.013/9.28e-07
dixmaani	360	4818/5451/7.748/8.01e-07	3702/4439/7.275/9.46e-07	3576/4178/6.944/9.88e-07
dixmaanj	600	3860/4548/25.622/9.64e-07	5469/6408/47.034/9.88e-07	3924/4550/34.264/8.72e-07
dixmaank	300	2704/3225/3.677/9.84e-07	2419/2914/3.924/5.35e-07	2340/2808/3.804/9.18e-07
dixmaanl	300	2787/3166/3.776/9.84e-07	3010/3559/4.868/9.42e-07	2382/2826/3.845/8.92e-07
dixon3dq	100	2240/2517/0.521/8.87e-07	1402/1691/0.360/7.03e-07	1850/2132/0.467/9.71e-07
dqrtic	4000	245/345/39.458/3.01e-08	173/274/35.476/9.00e-07	156/253/31.649/8.29e-07
edensch	60	276/1719/0.151/9.85e-07	316/1913/0.150/4.02e-07	53/117/0.014/5.83e-07
eg2	90	1612/6930/0.572/7.94e-07	4074/33581/2.179/4.20e-07	2164/14247/1.013/6.40e-07
fletchcr	100	1336/11340/0.715/9.99e-07	4005/37731/2.399/9.91e-07	175/354/0.050/9.86e-07
freuroth	4	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN	372/664/0.014/5.55e-07
genrose	10000	345/540/307.401/8.20e-07	256/467/301.828/8.74e-07	302/541/352.489/4.67e-07
himmelbg	7000	10/15/3.993/9.62e-89	10/15/5.002/3.73e-98	10/15/5.049/2.62e-97
liarwhd	30	200/531/0.028/7.86e-07	423/738/0.027/5.67e-07	305/833/0.026/9.99e-07
liarwhd	100	609/947/0.158/8.84e-07	1310/10452/0.734/8.18e-07	1278/7729/0.606/8.00e-07
penalty1	400	5931/62812/52.135/6.91e-07	NaN/NaN/NaN/NaN	3578/35368/47.665/9.25e-07
quartc	4000	245/345/39.672/3.01e-08	173/274/35.207/9.00e-07	156/253/31.673/8.29e-07
tridia	300	1317/1587/1.517/8.34e-07	1430/1748/2.059/8.80e-07	1297/1607/1.801/9.88e-07
woods	1200	586/987/11.051/1.58e-07	553/824/12.976/9.07e-07	456/639/9.842/8.86e-07
VARDIM	160	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN
himmelh	300	NaN/NaN/NaN/NaN	174/519/0.222/6.38e-07	130/371/0.166/1.14e-07
engval1	1000	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN	2536/24677/44.083/9.98e-07
bdexp	5000	27/28/6.250/0.00e+00	27/28/7.874/0.00e+00	27/28/7.818/0.00e+00
exdenschnb	1200	166/655/3.118/7.91e-07	178/811/4.156/9.15e-07	64/205/1.459/7.75e-07
exdenschnb	3000	180/522/17.121/7.91e-07	122/426/14.506/6.98e-08	177/724/21.051/6.74e-07
exdenschnb	6000	119/447/40.595/1.37e-07	173/645/71.994/9.40e-07	118/377/48.837/8.24e-07
exdenschnf	1200	161/628/2.992/7.98e-07	135/413/3.098/1.36e-07	133/512/3.120/8.58e-07
exdenschnf	9000	192/515/139.351/1.37e-07	209/864/225.577/4.80e-07	129/313/143.359/9.11e-07
genquartic	1600	134/467/4.055/7.79e-07	156/413/5.926/3.11e-07	112/385/4.263/5.12e-07
genquartic	9000	224/516/184.527/9.43e-07	160/455/179.843/8.94e-07	167/459/192.917/7.90e-07

Continued on next page

		SMABFGS	AMBFGS-OS	AMBFGS
Problems	n	$Itr/Ng/Tcpu/\|\|g_k\|\|$	$Itr/Ng/Tcpu/\|\|g_k\|\|$	$Itr/Ng/Tcpu/\|\|g_k\|\|$
biggsb1	500	5191/5993/25.611/8.34e-07	4187/4921/26.456/7.89e-07	3962/4625/25.344/8.53e-07
biggsb1	1000	9326/10891/124.762/7.79e-07	7898/9011/133.143/9.26e-07	8819/10180/146.283/7.83e-07
sine	9	99/344/0.011/4.35e-07	NaN/NaN/NaN/NaN	100/364/0.006/7.25e-07
fletcbv3	120	964/1304/0.172/4.69e-07	885/1312/0.186/6.01e-07	563/854/0.127/6.22e-07
nonscomp	500	3022/3601/15.610/6.80e-07	3861/4582/26.791/5.43e-07	2304/2793/15.428/9.01e-07
nonscomp	5000	2102/2671/503.967/9.83e-07	1779/2285/534.087/9.32e-07	1948/2775/583.989/9.88e-07
power1	160	4649/5423/2.199/9.92e-07	5390/6328/3.090/7.32e-07	4178/5012/2.379/9.92e-07
raydan1	600	787/1141/5.836/9.90e-07	1103/2017/10.611/4.99e-07	759/1070/7.263/9.66e-07
raydan2	2000	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN	360/2390/19.695/9.93e-07
diagonal1	100	1553/12157/0.423/9.98e-07	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN
diagonal2	1000	665/846/9.306/9.46e-07	496/630/8.655/3.64e-07	562/677/9.774/9.94e-07
diagonal3	60	989/7265/0.197/9.99e-07	NaN/NaN/NaN/NaN	167/236/0.014/7.36e-07
diagonal8	100	142/649/0.045/6.44e-07	131/817/0.040/8.78e-07	168/1377/0.059/9.95e-07
bv	2000	129/246/8.215/9.78e-07	118/235/8.906/9.89e-07	133/250/9.927/9.98e-07
bv	20000	0/1/1.842/1.25e-08	0/1/0.672/1.25e-08	0/1/0.738/1.25e-08
ie	500	95/301/22.696/6.48e-07	105/518/39.082/7.29e-08	85/305/23.095/1.03e-07
ie	1500	125/473/317.448/5.36e-07	155/674/395.686/9.85e-07	88/293/143.732/7.95e-08
singx	1000	783/1242/14.044/1.92e-07	1367/2357/32.845/9.61e-07	827/1175/19.524/6.34e-07
singx	2000	1277/2319/84.054/8.45e-07	1056/1598/78.703/6.97e-07	939/1388/69.404/6.82e-07
lin	100	218/1368/1.121/8.25e-07	242/1720/1.344/2.49e-07	123/901/0.704/6.16e-07
lin	500	258/1515/8.617/6.69e-07	276/1830/10.716/8.19e-07	264/1609/9.544/7.68e-07
osb2	11	1164/1457/0.119/9.97e-07	1361/1701/0.095/9.49e-07	1210/1493/0.071/8.77e-07
pen1	200	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN	7019/72994/15.319/8.18e-07
pen2	120	1231/4906/1.216/9.31e-07	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN
rosex	300	997/1943/1.191/9.34e-07	1281/2176/1.770/7.14e-07	827/1963/1.245/8.88e-07
rosex	700	803/1530/12.360/4.96e-07	831/1253/13.887/1.00e-06	712/1620/13.958/9.98e-07
trid	900	151/297/2.826/8.64e-07	128/226/2.633/8.40e-07	137/247/2.882/7.66e-07
trid	9000	268/603/273.360/9.25e-07	200/489/255.239/8.67e-08	117/194/135.029/4.13e-07
ExFreudenstein	100	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN
ExBeale	100	436/982/0.101/9.67e-07	231/532/0.058/8.63e-07	398/598/0.077/5.35e-07
hager	150	1225/10722/0.741/9.90e-07	290/2112/0.169/9.98e-07	134/292/0.055/8.64e-07

As can be seen from Table 1, the algorithm presented in the paper is clearly effective for solving most of the tested problems, and it is competitive with the other two algorithms in Itr, Ng, and Tcpu on the tested problems. Figures 1–3 also indicate that the numerical results of the AMBFGS algorithm are better than that of the SMABFGS algorithm and the AMBFGS-OS algorithm. Compared with the SMABFGS algorithm and AMBFGS-OS algorithm, the AMBFGS algorithm is generally in an advantageous position, has better numerical performance, and can solve large-scale unconstrained optimization problems quickly and effectively.

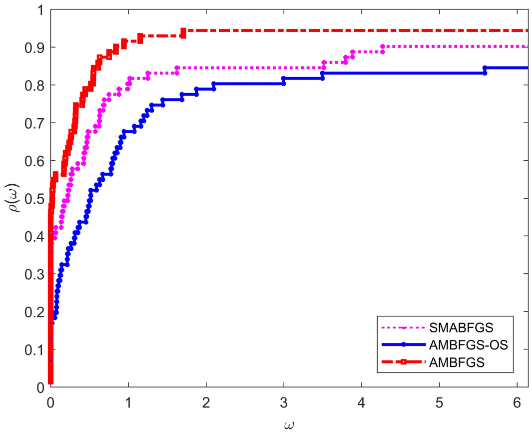

Figure 1. Performance profiles based on CPU time.

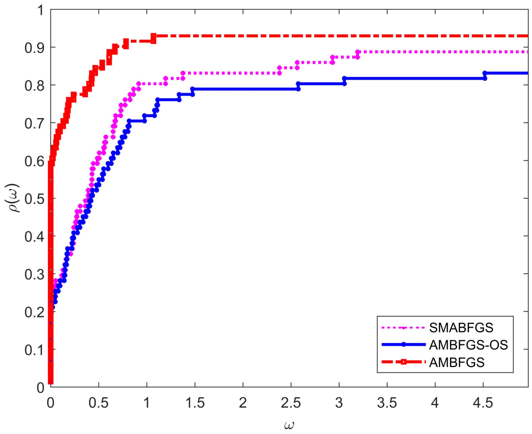

Figure 2. Performance profiles based on number of iterations.

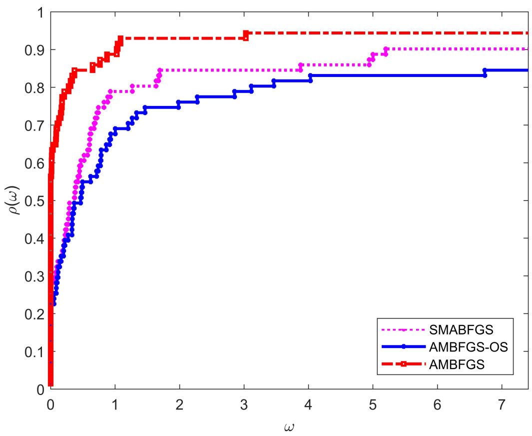

Figure 3. Performance profiles based on number of gradient evaluation.

4.2. Experiment II: test for nonlinear equations

For experiment II, we compare the performance of SMABFGS, AMBFGS-OS with AMBFGS in solving nonlinear equations, and the following mathematical model is considered:

$$\min_{x\in\mathbb{R}^n} f(x) = \frac{1}{2}\|F(x)\|_2^2.$$

Define $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))^T$, $x \in \mathbb{R}^n$ and 7 problems are shown below.
Problem 1. [32] Set $F_i(x) = e^{x_i} - 1$, for $i = 1, 2, \ldots, n$ and $x \in \mathbb{R}^n$.
Problem 2. [32] Set

$$F(x) = \begin{pmatrix} 2.5 & 1 & 0 & ... & 0 \\ 1 & 2.5 & 1 & ... & 0 \\ 0 & 1 & 2.5 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & 2.5 \end{pmatrix} x + (-1, \ldots, -1)^T,$$

and $x \in \mathbb{R}^n$.
Problem 3. [32] Set

$$F(x) = \begin{pmatrix} 2 & -1 & 0 & ... & 0 \\ 0 & 2 & -1 & ... & 0 \\ 0 & 0 & 2 & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix} x + (\sin x_1 - 1, \ldots, \sin x_n - 1)^T,$$

and $x \in \mathbb{R}^n$.
Problem 4. [32] Set $F_i(x) = (e^{x_i})^2 + 3\sin x_i \cos x_i - 1$, for $i = 1, 2, \ldots, n$ and $x \in \mathbb{R}^n$.
Problem 5. [32] Set $F_i(x) = (x_i - 1)^2 - 1.01$, for $i = 1, 2, \ldots, n$ and $x \in \mathbb{R}^n$.
Problem 6. [33] Set

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

and $x \in \mathbb{R}^n$.
Problem 7. [34] Set

$$\begin{aligned} F_1(x) &= \sum_{j=1}^{n} x_j^2, \\ F_i(x) &= -2x_1x_i, \text{for } i = 2, 3, \ldots, n, \end{aligned}$$

and $x \in \mathbb{R}^n$.

The number of iterations (Itr), the total number of gradient evaluations (Ng), CPU time (Tcpu), and the value F_k at the end of iteration are also reported in Tables 2–8. The performance of these algorithms is visually described in terms of Tcpu, Itr, and Ng in Figures 4–6, respectively, using the performance profiles suggested by Dolan and Moré [19]. In general, the top curve indicates that the applicable approach is the winner for the interpretation of the performance profiles. For each problem, we select 4 to 5 initial points from the following 7 points, that is, $x_1 = (1, 1, \dots, 1)^T$, $x_2 = (0.1, 0.1, \dots, 0.1)^T$, $x_3 = (\frac{1}{2}, \frac{1}{2^2}, \dots, \frac{1}{2^n})^T$, $x_4 = (0, \frac{1}{n}, \dots, \frac{n-1}{n})^T$, $x_5 = (1, \frac{1}{2}, \dots, \frac{1}{n})^T$, $x_6 = (\frac{1}{n}, \frac{2}{n}, \dots, 1)^T$, $x_7 = (1 - \frac{1}{n}, 1 - \frac{2}{n}, \dots, 0)^T$.

Table 2. Numerical results (Problem 1).

		SMABFGS	AMBFGS-OS	AMBFGS
x_0	n	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$
x_2	50	164/904/0.030/1.27e-06	131/807/0.021/5.69e-06	153/1002/0.022/1.71e-06
	100	1/2/0.000/1.05e+00	1/2/0.001/1.05e+00	1/2/0.000/1.05e+00
	500	165/933/0.733/1.29e-06	191/1127/1.188/1.22e-06	166/1095/1.135/5.70e-06
x_3	50	63/197/0.015/1.80e-06	68/289/0.011/1.69e-06	41/213/0.006/3.79e-06
	100	63/197/0.015/1.80e-06	68/289/0.020/1.69e-06	41/213/0.010/3.79e-06
	500	63/197/0.275/1.80e-06	68/289/0.504/1.69e-06	41/213/0.353/3.79e-06
x_4	50	86/342/0.017/1.17e-06	72/183/0.007/1.73e-06	74/410/0.012/1.59e-06
	100	93/446/0.024/2.26e-06	77/423/0.022/1.32e-06	43/177/0.011/2.72e-06
	500	132/426/0.824/1.12e-06	75/343/0.650/2.14e-06	133/533/1.130/1.44e-06
x_6	50	136/356/0.030/2.31e-06	117/376/0.030/1.36e-06	178/496/0.041/3.44e-06
	100	96/423/0.041/1.00e-06	114/573/0.053/1.01e-06	61/309/0.027/1.36e-06
	500	111/363/0.695/1.00e-06	127/354/1.084/1.65e-06	91/390/0.788/1.29e-06
x_7	50	81/335/0.021/2.06e-06	72/183/0.016/1.73e-06	74/410/0.024/1.50e-06
	100	89/423/0.036/1.28e-06	77/423/0.035/1.25e-06	43/177/0.018/2.77e-06
	500	147/440/0.938/1.13e-06	74/342/0.634/2.85e-06	134/544/1.142/3.52e-06

Table 3. Numerical results (Problem 2).

		SMABFGS	AMBFGS-OS	AMBFGS
x_0	n	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$
x_2	50	200/319/0.047/2.50e-01	187/279/0.029/2.50e-01	180/274/0.022/2.50e-01
	200	169/286/0.148/2.50e-01	194/356/0.204/2.50e-01	181/350/0.201/2.50e-01
	600	209/342/1.659/2.50e-01	181/327/1.751/2.50e-01	213/373/2.077/2.50e-01
x_5	50	197/290/0.031/2.50e-01	215/322/0.033/2.50e-01	190/301/0.033/2.50e-01
	200	182/293/0.166/2.50e-01	199/310/0.212/2.50e-01	206/359/0.219/2.50e-01
	600	200/403/1.537/2.50e-01	220/368/2.177/2.50e-01	190/330/1.878/2.50e-01
x_6	50	214/447/0.036/2.50e-01	166/359/0.028/2.50e-01	173/341/0.028/2.50e-01
	200	284/601/0.244/2.50e-01	217/514/0.217/2.50e-01	241/524/0.248/2.50e-01
	600	203/375/1.599/2.50e-01	219/462/2.124/2.50e-01	239/480/2.349/2.50e-01
x_7	50	169/265/0.026/2.50e-01	201/322/0.032/2.50e-01	192/318/0.028/2.50e-01
	200	283/624/0.259/2.50e-01	225/540/0.246/2.50e-01	240/579/0.245/2.50e-01
	600	202/440/1.545/2.50e-01	223/442/2.222/2.50e-01	214/427/2.024/2.50e-01

Table 4. Numerical results (Problem 3).

		SMABFGS	AMBFGS-OS	AMBFGS
x_0	n	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$
x_2	60	179/632/0.064/2.69e-07	133/296/0.031/6.83e-07	118/255/0.028/4.13e-07
	100	101/366/0.037/7.11e-07	101/258/0.033/6.26e-07	78/292/0.030/1.22e-06
	500	109/285/0.683/5.31e-07	101/431/0.879/4.88e-07	89/194/0.762/4.85e-07
x_3	60	199/596/0.049/2.77e-07	126/241/0.026/4.99e-07	128/192/0.025/3.69e-07
	100	95/254/0.030/3.18e-07	142/505/0.054/3.51e-07	70/244/0.026/1.19e-06
	500	145/485/0.920/3.90e-07	145/314/1.229/6.28e-07	136/426/1.167/4.31e-07
x_4	60	116/427/0.033/3.10e-07	124/416/0.034/3.34e-07	149/400/0.037/4.58e-07
	100	190/536/0.062/5.61e-07	55/176/0.020/5.43e-07	83/201/0.027/8.31e-07
	500	109/203/0.683/2.50e-06	97/369/0.832/5.04e-07	105/181/0.884/1.16e-06
x_5	60	149/230/0.029/2.74e-07	92/229/0.022/4.93e-07	110/315/0.028/3.83e-07
	100	108/287/0.033/3.72e-07	130/396/0.045/7.37e-07	94/212/0.030/3.90e-07
	500	125/328/0.774/5.23e-07	78/245/0.688/3.87e-06	86/262/0.748/6.28e-07
x_7	60	128/283/0.031/4.34e-07	111/287/0.029/4.78e-07	111/321/0.032/3.34e-07
	100	118/347/0.046/5.85e-07	62/162/0.025/8.92e-07	85/264/0.035/3.85e-07
	500	109/223/0.807/3.90e-07	152/430/1.342/4.11e-07	111/193/0.925/5.41e-07

Table 5. Numerical results (Problem 4).

		SMABFGS	AMBFGS-OS	AMBFGS
x_0	n	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$	$Itr/Ng/Tcpu/\|F_k\|$
x_1	60	212/1199/0.094/2.11e-07	180/1058/0.070/2.77e-07	193/1016/0.070/2.01e-07
	200	219/1117/0.255/2.58e-07	144/1001/0.206/2.38e-07	135/773/0.180/9.55e-07
	500	191/1030/1.304/2.36e-07	208/1125/1.820/2.16e-07	180/1099/1.608/2.62e-07
x_2	60	144/777/0.054/7.60e-06	164/848/0.063/1.66e-06	148/873/0.062/7.31e-07
	200	188/1035/0.223/2.60e-07	145/753/0.191/4.09e-06	196/1038/0.251/2.34e-07
	500	202/1282/1.452/2.01e-07	147/953/1.373/9.18e-07	141/863/1.318/2.10e-07
x_5	60	146/516/0.039/2.99e-07	97/364/0.022/2.39e-07	153/570/0.034/6.92e-07
	200	89/263/0.071/5.99e-07	131/259/0.126/6.19e-07	131/336/0.129/4.29e-07
	500	82/253/0.557/2.89e-07	89/233/0.845/2.55e-07	56/127/0.549/4.42e-07
x_6	60	156/399/0.037/2.16e-07	109/495/0.024/2.90e-07	160/619/0.034/2.54e-07
	200	189/639/0.181/2.10e-07	165/509/0.189/4.42e-07	81/369/0.101/2.36e-07
	500	82/255/0.620/8.53e-07	137/467/1.379/2.24e-07	117/526/1.110/2.14e-07

Table 6. Numerical results (Problem 5).

x_0	n	SMABFGS $Itr/Ng/Tcpu/\|\|F_k\|\|$	AMBFGS-OS $Itr/Ng/Tcpu/\|\|F_k\|\|$	AMBFGS $Itr/Ng/Tcpu/\|\|F_k\|\|$
x_1	50	0/1/0.004/0.00e+00	0/1/0.000/0.00e+00	0/1/0.000/0.00e+00
	200	0/1/0.000/0.00e+00	0/1/0.001/0.00e+00	0/1/0.000/0.00e+00
	600	0/1/0.000/0.00e+00	0/1/0.000/0.00e+00	0/1/0.000/0.00e+00
x_2	50	219/1315/0.047/5.46e-07	113/788/0.022/1.34e-06	165/1170/0.031/5.26e-07
	200	131/972/0.122/6.45e-07	169/889/0.171/3.38e-06	134/838/0.149/3.20e-06
	600	128/869/1.278/5.24e-07	129/839/1.586/6.79e-06	175/1070/1.930/5.12e-07
x_3	50	84/262/0.025/6.97e-06	106/482/0.016/7.14e-07	44/167/0.009/7.72e-06
	200	105/522/0.095/5.16e-07	90/289/0.084/4.12e-06	62/197/0.052/1.26e-06
	600	59/287/0.501/5.99e-07	72/282/0.757/6.40e-07	28/84/0.300/5.70e-06
x_5	50	101/433/0.031/1.01e+00	100/471/0.015/1.01e+00	111/347/0.013/1.01e+00
	200	79/290/0.070/1.01e+00	135/426/0.133/1.01e+00	102/383/0.105/1.01e+00
	600	97/292/0.819/1.01e+00	92/329/0.967/1.01e+00	55/173/0.587/1.01e+00
x_6	50	47/113/0.008/8.20e-07	112/453/0.024/5.22e-08	84/373/0.019/8.33e-07
	200	173/509/0.157/4.90e-07	171/602/0.189/5.13e-07	106/325/0.114/9.78e-07
	600	107/321/0.813/9.48e-07	105/380/1.040/5.60e-07	125/532/1.247/6.48e-07

Table 7. Numerical results (Problem 6).

x_0	n	SMABFGS $Itr/Ng/Tcpu/\|\|F_k\|\|$	AMBFGS-OS $Itr/Ng/Tcpu/\|\|F_k\|\|$	AMBFGS $Itr/Ng/Tcpu/\|\|F_k\|\|$
x_1	50	215/347/0.074/3.84e-07	156/294/0.028/8.83e-07	179/349/0.022/7.99e-07
	200	141/236/0.139/1.31e-06	208/309/0.212/5.07e-07	216/445/0.256/3.49e-07
	500	236/366/2.194/6.31e-07	175/302/2.305/1.45e-06	172/402/2.117/5.08e-07
x_2	50	148/293/0.026/3.86e-07	197/339/0.034/4.32e-07	159/274/0.047/4.71e-07
	200	228/427/0.214/1.15e-06	117/282/0.137/1.44e-06	172/343/0.242/4.65e-07
	500	198/424/1.949/1.20e-06	166/253/2.328/1.16e-06	171/279/2.677/5.59e-07
x_3	50	693/761/0.163/3.10e-07	708/783/0.062/8.10e-07	412/496/0.038/7.75e-07
	200	2595/2733/3.698/6.50e-07	2645/2857/4.039/2.76e-07	1466/1539/2.077/1.89e-06
	500	6512/6606/55.638/3.15e-07	6425/6485/54.137/7.41e-07	3566/3639/29.952/1.89e-06
x_4	50	193/260/0.016/1.62e-06	195/331/0.015/1.30e-06	215/369/0.016/6.72e-07
	200	279/556/0.212/2.24e-07	270/371/0.208/1.31e-06	230/307/0.187/9.36e-07
	500	256/347/1.606/9.32e-07	310/402/2.603/7.21e-07	246/344/2.026/1.45e-06
x_7	50	132/247/0.014/5.59e-07	136/206/0.011/4.30e-07	156/363/0.014/6.96e-07
	200	159/301/0.123/4.65e-07	193/363/0.149/2.78e-07	156/246/0.118/5.53e-07
	500	171/261/1.060/5.29e-07	225/350/1.871/4.36e-07	170/268/1.386/2.69e-07

Table 8. Numerical results (Problem 7).

		SMABFGS	AMBFGS-OS	AMBFGS
x_0	n	$Itr/Ng/Tcpu/\|\|F_k\|\|$	$Itr/Ng/Tcpu/\|\|F_k\|\|$	$Itr/Ng/Tcpu/\|\|F_k\|\|$
x_2	50	81/176/0.029/6.38e-05	113/457/0.029/3.63e-05	103/450/0.029/6.93e-05
	200	31/76/0.031/6.62e-05	34/145/0.041/8.57e-05	20/37/0.021/8.73e-03
	500	65/147/0.392/5.95e-05	119/530/1.090/2.02e-04	20/40/0.165/1.19e-02
x_4	50	69/596/0.029/9.63e-05	74/596/0.028/2.45e-04	74/643/0.031/8.24e-05
	200	91/703/0.107/6.80e-05	171/1075/0.223/6.46e-05	114/796/0.152/8.24e-05
	500	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN	NaN/NaN/NaN/NaN
x_6	50	135/253/0.024/4.99e-05	180/853/0.058/5.59e-05	132/456/0.035/7.84e-03
	200	81/214/0.082/1.26e-04	59/197/0.074/6.21e-05	98/520/0.133/6.58e-05
	500	93/221/0.656/3.83e-03	40/201/0.371/1.09e-04	83/219/0.769/1.29e-02
x_7	50	24/91/0.006/3.18e-04	74/470/0.028/6.85e-05	21/84/0.006/3.37e-04
	200	97/194/0.089/1.59e-04	269/893/0.321/1.38e-04	88/337/0.107/5.85e-05
	500	73/212/0.503/3.32e-05	148/840/1.311/4.55e-05	35/126/0.295/1.98e-04

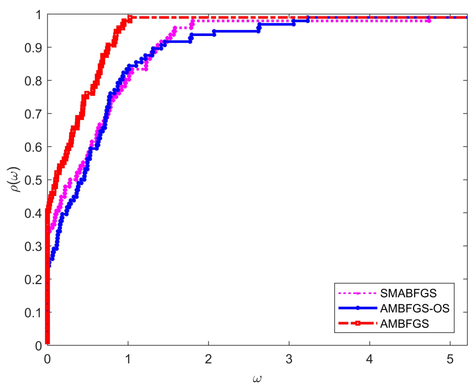

Figure 4. Performance profiles based on CPU time.

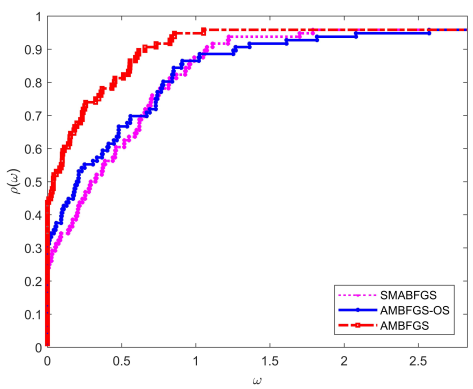

Figure 5. Performance profiles based on number of iterations.

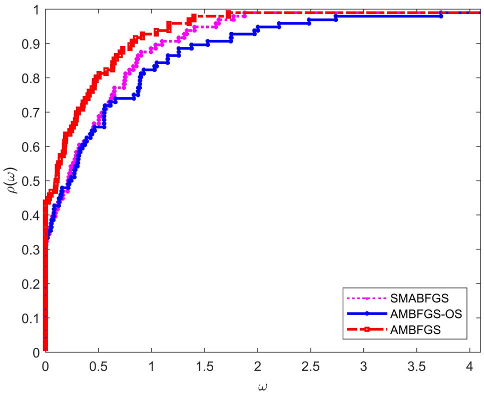

Figure 6. Performance profiles based on number of gradient evaluation.

As can be seen from Tables 2–8, the algorithm presented in the paper is clearly effective for solving most of the tested problems and is competitive with the other two algorithms in Itr, Ng, and Tcpu on the tested problems. Figures 4–6 also indicate that the AMBFGS algorithm, when compared with the SMABFGS and AMBFGS-OS algorithms, generally occupies an advantageous position. It exhibits better numerical performance and can solve nonlinear equations quickly and effectively.

5. Conclusions

In this research, we presented an augmented memoryless BFGS algorithm based on a modified secant condition, which ensures a descent search direction. We determined the scaling parameter by reducing the upper bound of the condition number using an eigenvalue analysis. Global convergence of our approach has been demonstrated under appropriate assumptions. Finally, numerical results obtained by applying the AMBFGS method to solve large-scale unconstrained optimization problems and nonlinear equations demonstrate its encouraging efficiency, even when compared to the SMABFGS method and AMBFGS-OS method.

Author contributions

Yulin Cheng and Jing Gao: Methodology, Software, Visualization, Writing-original draft. All authors of this article have been contributed equally. All authors have read and approved the final version of the manuscript for publication.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by the Scientific and Technological Developing Scheme of Jilin Province (YDZJ202101ZYTS167, YDZJ202201ZYTS303, 20230508184RC); the project of education department of Jilin Province (JJKH20210030KJ, JJKH20230054KJ); the doctoral research project start-up fund of Beihua University; the graduate innovation project of Beihua University (2023037).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. W. Y. Sun, Y. X. Yuan, *Optimization theory and methods: nonlinear programming*, New York: Springer, 2006. https://doi.org/10.1007/b106451
2. J. Z. Zhang, N. Y. Deng, L. H. Chen, New quasi-Newton equation and related methods for unconstrained optimization, *J. Optim. Theory Appl.*, **102** (1999), 147–167. https://doi.org/10.1023/A:1021898630001
3. J. Z. Zhang, C. X. Xu, Properties and numerical performance of quasi-Newton methods with modified quasi-Newton equations, *J. Comput. Appl. Math.*, **137** (2001), 269–278. https://doi.org/10.1016/S0377-0427(00)00713-5
4. S. Babaie-Kafaki, On optimality of the parameters of self-scaling memoryless quasi-Newton updating formulae, *J. Optim. Theory Appl.*, **167** (2015), 91–101. https://doi.org/10.1007/s10957-015-0724-x
5. S. Babaie-Kafaki, Z. Aminifard, Two-parameter scaled memoryless BFGS methods with a nonmonotone choice for the initial step length, *Numer. Algorithms*, **82** (2019), 1345–1357. https://doi.org/10.1007/s11075-019-00658-1
6. Z. Aminifard, S. Babaie-Kafaki, S. Ghafoori, An augmented memoryless BFGS method based on a modified secant equation with application to compressed sensing, *Appl. Numer. Math.*, **167** (2021), 187–201. https://doi.org/10.1016/j.apnum.2021.05.002
7. S. Babaie-Kafaki, Z. Aminifard, S. Ghafoori, A hybrid quasi-Newton method with application in sparse recovery, *Comput. Appl. Math.*, **41** (2022), 249. https://doi.org/10.1007/s40314-022-01962-8
8. M. Jourak, S. Nezhadhosein, F. Rahpeymaii, A new self-scaling memoryless quasi-Newton update for unconstrained optimization, *4OR*, **22** (2024), 235–252. https://doi.org/10.1007/s10288-023-00544-6
9. C. G. Broyden, The convergence of a class of double-rank minimization algorithms 1. General considerations, *IMA J. Appl. Math.*, **6** (1970), 76–90. https://doi.org/10.1093/imamat/6.1.76
10. C. G. Broyden, The convergence of a class of double-rank minimization algorithms 2. The new algorithm, *IMA J. Appl. Math.*, **6** (1970), 222–231. https://doi.org/10.1093/imamat/6.3.222
11. R. Fletcher, A new approach to variable metric algorithms, *Comput. J.*, **13** (1970), 317–322. https://doi.org/10.1093/comjnl/13.3.317
12. D. Goldfarb, A family of variable-metric methods derived by variational means, *Math. Comput.*, **24** (1970), 23–26.

13. D. F. Shanno, Conditioning of quasi-Newton methods for function minimization, *Math. Comput.*, **24** (1970), 647–656.

14. S. S. Oren, E. Spedicato, Optimal conditioning of self-scaling variable metric algorithms, *Math. Program.*, **10** (1976), 70–90. https://doi.org/10.1007/BF01580654

15. S. S. Oren, D. G. Luenberger, Self-scaling variable metric (SSVM) algorithms: Part I: Criteria and sufficient conditions for scaling a class of algorithms, *Manag. Sci.*, **20** (1974), 845–862. https://doi.org/10.1287/mnsc.20.5.845

16. D. S. Watkins, *Fundamentals of matrix computations*, John Wiley & Sons, 2004.

17. S. Babaie-Kafaki, A modified scaled memoryless BFGS preconditioned conjugate gradient method for unconstrained optimization, *4OR*, **11** (2013), 361–374. https://doi.org/10.1007/s10288-013-0233-4

18. J. Nocedal, S. J. Wright, *Numerical optimization*, 2 Eds., New York: Springer, 2006. https://doi.org/10.1007/978-0-387-40065-5

19. E. D. Dolan, J. J. More, Benchmarking optimization software with performance profiles, *Math. Program.*, **91** (2002), 201–213. https://doi.org/10.1007/s101070100263

20. G. Zoutendijk, Nonlinear programming, computational methods, In: *Integer and nonlinear programming*, Amsterdam: North-Holland, 1970, 37–86.

21. N. I. M. Gould, D. Orban, P. L. Toint, CUTEr and SifDec: A constrained and unconstrained testing environment, revisited, *ACM Trans. Math. Software*, **29** (2003), 373–394. https://doi.org/10.1145/962437.962439

22. Y. H. Dai, A perfect example for the BFGS method, *Math. Program.*, **138** (2013), 501–530. https://doi.org/10.1007/s10107-012-0522-2

23. N. Andrei, An adaptive scaled BFGS method for unconstrained optimization, *Numer. Algorithms*, **77** (2018), 413–432. https://doi.org/10.1007/s11075-017-0321-1

24. B. A. Hassan, I. A. R. Moghrabi, A modified secant equation quasi-Newton method for unconstrained optimization, *J. Appl. Math. Comput.*, **69** (2023), 451–464. https://doi.org/10.1007/s12190-022-01750-x

25. G. L. Yuan, X. Zhao, K. J. Liu, X. X. Chen, An adaptive projection BFGS method for nonconvex unconstrained optimization problems, *Numer. Algorithms*, **95** (2024), 1747–1767. https://doi.org/10.1007/s11075-023-01626-6

26. X. M. Lu, C. F. Yang, Q. Wu, J. X. Wang, Y. H. Wei, L. Y. Zhang, et al., Improved reconstruction algorithm of wireless sensor network based on BFGS quasi-Newton method, *Electronics*, **12** (2023), 1–15. https://doi.org/10.3390/electronics12061267

27. V. Krutikov, E. Tovbis, P. Stanimirović, L. Kazakovtsev, D. Karabašević, Machine learning in quasi-Newton methods, *Axioms*, **13** (2024), 1–29. https://doi.org/10.3390/axioms13040240

28. A. B. Abubakar, P. Kumam, H. Mohammad, A. H. Ibrahim, T. Seangwattana, B. A. Hassan, A hybrid BFGS-Like method for monotone operator equations with applications, *J. Comput. Appl. Math.*, **446** (2024), 115857. https://doi.org/10.1016/j.cam.2024.115857

29. Y. Narushima, S. Nakayama, M. Takemura, H. Yabe, Memoryless quasi-Newton methods based on the spectral-scaling Broyden family for Riemannian optimization, *J. Optim. Theory Appl.*, **197** (2023), 639–664. https://doi.org/10.1007/s10957-023-02183-7

30. J. R. Rice, J. J. Moré, B. S. Garbow, K. E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Software*, **7** (1981), 17–41. https://doi.org/10.1145/355934.355936

31. N. Andrei, An unconstrained optimization test functions collection, *Adv. Model. Optim.*, **10** (2008), 147–161.

32. P. J. Liu, X. Y. Wu, H. Shao, Y. Zhang, S. H. Cao, Three adaptive hybrid derivative-free projection methods for constrained monotone nonlinear equations and their applications, *Numer. Linear Algebra Appl.*, **30** (2023), e2471. https://doi.org/10.1002/nla.2471

33. W. J. Zhou, D. M. Shen, Convergence properties of an iterative method for solving symmetric nonlinear equations, *J. Optim. Theory Appl.*, **164** (2015), 277–289. https://doi.org/10.1007/s10957-014-0547-1

34. X. W. Fang, Q. Ni, M. L. Zeng, A modified quasi-Newton method for nonlinear equations, *J. Comput. Appl. Math.*, **328** (2018), 44–58. https://doi.org/10.1016/j.cam.2017.06.024