ANZIAM J. 61(2019), 76–98
doi:10.1017/S1446181118000263

NEW ADAPTIVE BARZILAI–BORWEIN STEP SIZE AND ITS APPLICATION IN SOLVING LARGE-SCALE OPTIMIZATION PROBLEMS

TING LI[1] and ZHONG WAN✉[1]

(Received 8 February, 2018; accepted 30 July, 2018; first published online 3 December 2018)

Abstract

We propose a new adaptive and composite Barzilai–Borwein (BB) step size by integrating the advantages of such existing step sizes. Particularly, the proposed step size is an optimal weighted mean of two classical BB step sizes and the weights are updated at each iteration in accordance with the quality of the classical BB step sizes. Combined with the steepest descent direction, the adaptive and composite BB step size is incorporated into the development of an algorithm such that it is efficient to solve large-scale optimization problems. We prove that the developed algorithm is globally convergent and it R-linearly converges when applied to solve strictly convex quadratic minimization problems. Compared with the state-of-the-art algorithms available in the literature, the proposed step size is more efficient in solving ill-posed or large-scale benchmark test problems.

2010 *Mathematics subject classification*: primary 90C30; secondary 62K05, 68T37.

Keywords and phrases: nonlinear program, step size, algorithm, convergence, large-scale optimization.

1. Introduction

It is well known that large-scale optimization techniques are applied more and more to the fields of industrial engineering and management sciences, such as signal processing [25], machine learning [22] and newsboy problems [5, 20, 28]. On the other hand, it is still a challenge to develop an efficient numerical algorithm for solving these large-scale optimization problems, especially for improvement of computational efficiency which meets practical requirements [26, 29]. In the past two decades, it has attracted much attention from applied mathematicians and experts in engineering, since the existing optimizers cannot be directly applied to solve these problems.

We consider the following large-scale optimization problem:

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

[1]School of Mathematics and Statistics, Central South University, Hunan Changsha, China;
e-mail: math-lit@csu.edu.cn, wanmath@csu.edu.cn.

where $f : R^n \to R$ is continuously differentiable and $n \geq 1000$ in general. To solve problem (1.1), a popular iterative format is constructed as

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

where α_k is a step size and d_k is a search direction. Thus, $\{x_k\}$ is an approximate solution sequence of problem (1.1).

One of the most popular choices for the search directions is generated by the Newton method or the quasi-Newton method (see the articles by Dennis and Moré [12] and Wan et al. [24]). Although these methods have superlinear convergence, they are not suitable for solving large-scale optimization problems, since it is difficult to compute and store the (approximate) Hessian matrices of f at each iteration (see the articles by Deng and Wan [11] and Huang et al. [18]). To overcome the drawbacks in the Newton-type or the quasi-Newton-type methods, numerous spectral gradient methods [6, 17, 19] and conjugate gradient methods [11] have been presented to solve (1.1). Numerical experiments are often employed to show the advantages of their numerical efficiency in solving large-scale optimization problems.

The most direct and the simplest search direction is $d_k = -\nabla f(x_k)$, which is referred to as the steepest descent method. However, owing to its poor convergence rate [17, 18], this method is not often applicable to solve problem (1.1). To exploit its advantages to the full, choice of a suitable step size is often necessary such that its numerical efficiency may beat that of the other methods. Actually, it has been shown that the steepest descent method with the Barzilai–Borwein (BB) step size is R-superlinearly convergent for the two-dimensional quadratic minimization problems [2]. For quadratic minimization problems with any dimension, it has also been proved that the method is still globally convergent [21] and the convergence rate is R-linear [10].

The so-called BB step sizes were first proposed by Barzilai and Borwein [2], where two formats of the step sizes were constructed by approximations to the secant equations. Specifically, the BB step sizes are given by

$$\alpha_k^{\mathrm{BB1}} = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}} \tag{1.2}$$

and

$$\alpha_k^{\mathrm{BB2}} = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}, \tag{1.3}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$. Denote $g_k = \nabla f(x_k)$. Then $y_{k-1} = g_k - g_{k-1}$. The two classical BB step sizes, α_k^{BB1} and α_k^{BB2}, have been successfully applied for solving many problems, such as convex constrained optimization [3, 9], nonnegative matrix factorization [15], compressive sensing [14] and image restoration [4].

Inspired by the BB methods [2], Zhou et al. [30] presented an adaptive Barzilai and Borwein (ABB) step size, given by

$$\alpha_k^{\mathrm{ABB}} = \begin{cases} \alpha_k^{\mathrm{BB2}} & \text{if } \alpha_k^{\mathrm{BB2}}/\alpha_k^{\mathrm{BB1}} < \kappa, \\ \alpha_k^{\mathrm{BB1}} & \text{otherwise}, \end{cases} \tag{1.4}$$

where $\kappa \in (0, 1)$. It is clear that the above ABB method can adaptively choose a small step size or a large step size at each iteration. Although the ABB method in [30] is still linearly convergent in the quadratic case, numerical experiments on some typical test problems indicate that it outperforms the classical BB methods. Huang et al. [16] used the step size (1.4) in nonnegative matrix factorization, combined with a descent direction on the basis of a projected gradient strategy.

Dai et al. [8] conducted an analysis on the geometrical mean of α_k^{BB1} and α_k^{BB2} for the two-dimensional strictly convex quadratic minimization problems, and R-superlinear convergence was established. Very recently, using a new approximation model, Huang and Wan [17] also obtained the BB step size

$$\alpha_k^{\mathrm{NBB}} = \sqrt{\frac{s_{k-1}^T s_{k-1}}{y_{k-1}^T y_{k-1}}} = \sqrt{\alpha_k^{\mathrm{BB1}} \alpha_k^{\mathrm{BB2}}}. \tag{1.5}$$

It turns out to be the geometrical mean of α_k^{BB1} and α_k^{BB2}, which was directly presented by Dai et al. [8]. Apart from their results, the global convergence has been established, as the step size (1.5) is applied to solving the quadratic minimization problems with any size of dimension [17]. Furthermore, based on a new nonmonotone line search, α_k^{NBB} was employed to develop an efficient algorithm for solving nonsmooth and nonlinear systems of equations [17].

Motivated by the superlinear convergence behaviour of the BB methods in the literature, here we intend to construct a new adaptive and composite BB step size by the weighted mean of two classical BB step sizes. To integrate the advantages of the existing BB step sizes, we update the weights at each iteration in accordance with the quality of two types of BB step sizes. Thus, our step size is adaptive and composite, which is different from the other BB step sizes. We are also interested in the convergence properties and numerical performance of the developed algorithm in this paper, which utilizes the simplest gradient method and the proposed step size. Owing to lower cost of computation and storage, the developed algorithm presented in the paper is suitable for solving large-scale optimization problems.

The rest of this paper is organized as follows. In the next section, we present the new adaptive and composite BB step size and develop an algorithm. In Section 3, convergence theory is established as the developed algorithm is applied to solve quadratic minimization problems. Numerical performance of the algorithm is reported in Section 4. Finally, we draw some conclusions in Section 5.

2. Adaptive and composite BB step size and algorithm

In this section, we propose a new adaptive and composite BB step size. Then, combined with the steepest descent direction, a new gradient algorithm has been developed.

Following Barzilai and Borwein [2], we know that α_k^{BB1} and α_k^{BB2} are the optimal solutions of the optimization problems

$$\min_{\alpha} \|\alpha^{-1} s_{k-1} - y_{k-1}\|^2 \quad \text{and} \quad \min_{\alpha} \|\alpha y_{k-1} - s_{k-1}\|^2,$$

respectively.

Clearly, compared with each other, the potential inferiority of α_k^{BB1} and α_k^{BB2} can be observed in

$$\|\alpha_k^{\mathrm{BB1}} y_{k-1} - s_{k-1}\|^2 \geq \|\alpha_k^{\mathrm{BB2}} y_{k-1} - s_{k-1}\|^2$$

and

$$\|(\alpha_k^{\mathrm{BB2}})^{-1} s_{k-1} - y_{k-1}\|^2 \geq \|(\alpha_k^{\mathrm{BB1}})^{-1} s_{k-1} - y_{k-1}\|^2,$$

respectively. Therefore, to integrate the individual advantages of α_k^{BB1} and α_k^{BB2}, we construct a new step size

$$\alpha_k^{\mathrm{CBB}} = \frac{R_2}{R_1 + R_2} \alpha_k^{\mathrm{BB1}} + \frac{R_1}{R_1 + R_2} \alpha_k^{\mathrm{BB2}}, \tag{2.1}$$

where

$$\begin{cases} R_1 = \|\alpha_k^{\mathrm{BB1}} y_{k-1} - s_{k-1}\|^2 - \|\alpha_k^{\mathrm{BB2}} y_{k-1} - s_{k-1}\|^2, \\ R_2 = \|(\alpha_k^{\mathrm{BB2}})^{-1} s_{k-1} - y_{k-1}\|^2 - \|(\alpha_k^{\mathrm{BB1}})^{-1} s_{k-1} - y_{k-1}\|^2. \end{cases} \tag{2.2}$$

Note that

$$\|\alpha_k^{\mathrm{BB2}} y_{k-1} - s_{k-1}\|^2 = 0, \quad \|(\alpha_k^{\mathrm{BB1}})^{-1} s_{k-1} - y_{k-1}\|^2 = 0,$$

$$R_1 = \|\alpha_k^{\mathrm{BB1}} y_{k-1} - s_{k-1}\|^2, \quad R_2 = \|(\alpha_k^{\mathrm{BB2}})^{-1} s_{k-1} - y_{k-1}\|^2.$$

Denote $\mu_k = R_2/(R_1 + R_2)$. Then the equation (2.1) can be written as

$$\alpha_k^{\mathrm{CBB}} = \mu_k \alpha_k^{\mathrm{BB1}} + (1 - \mu_k) \alpha_k^{\mathrm{BB2}}, \tag{2.3}$$

where $0 \leq \mu_k \leq 1$. Observe that α_k^{CBB} is the weighted mean of α_k^{BB1} and α_k^{BB2}. Therefore, α_k^{CBB} is a composite step size. On the other hand, the weight μ_k may be different at each iteration, which is updated by integrating the advantages of α_k^{BB1} and α_k^{BB2}. From this viewpoint, α_k^{CBB} is adaptive.

Remark 2.1. The step size α_k^{ABB} in (1.4) is also called the ABB step size [30], since one can choose a small step size α_k^{BB2} or a large step size α_k^{BB1} at each iteration. However, by definition, the meaning of the word "adaptive" in our paper is different from that of Zhou et al. [30]. In particular, our step size α_k^{CBB} in (2.3) can integrate the advantages of α_k^{BB1} and α_k^{BB2}, since it is the weighted mean of α_k^{BB1} and α_k^{BB2} based on a measure of their qualities by (2.2), rather than an either–or decision as done by Zhou et al. [30].

REMARK 2.2. Additionally, as shown by Barzilai and Borwein [2], both $\alpha_k^{\mathrm{BB1}} I$ and $\alpha_k^{\mathrm{BB2}} I$ are scalar approximations to the inverse of $\nabla^2 f(x_k)$. From (2.3), we know that α_k^{CBB} is also an approximation to the inverse of $\nabla^2 f(x_k)$.

In virtue of an earlier idea [30], we modify α_k^{ABB} in (1.4) as

$$\alpha_k^{\mathrm{CABB}} = \begin{cases} \alpha_k^{\mathrm{BB2}} & \text{if } \alpha_k^{\mathrm{BB2}}/\alpha_k^{\mathrm{BB1}} < \kappa, \\ \alpha_k^{\mathrm{CBB}} & \text{otherwise.} \end{cases} \tag{2.4}$$

Similar to α_k^{ABB}, note that α_k^{CABB} is also adaptive.

With the help of the new BB step size (2.3) or (2.4), we develop Algorithm 1 on the basis of the simplest steepest descent method.

Algorithm 1 New gradient method with adaptive and composite BB step size

```
Step 0 (Initialization). Given initial $x_0 \in R^n$ and $\alpha_0 > 0 \in R$, choose a tolerance $\varepsilon > 0$. Set $k := 0$.
Step 1 (Termination). If $\|\nabla f(x_k)\| \le \varepsilon$, then the algorithm stops. Otherwise, go to Step 2.
Step 2 (Search direction). Compute $d_k := -\nabla f(x_k)$. Go to Step 3.
Step 3 (Update). Set $x_{k+1} := x_k + \alpha_k d_k$.
Step 4 (CBB step size). Compute $s_k$, $y_k$ and $\alpha_{k+1} = \alpha_{k+1}^{\mathrm{CBB}}$ by (2.3). Go to Step 5.
Step 5 (Update). Set $k := k + 1$. Go to Step 1.
```

A special case of problem (1.1) is the following quadratic minimization model:

$$\min f(x) = \tfrac{1}{2} x^T A x - b^T x, \quad x \in R^n, \tag{2.5}$$

where $A \in R^{n\times n}$ is a symmetric and positive-definite (SPD) matrix and $b \in R^n$ is a given vector. In the next section, we will give some convergence analyses about Algorithm 1 for solving model (2.5). We first present a basic property for α_k^{CBB} and α_k^{CABB}.

Let $x_k, x_{k-1} \in R^n$ be two given points. Then, since $g_k = \nabla f(x_k) = Ax_k - b$, we have $y_{k-1} = As_{k-1}$. Now, from (1.2) and (1.3),

$$\alpha_k^{\mathrm{BB1}} = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T A s_{k-1}}, \tag{2.6}$$

$$\alpha_k^{\mathrm{BB2}} = \frac{s_{k-1}^T A s_{k-1}}{s_{k-1}^T A^2 s_{k-1}}. \tag{2.7}$$

PROPOSITION 2.3. *Let α_k be defined by (2.3) or (2.4). Then, for any SPD matrix A, the following inequalities hold:*

$$\begin{cases} 0 < \lambda_n^{-1} \le \alpha_k^{\mathrm{CBB}} \le \lambda_1^{-1} & \text{for all } k, \\ 0 < \lambda_n^{-1} \le \alpha_k^{\mathrm{CABB}} \le \lambda_1^{-1} & \text{for all } k, \end{cases} \tag{2.8}$$

where λ_1 and λ_n are the smallest and the largest eigenvalues of A, respectively.

Proof. From (2.6), (2.7) and by definition [23, Definition 1.2.9], it follows that $(\alpha_k^{\mathrm{BB1}} I)^{-1}$ is the Rayleigh quotient of A at s_{k-1} and $(\alpha_k^{\mathrm{BB2}} I)^{-1}$ is that of A at $\sqrt{A} s_{k-1}$. Since the matrix A is SPD, using the properties of the Rayleigh quotient [23, Theorem 1.2.10],

$$0 < \lambda_1 \le (\alpha_k^{\mathrm{BB1}})^{-1} \le \lambda_n, \quad 0 < \lambda_1 \le (\alpha_k^{\mathrm{BB2}})^{-1} \le \lambda_n \quad \text{for all } k,$$

where λ_1 and λ_n are the smallest and the largest eigenvalues of A, respectively. Consequently,

$$0 < \lambda_n^{-1} \le \alpha_k^{\mathrm{BB1}} \le \lambda_1^{-1}, \quad 0 < \lambda_n^{-1} \le \alpha_k^{\mathrm{BB2}} \le \lambda_1^{-1} \quad \text{for all } k.$$

Thus, for α_k^{CBB} defined by (2.3),

$$0 < \lambda_n^{-1} = \mu_k \lambda_n^{-1} + (1-\mu_k)\lambda_n^{-1} \le \alpha_k^{\mathrm{CBB}} \le \mu_k \lambda_1^{-1} + (1-\mu_k)\lambda_1^{-1} = \lambda_1^{-1}.$$

Since both α_k^{CBB} and α_k^{BB2} satisfy the inequalities in (2.8), it follows from the definition of α_k^{CABB} that

$$0 < \lambda_n^{-1} \le \alpha_k^{\mathrm{CABB}} \le \lambda_1^{-1} \quad \text{for all } k,$$

which completes the proof. □

3. Convergence analysis

In this section, we establish the global convergence of Algorithm 1 as it is applied to solve the quadratic minimization problems. In addition, its R-linear convergence is also established. Before establishing the convergence theory, we first state the following results.

Lemma 3.1. *Let f be defined by (2.5) and let x_* be the unique minimizer of f. Suppose that $\{x_k\}$ is a sequence generated by Algorithm 1. Denote $e_k = x_* - x_k$. Then:*

(1) $Ae_k = d_k$;
(2) $e_{k+1} = \alpha_k((1/\alpha_k)I - A)e_k$.

Lemma 3.2. *Let f be defined by (2.5). Let $\{v_1, v_2, \ldots, v_n\}$ be the orthonormal eigenvectors of A, corresponding to the eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, respectively, where $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$. Suppose that $\{x_k\}$ is a sequence generated by Algorithm 1. Then*

$$e_{k+1} = \sum_{i=1}^{n} c_i^{(k+1)} v_i, \tag{3.1}$$

where

$$c_i^{(k+1)} = \prod_{j=0}^{k} (1 - \alpha_j \lambda_i) c_i^0. \tag{3.2}$$

Proofs of Lemmas 3.1 and 3.2 are similar to those of Huang and Wan [17] and hence omitted here.

REMARK 3.3. By Lemma 3.2, we know that the convergence of the sequence $\{e_k\}$ depends on the behaviour of each of the sequences $\{c_i^{(k)}\}$, $i = 1, 2, \ldots, n$. In Theorem 3.6, we prove that all of these sequences converge to zero, that is, $e_k \to 0$ as $k \to \infty$.

LEMMA 3.4. *Let* $\{c_1^{(k)}\}$ *be a sequence defined by* (3.2). *Then* $c_1^{(k)}$ *converges Q-linearly to zero with convergence factor* $\hat{c} = 1 - (\lambda_1/\lambda_n)$.

Since α_k^{CBB} satisfies (2.8), it follows directly from [21] that Lemma 3.4 is true. Using Lemmas 3.1, 3.2 and 3.4, we now further prove the following result.

LEMMA 3.5. *Let* $\{c_l^{(k)}\}$ *be the sequences defined by* (3.2), $l = 1, \ldots, n$. *If all the sequences* $\{c_1^{(k)}\}, \{c_2^{(k)}\}, \ldots, \{c_l^{(k)}\}$ *converge to zero, then*

$$\liminf_{k\to\infty} |c_{l+1}^{(k)}| = 0.$$

PROOF. By contradiction, suppose that there exists a constant $\epsilon > 0$ such that for all k,

$$(c_{l+1}^{(k)})^2 \lambda_{l+1}^2 > \epsilon. \tag{3.3}$$

From Lemma 3.1,

$$s_k = x_{k+1} - x_k = \alpha_k d_k = \alpha_k A e_k \quad \text{and} \quad y_k = A s_k = \alpha_k A^2 e_k.$$

Using equation (3.1) and the orthonormality of $\{v_1, v_2, \ldots, v_n\}$ yields

$$\alpha_{k+1}^{\mathrm{BB1}} = \frac{s_k^T s_k}{s_k^T y_k} = \frac{(Ae_k)^T (Ae_k)}{(Ae_k)^T (A^2 e_k)} = \frac{\sum_{i=1}^n (c_i^{(k)})^2 \lambda_i^2}{\sum_{i=1}^n (c_i^{(k)})^2 \lambda_i^3} \tag{3.4}$$

and

$$\alpha_{k+1}^{\mathrm{BB2}} = \frac{s_k^T y_k}{y_k^T y_k} = \frac{(Ae_k)^T (A^2 e_k)}{(A^2 e_k)^T (A^2 e_k)} = \frac{\sum_{i=1}^n (c_i^{(k)})^2 \lambda_i^3}{\sum_{i=1}^n (c_i^{(k)})^2 \lambda_i^4}.$$

Since the sequences $\{c_1^{(k)}\}, \{c_2^{(k)}\}, \ldots, \{c_l^{(k)}\}$ are all convergent to zero, there exists $\hat{k}$ sufficiently large such that for any $k \geq \hat{k}$,

$$\sum_{i=1}^{l} (c_i^{(k)})^2 \lambda_i^2 < \frac{\epsilon}{2}. \tag{3.5}$$

From (3.4) and (3.5),

$$\alpha_{k+1}^{\mathrm{BB1}} < \frac{(\epsilon/2) + \sum_{i=l+1}^n (c_i^{(k)})^2 \lambda_i^2}{\sum_{i=l+1}^n (c_i^{(k)})^2 \lambda_i^3}. \tag{3.6}$$

Note that the function $q : R^+ \to R$, given by

$$q(x) = \frac{a + x}{bx}, \quad a, b > 0,$$

is monotonically decreasing in $(0, \infty)$. Combined with (3.6) and, since

$$\sum_{i=l+1}^{n}(c_i^{(k)})^2\lambda_i^3 \geq \Big(\sum_{i=l+1}^{n}(c_i^{(k)})^2\lambda_i^2\Big)\lambda_{l+1} \quad \sum_{i=l+1}^{n}(c_i^{(k)})^2\lambda_i^2 \geq (c_{l+1}^{k})^2\lambda_{l+1}^2 > \epsilon,$$

$$\alpha_{k+1}^{\mathrm{BB1}} < \frac{(\epsilon/2) + \sum_{i=l+1}^{n}(c_i^{(k)})^2\lambda_i^2}{\sum_{i=l+1}^{n}(c_i^{(k)})^2\lambda_i^2\lambda_{l+1}} \leq \frac{(\epsilon/2)+\epsilon}{\epsilon\lambda_{l+1}} = \frac{3}{2}\lambda_{l+1}^{-1}.$$

Consequently, for all $k \geq \hat{k}$,

$$\lambda_n^{-1} \leq \alpha_{k+1}^{\mathrm{BB1}} \leq \tfrac{3}{2}\lambda_{l+1}^{-1}.$$

Similarly, we can prove that for all $k \geq \hat{k}$,

$$\alpha_{k+1}^{\mathrm{BB2}} < \frac{(\epsilon/2)\lambda_{l+1} + \sum_{i=l+1}^{n}(c_i^{(k)})^2\lambda_i^3}{\sum_{i=l+1}^{n}(c_i^{(k)})^2\lambda_i^4} < \frac{(\epsilon/2)\lambda_{l+1} + \epsilon\lambda_{l+1}}{\epsilon\lambda_{l+1}^2} = \frac{3}{2}\lambda_{l+1}^{-1}$$

and, therefore,

$$\lambda_n^{-1} \leq \alpha_{k+1}^{\mathrm{BB2}} \leq \tfrac{3}{2}\lambda_{l+1}^{-1}.$$

By definition of α_k in (2.3), it follows that for all $k \geq \hat{k}+1$,

$$\lambda_n^{-1} \leq \alpha_k \leq \tfrac{3}{2}\lambda_{l+1}^{-1}.$$

Using the fact that $c_{l+1}^{(k+1)} = (1-\alpha_k\lambda_{l+1})c_{l+1}^{(k)}$, we know that for all $k \geq \hat{k}+1$,

$$|c_{l+1}^{(k+1)}| \leq |1-\alpha_k\lambda_{l+1}||c_{l+1}^{(k)}| \leq \hat{c}|c_{l+1}^{(k)}|,$$

where

$$\hat{c} = \max\Big\{\frac{1}{2}, 1-\frac{\lambda_{l+1}}{\lambda_n}\Big\} < 1.$$

Thus, $c_{l+1}^{(k)}$ converges to zero as $k \to \infty$, which contradicts (3.3). Therefore, we have proved that

$$\liminf_{k\to\infty}|c_{l+1}^{(k)}| = 0. \qquad \square$$

Based on Proposition 2.3 and Lemmas 3.1, 3.2 and 3.5, now we establish the global convergence theory of Algorithm 1.

THEOREM 3.6. *Let $\{x_k\}$ be a sequence generated when Algorithm 1 is used to solve model (2.5). Let x_* be the unique minimizer of (2.5). Then either there exists an integer j such that $x_j = x_*$, or the sequence $\{x_k\}$ converges to x_*.*

PROOF. If there exists an integer j such that $x_j = x_*$, then the result holds. Suppose that there is no finite integer j such that $x_j = x_*$. We are going to prove that the sequence $\{e_k\}$ converges to zero.

From (3.1) and the orthonormality of the eigenvectors,

$$\|e_k\|_2^2 = \sum_{i=1}^{n} (c_i^{(k)})^2.$$

Thus, the error sequence $\{e_k\}$ converges to zero if and only if each of the sequences $\{c_i^{(k)}\}$, $i = 1, 2, \dots, n$, converges to zero.

By contradiction, we assume that some of the sequences of $\{c_i^{(k)}\}$ do not converge to zero. Suppose that $\{c_p^{(k)}\}$ is the first sequence which does not converge to zero. By Lemma 3.4, we have $p \geq 2$. Since all the sequences $\{c_1^{(k)}\}, \dots, \{c_{p-1}^{(k)}\}$ converge to zero, for a given $\epsilon > 0$, there exists $\hat{k}$ sufficiently large such that for all $k \geq \hat{k}$, the following inequality holds:

$$\sum_{i=1}^{p-1} (c_i^{(k)})^2 \lambda_i^2 < \frac{\epsilon}{2}. \tag{3.7}$$

By Lemma 3.5, we know that $\liminf_{k\to\infty} |c_p^{(k)}| = 0$. Thus, there exists $k_p \geq \hat{k}$ such that

$$(c_p^{(k_p)})^2 \lambda_p^2 < \epsilon.$$

Let us analyse the properties of $\{c_p^{(k)}\}$ for $k \geq k_p$. Our aim is to prove that there exists a positive constant C such that

$$(c_p^{(k)})^2 \leq C \frac{\epsilon}{\lambda_p^2} \quad \text{for all } k \geq k_p.$$

We conduct the analysis by the following three cases.

Case (a): $k_p \leq k \leq k_0 - 1$.

Let $k_0 (> k_p)$ be the first positive integer satisfying

$$(c_p^{(k_0-1)})^2 \lambda_p^2 < \epsilon, \quad (c_p^{(k_0)})^2 \lambda_p^2 > \epsilon. \tag{3.8}$$

Thus, for all $k_p \leq k \leq k_0 - 1$,

$$\max_{k_p \leq k \leq k_0 - 1} \{(c_p^{(k)})^2\} < \frac{\epsilon}{\lambda_p^2}. \tag{3.9}$$

Case (b): k_0, $k_0 + 1$ and $k_0 + 1 < k \leq j + 1$.

If j is the first positive integer greater than k_0 that satisfies

$$(c_p^{(k)})^2 \lambda_p^2 > \epsilon, \quad (c_p^{(j)})^2 \lambda_p^2 < \epsilon \quad \text{for all } k_0 \leq k \leq j - 1, \tag{3.10}$$

then we can replace $\hat{k}$ in Lemma 3.5 by k_0 and, for $k_0 + 1 \leq k \leq j$, it follows from (3.7) and (3.10) that

$$\lambda_n^{-1} \leq \alpha_k \leq \tfrac{3}{2} \lambda_p^{-1}. \tag{3.11}$$

From Lemma 3.2, we obtain that for all $k' > 0$,

$$\begin{aligned}(c_p^{(k'+1)})^2 &= (1-\alpha_{k'}\lambda_p)^2 (c_p^{(k')})^2 \\ &\leq \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^2 (c_p^{(k')})^2 \\ &\leq \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^4 (c_p^{(k'-1)})^2.\end{aligned} \tag{3.12}$$

As for k_0, from (3.8) and (3.12),

$$\begin{aligned}(c_p^{(k_0)})^2 &\leq \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^2 (c_p^{(k_0-1)})^2 \\ &\leq \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^2 \frac{\epsilon}{\lambda_p^2}.\end{aligned} \tag{3.13}$$

As for k_0+1, from (3.8) and (3.12),

$$(c_p^{(k_0+1)})^2 \leq \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^4 \frac{\epsilon}{\lambda_p^2}. \tag{3.14}$$

Since α_k satisfies (3.11), where $k_0+1 \leq k \leq j$,

$$\begin{aligned}(c_p^{(k+1)})^2 &\leq \max\left\{\frac{1}{2}, 1-\frac{\lambda_p}{\lambda_n}\right\}^2 (c_p^{(k)})^2 \\ &\leq (c_p^{(k_0+1)})^2 \\ &\leq \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^4 \frac{\epsilon}{\lambda_p^2}.\end{aligned} \tag{3.15}$$

From the inequalities (3.9) and (3.13)–(3.15),

$$\max_{k_p \leq k \leq j+1} \{(c_p^{(k)})^2\} \leq \max\left\{1, \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^2, \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^4\right\} \frac{\epsilon}{\lambda_p^2}.$$

Case (c): $k \geq j+2$.

Repeating the processes of Cases (a) and (b), as for $k \geq j+2$, we can also prove that

$$\max_{k \geq j+2} \{(c_p^{(k)})^2\} \leq \max\left\{1, \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^2, \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^4\right\} \frac{\epsilon}{\lambda_p^2}.$$

Combining Cases (a)–(c), we conclude that there exists

$$C = \max\left\{1, \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^2, \left(\frac{\lambda_n-\lambda_1}{\lambda_1}\right)^4\right\}$$

such that

$$(c_p^{(k)})^2 \le C\frac{\epsilon}{\lambda_p^2} \quad \text{for all } k \ge k_p.$$

Define

$$M_\epsilon = \sup_{k\ge k_p}\{(c_p^{(k)})^2\}.$$

Then, as $\epsilon(>0) \to 0$, $M_\epsilon \to 0$, which implies that $\limsup_{k\to\infty}(c_p^{(k)})^2 = 0$. Consequently,

$$\lim_{k\to\infty}|c_p^{(k)}| = 0,$$

which contradicts that $\{c_p^{(k)}\}$ does not converge to zero. Therefore, the sequence $\{e_k\}$ converges to zero. This completes the proof. □

Next, we study local convergence of Algorithm 1 and establish its R-linear convergence result.

THEOREM 3.7. *Let model (2.5) be solved by Algorithm 1 and $\{x_k\}$ be a sequence generated by this algorithm. Then either $g_k = 0$ for some finite k, or the sequence $\{\|g_k\|\}$ R-linearly converges to zero.*

PROOF. Recall that $g_k = \nabla f(x_k)$, where f is the function as defined in Section 1. Since $s_k = x_{k+1} - x_k = -\alpha_k g_k$ and $y_k = As_k$, it directly follows from (2.6) and (2.7) that

$$\alpha_{k+1}^{\mathrm{BB1}} = \frac{g_k^T g_k}{g_k^T A g_k} \quad \text{and} \quad \alpha_{k+1}^{\mathrm{BB2}} = \frac{g_k^T A g_k}{g_k^T A^2 g_k}.$$

By Lemma 3.2,

$$g_k = \sum_{i=1}^n g_i^{(k)} v_i,$$

where $g_i^{(k)} = (1-\alpha_{k-1}\lambda_i)g_i^{(k-1)}$. The orthonormality of $\{v_1, v_2, \dots, v_n\}$ yields

$$g_k^T g_k = \sum_{i=1}^n (g_i^{(k)})^2, \quad Ag_k = \sum_{i=1}^n \lambda_i g_i^{(k)} v_i.$$

Consequently,

$$\alpha_{k+1}^{\mathrm{BB1}} = \frac{\sum_{i=1}^n (g_i^{(k)})^2}{\sum_{i=1}^n (g_i^{(k)})^2\lambda_i} \quad \text{and} \quad \alpha_{k+1}^{\mathrm{BB2}} = \frac{\sum_{i=1}^n (g_i^{(k)})^2\lambda_i}{\sum_{i=1}^n (g_i^{(k)})^2\lambda_i^2}.$$

Since $g_k = Ax_k - b = -Ae_k$, by Lemma 3.2,

$$g_k = \sum_{i=1}^n g_i^{(k)} v_i = -\sum_{i=1}^n c_i^{(k)}\lambda_i v_i.$$

Thus, $g_i^{(k)} = -c_i^{(k)}\lambda_i$.

From the proof of Lemma 3.5, we know that for any integer l, $l = 1, 2, \dots, n-1$, and for a real number $\epsilon > 0$,

$$\lambda_n \geq (\alpha_k^{\mathrm{CBB}})^{-1} \geq \tfrac{2}{3}\lambda_{l+1}$$

in the case that

$$\begin{cases} \displaystyle\sum_{i=1}^{l} (c_i^{(k)})^2 \lambda_i^2 = \sum_{i=1}^{l} (g_i^{(k)})^2 < (1/2)\epsilon < \epsilon, \\ \qquad (c_{l+1}^{(k)})^2 \lambda_{l+1}^2 = (g_{l+1}^{(k)})^2 > \epsilon. \end{cases}$$

Therefore, α_k^{CBB} satisfies Dai's property [7, Property A] and, similar to the proof of a theorem there [7, Theorem 4.1], we get the R-linear convergence of Algorithm 1. □

REMARK 3.8. As in [21, Lemma 2], under the sufficient condition $\lambda_n < 2\lambda_1$ for A, we can also prove that the sequence $\{x_k\}$ Q-linearly converges to x_*. The R-linear or Q-linear convergence is defined by Sun and Yuan [23, Section 1.5.4].

Note that α_k^{BB1}, α_k^{BB2}, α_k^{ABB}, α_k^{CBB} and α_k^{CABB} can all be rewritten in a unified form:

$$\alpha_k = \mu_k \frac{g_{k-1}^T A^{\rho(k)} g_{k-1}}{g_{k-1}^T A^{\rho(k)+1} g_{k-1}} + (1-\mu_k) \frac{g_{k-1}^T A^{\gamma(k)} g_{k-1}}{g_{k-1}^T A^{\gamma(k)+1} g_{k-1}}, \tag{3.16}$$

where $\rho(k), \gamma(k) \in \{0, 1\}$, $\mu_k \in [0, 1]$ and $\rho(k) \neq \gamma(k)$. At the end of this section, we will give the global and local convergence analyses for the unified step size (3.16).

COROLLARY 3.9. *Let $\{x_k\}$ be a sequence generated when Algorithm 1 is used to solve model (2.5), where the step size α_k is computed by (3.16). Let x_* be the unique minimizer of (2.5). Then either there exists an integer j such that $x_j = x_*$, or the sequence $\{x_k\}$ converges to x_*.*

PROOF. Since α_k^{BB1}, α_k^{BB2}, α_k^{ABB}, α_k^{CBB} and α_k^{CABB} all have the property of Proposition 2.3, if α_k is defined by (3.16), then

$$0 < \lambda_n^{-1} \leq \alpha_k \leq \lambda_1^{-1} \quad \text{for all } k.$$

For any integer l, $l = 1, 2, \dots, n-1$, and any real number $\epsilon > 0$, if

$$\begin{cases} \displaystyle\sum_{i=1}^{l} (g_i^{(k)})^2 < 1/2\epsilon, \\ \qquad (g_{l+1}^{(k)})^2 > \epsilon \end{cases}$$

hold, then it follows from (3.16) that

$$\begin{aligned}
\alpha_k &= \mu_k \frac{\sum_{i=1}^{n}(g_i^{(k)})^2\lambda_i^{\rho(k)}}{\sum_{i=1}^{n}(g_i^{(k)})^2\lambda_i^{\rho(k)+1}} + (1-\mu_k)\frac{\sum_{i=1}^{n}(g_i^{(k)})^2\lambda_i^{\gamma(k)}}{\sum_{i=1}^{n}(g_i^{(k)})^2\lambda_i^{\gamma(k)+1}} \\
&\le \mu_k \frac{\sum_{i=1}^{l}(g_i^{(k)})^2\lambda_i^{\rho(k)} + \sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\rho(k)}}{\sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\rho(k)+1}} \\
&\quad + (1-\mu_k)\frac{\sum_{i=1}^{l}(g_i^{(k)})^2\lambda_i^{\gamma(k)} + \sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\gamma(k)}}{\sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\gamma(k)+1}} \\
&\le \mu_k \frac{\lambda_{l+1}^{\rho(k)}\sum_{i=1}^{l}(g_i^{(k)})^2 + \sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\rho(k)}}{\lambda_{l+1}\sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\rho(k)}} \\
&\quad + (1-\mu_k)\frac{\lambda_{l+1}^{\gamma(k)}\sum_{i=1}^{l}(g_i^{(k)})^2 + \sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\gamma(k)}}{\lambda_{l+1}\sum_{i=l+1}^{n}(g_i^{(k)})^2\lambda_i^{\gamma(k)}} \\
&\le \mu_k \frac{(1/2)\epsilon\lambda_{l+1}^{\rho(k)} + \epsilon\lambda_{l+1}^{\rho(k)}}{\epsilon\lambda_{l+1}^{\rho(k)+1}} + (1-\mu_k)\frac{(1/2)\epsilon\lambda_{l+1}^{\gamma(k)} + \epsilon\lambda_{l+1}^{\gamma(k)}}{\epsilon\lambda_{l+1}^{\gamma(k)+1}} \\
&= \mu_k\frac{3/2}{\lambda_{l+1}} + (1-\mu_k)\frac{3/2}{\lambda_{l+1}} = \frac{3}{2}(\lambda_{l+1})^{-1}.
\end{aligned}$$

Thus, by recalling the proofs of Lemma 3.5 and Theorem 3.6, we complete the proof of Corollary 3.9. □

COROLLARY 3.10. *Let model* (2.5) *be solved by Algorithm* 1*, where the step size* α_k *is computed by* (3.16)*. Then either* $g_k = 0$ *for some finite* k*, or the sequence* $\{\|g_k\|\}$ *R-linearly converges to zero.*

PROOF. From the proof of Corollary 3.9, we have that [7, Property (A)] holds for the step size defined by (3.16). By [7, Theorem 4.1], the result in this corollary follows. □

4. Numerical experiments

In this section, we will test the numerical efficiency of Algorithm 1 in virtue of the adaptive and composite BB step size. Especially, we will compare our methods with the other ones available in the literature.

Specifically, we call Algorithm 1 the CBB method. If the step size in Algorithm 1 is computed by (1.2), (1.3), (1.4), (1.5) or (2.4), we call the corresponding algorithms the BB1, BB2, ABB, NBB and CABB methods, respectively. Our first aim is to investigate whether or not the new step size α_k^{CBB} is helpful to improve the numerical efficiency of the state-of-the-art algorithms available in the literature.

For further investigation on the effect of α_k^{CBB}, we are also interested in whether or not the CABB method outperforms the ABB method proposed in [30]. In order to reveal what are the advantages of the adaptive weight μ_k, we compare our method with a modified version of Algorithm 1, where μ_k is a fixed constant μ.

All of the computer codes are written in MATLAB R2014a and numerical experiments are conducted on a PC with a 2.60 GHz CPU and a 4.00 GB memory-based operation system of Windows 7.

For simplicity of the statement, we introduce the following notation:

CBB(μ):	the gradient method with $\alpha_k = \alpha_k^{\mathrm{CBB}}$ and $\mu_k = \mu$;
ABB:	the gradient method with $\alpha_k = \alpha_k^{\mathrm{ABB}}$ and $\kappa = 0.5$;
C(0.8):	the gradient method with $\alpha_k = \alpha_k^{\mathrm{CABB}}$, $\mu_k = 0.8$ and $\kappa = 0.5$;
CABB:	the gradient method with $\alpha_k = \alpha_k^{\mathrm{CABB}}$ and $\kappa = 0.5$;
P:	the serial number of the test problems of Andrei [1];
n:	the size of the problem;
Cd:	the condition number of the Hessian matrix of f;
NI:	the number of iterations;
$f_{\min}$:	the minimal value of the objective function;
T:	the CPU time (s) being accurate to the third decimal place;
F:	the number of iterations exceeds 10 000 or $f_{\min}$ is not the minimal value.

4.1. Test by ill-posed quadratic problems We first investigate the performance of all the algorithms as they are applied to solve the ill-posed quadratic minimization problems.

Consider the following test problem, suggested by Yuan [27]:

$$f(x) = (x - x_*)^T \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)(x - x_*), \quad x \in R^n, \tag{4.1}$$

where $\mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ creates a square diagonal matrix with the elements of the vector $(\lambda_1, \lambda_2, \ldots, \lambda_n)^T$ on the main diagonal.

We solve problem (4.1) for the cases $n = 10, 10^2, 10^3, 10^4$. We randomly choose x_*^i in the interval $(-5, 5)$, $i = 1, 2, \ldots, n$, to randomly generate different test problems. Let $\lambda_1 = 1$. Denote by

$$\lambda_n = Cd (= 10, 10^2, 10^3, 10^4, 10^5)$$

the so-called condition number of the Hessian matrix of the objective function f. Let λ_i be randomly generated in the interval $(1, \lambda_n)$ in order to generate different test problems, $i = 2, 3, \ldots, n-1$. For all the randomly generated test problems, we start with

$$x_0 = \mathrm{zeros}(n, 1), \quad \alpha_0 = \alpha_0^{SD} = \frac{g_0^T g_0}{g_0^T \nabla^2 f(x_0) g_0}.$$

The termination condition of all the algorithms is

$$\|g_k\|_2 \le 10^{-5} \|g_0\|_2.$$

First, for each case with different values of n and λ_n, 10 test problems are generated. Then we evaluate the numerical performance of each algorithm by the average number of iterations required by the algorithms. In Table 1, we report the average number of iterations (NI) required by the different algorithms. The underlined result, the smaller one in each row, indicates that the corresponding algorithm outperforms the others.

From the numerical results in Table 1, we observe the following results.

TABLE 1. Average NI of different algorithms.

n	Cd	BB1	BB2	NBB	CBB	ABB	CABB	C(0.8)
10	10	18.4	18.8	19	19	18.3	19.2	19.4
10	10^2	50.4	55.2	65	$\underline{48.1}$	40.6	$\underline{40.1}$	$\underline{39.6}$
10	10^3	87	96.8	142.2	$\underline{82.3}$	77.5	79.3	92.1
10	10^4	141.9	180.7	380.5	$\underline{139.9}$	151.4	$\underline{134.9}$	155.6
10	10^5	26.8	27	46.1	$\underline{26.8}$	24.7	$\underline{24.7}$	26.3
10^2	10	19.6	19.9	19.3	$\underline{19.5}$	19.6	$\underline{19.1}$	$\underline{19.4}$
10^2	10^2	53.9	54.8	63.2	$\underline{53.9}$	46.9	$\underline{46.7}$	48.1
10^2	10^3	103	114.1	158.9	103.3	117	$\underline{106.5}$	$\underline{100.7}$
10^2	10^4	159.7	180.5	262.5	$\underline{130.5}$	147.1	$\underline{130.8}$	$\underline{137.5}$
10^2	10^5	59.9	70.7	89.6	$\underline{59.9}$	59.3	$\underline{59.1}$	$\underline{53.9}$
10^3	10	19	19.9	18.9	$\underline{18.9}$	19	$\underline{18.9}$	$\underline{19}$
10^3	10^2	50.7	56.2	57.1	51.5	49	$\underline{48.4}$	54
10^3	10^3	114.3	114.6	150.3	115.7	115	$\underline{104.7}$	$\underline{108}$
10^3	10^4	112.9	115.2	158.8	115.4	100.2	102.4	104.5
10^3	10^5	119.4	122.7	163.3	$\underline{100.4}$	118.1	$\underline{116.5}$	$\underline{117.2}$
10^4	10	19	20	19.1	$\underline{19}$	19	19.1	$\underline{19}$
10^4	10^2	50.7	59	62.3	52.8	53.4	$\underline{52.8}$	$\underline{49.2}$
10^4	10^3	109.4	119.9	131.1	113.5	98.8	106.1	104.5
10^4	10^4	114.8	123.8	153.7	$\underline{114.8}$	113.6	$\underline{112.8}$	117.9
10^4	10^5	117.1	129.6	138.5	118.8	106.2	113.1	113.4

(1) The CBB, ABB, C(0.8) and CABB methods are better than the BB1, BB2 and NBB methods, and the BB1 method is better than the BB2 and NBB methods. Out of 20 cases, there are 12, 16, 15 and 16 cases where the CBB, ABB, C(0.8) and CABB methods perform better than the BB1 method, respectively.

(2) The ABB, C(0.8) and CABB methods perform better than the CBB method. In addition, the CBB, C(0.8) and CABB methods outperform the ABB method in eight, 10 and 14 out of 20 cases, respectively.

In Table 2, we further report the average number of iterations required by the CBB(μ) methods with different fixed values of μ, in order to investigate the influence of the parameter μ on the performance of the CBB method. We select the value of μ from $\{0.1, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

The underlined numerical results in Table 2 indicate that the optimal value of μ is different for the test problems with different dimensions or condition numbers. The best choice of μ seems to be chosen in the interval $[0.7, 0.9]$.

To further evaluate the numerical performance of these algorithms, we observe the performance profiles of iteration and CPU time, which are measured by the function $\rho_s(\tau)$ proposed by Dolan and Moré [13]. In Figures 1 and 2, we show the numerical performance of the CBB, CBB(0.8), BB1, CABB, C(0.8) and ABB methods with

TABLE 2. Average NI of CBB(μ) with different μ.

		CBB(μ)							
n	Cd	0.1	0.2	0.4	0.5	0.6	0.7	0.8	0.9
10	10	$\underline{19}$	$\underline{19}$	19.4	19.7	19.4	19.2	19.2	18.5
10	10^2	59	62.5	63.4	59.3	55.4	53.6	52.5	$\underline{49.7}$
10	10^3	209.4	161.6	129	117.9	113.9	110.4	102.4	$\underline{89.5}$
10	10^4	373.7	257.5	248.2	190	173.2	$\underline{140.5}$	144.9	157.6
10	10^5	43.8	42.6	35.4	36.9	35.1	28.6	30.3	$\underline{25.5}$
10^2	10	19.7	19.8	19.4	19.4	19.3	19.3	$\underline{19.2}$	19.5
10^2	10^2	57.8	62.5	68.2	59.2	59.2	60	$\underline{52.9}$	53.4
10^2	10^3	161	167.4	140.9	123.6	122.1	116	$\underline{97}$	114.1
10^2	10^4	235.9	216.8	186.5	203.3	191.6	174.7	163.7	$\underline{152.5}$
10^2	10^5	80.4	83.5	78.2	70.3	71.2	68.4	$\underline{67.8}$	68.1
10^3	10	19.2	19.5	19.2	19.1	19	$\underline{18.9}$	19.1	19
10^3	10^2	61.7	63.5	60.7	57.8	58.7	57.9	$\underline{54.6}$	55.3
10^3	10^3	165.8	149.3	137.6	125.6	123.9	127.3	117.1	$\underline{110.8}$
10^3	10^4	167.4	144.1	129.2	130.7	131.8	116.7	114.3	$\underline{110.6}$
10^3	10^5	167	156.2	140.5	125.3	122.8	122.9	$\underline{103.4}$	110.6
10^4	10	19.1	19.3	19.1	19	19	$\underline{18.7}$	18.9	19
10^4	10^2	67.1	61.3	59.1	60.6	60.1	58.7	53.7	$\underline{52.5}$
10^4	10^3	142.1	134.3	130.4	127.8	132.1	113.5	$\underline{107.8}$	116
10^4	10^4	160.2	157.4	139.1	142.3	133.1	115.7	119.5	$\underline{113.6}$
10^4	10^5	173.2	170.1	142.2	146.3	140.2	124.3	$\underline{118.8}$	131.9

respect to the number of iterations by using the average number of iterations reported in Tables 1 and 2. In Figures 3 and 4, we show the numerical performance of the CBB, CBB(0.8), BB1, CABB, C(0.8) and ABB methods with respect to the CPU time. For Figures 3 and 4, we test problem (4.1) in the cases $n = 1000, 10\,000$ and $Cd = 10, 10^2, 10^3, 10^4$ and 10^5. And, for each case, we randomly generated 10 test problems.

In Figures 1–4, the vertical axis gives the probability P of problems for that any given method is within a factor τ of the best possible ratio. The left-hand axis of the plot gives the percentage of the test problems for which a method needs the fewest iterations or least CPU time. The right-hand side of the plot gives the percentage of the test problems that are successfully solved by each of the methods. Dolan and Moré [13] pointed out that if the set of test problems is suitably large and representative of problems that are likely to occur in applications, the methods with large probability $P(\tau)$ are to be preferred.

From Figures 1 and 3, it is clear that the CBB method outperforms the BB1 and CBB(μ) methods. Therefore, we conclude that the new step size α_k^{CBB} did improve the numerical efficiency of the BB methods.

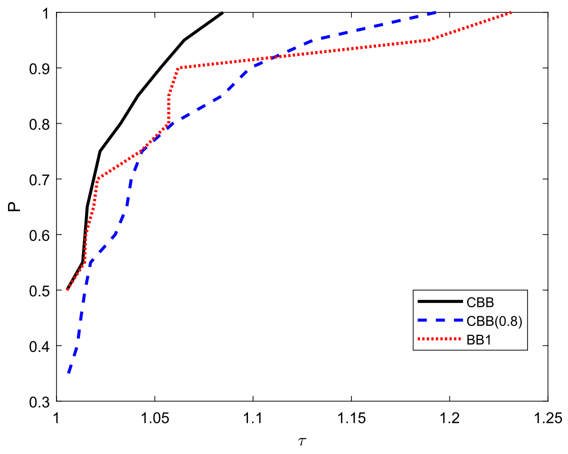

FIGURE 1. Performance profiles of iteration number for CBB, CBB(0.8) and BB1.

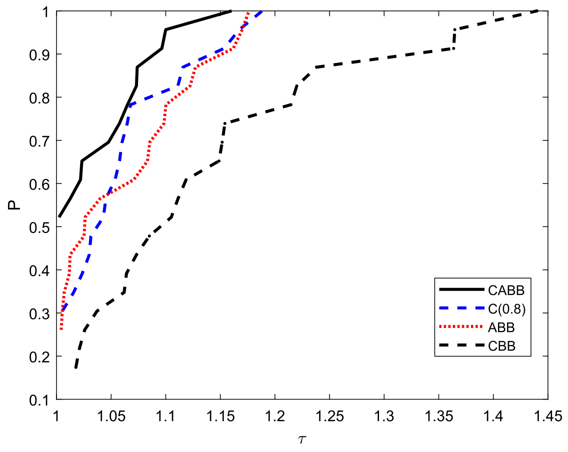

FIGURE 2. Performance profiles of iteration number for CABB, C(0.8), ABB and CBB.

From Figures 2 and 4, it is easy to see that the numerical performances of the CABB, C(0.8) and ABB methods are much better than the CBB method. In general, compared with the ABB and C(0.8) methods, the CABB method performs better.

4.2. Test by generic nonlinear problems At the end of this section, we test numerical performance of the BB1, CBB, CBB(0.8), ABB, CABB and C(0.8) methods as they are used to solve the generic nonlinear benchmark test problems (nonquadratic minimization problems) of Andrei [1]. For each problem, the initial point has been fixed as in [1]. The initial step size is generated by the Armijo linear search [17, 18]

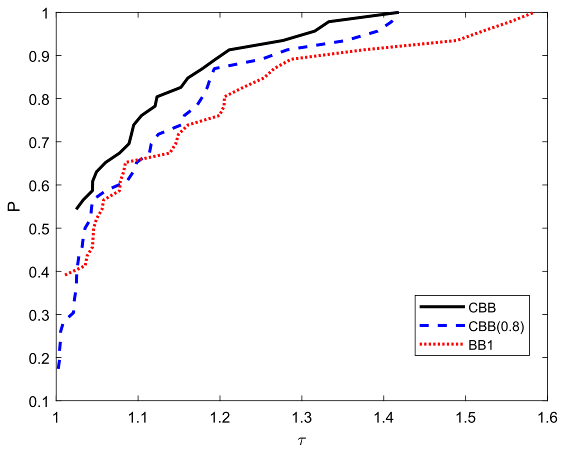

FIGURE 3. Performance profiles of CPU time for CBB, CBB(0.8) and BB1.

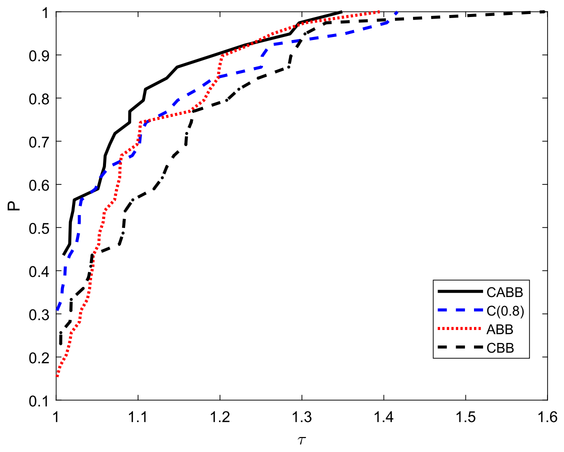

FIGURE 4. Performance profiles of CPU time for CABB, C(0.8), ABB and CBB.

and the termination condition for all the algorithms is $\|g_k\|_2 \le 10^{-6}$. All of the 14 generic nonlinear benchmark test problems of Andrei [1] are solved by the BB1, CBB, CBB(0.8), ABB, CABB and C(0.8) methods. In Table 3, we report the number of iterations, the minimal value of the objective function and the CPU time, respectively.

The numerical results in Table 3 demonstrate that all the six BB methods can successfully solve generic large-scale nonlinear (nonquadratic) optimization problems. Since these algorithms are used to solve problems 5, 6, 18, 29, 41, 49 and 53, it follows from the numerical performance that each method has its respective advantages. The underlined numerical results in Table 3 indicate that the CBB(0.8) and CBB (or C(0.8) and CABB) methods outperform the BB1 (or ABB) method.

TABLE 3. Numerical results for the test problems in [1].

P	Algorithm	n	NI	$f_{\min}$	T
5	BB1	100 000	F	F	F
	CBB	100 000	F	F	F
	CBB(0.8)	100 000	F	F	F
	ABB	100 000	109	8.461e-19	3.301
	CABB	100 000	101	7.233e-14	3.219
	C(0.8)	100 000	81	5.694e-18	2.640
6	BB1	100 000	55	1.855e-15	3.693
	CBB	100 000	65	7.722e-16	3.351
	CBB(0.8)	100 000	70	7.026e-17	2.846
	ABB	100 000	41	1.288e-11	1.802
	CABB	100 000	47	2.851e-16	2.024
	C(0.8)	100 000	52	2.796e-13	2.214
18	BB1	100	78	2.357e-15	0.017
	CBB	100	60	8.879e-16	0.009
	CBB(0.8)	100	F	F	F
	ABB	100	F	F	F
	CABB	100	F	F	F
	C(0.8)	100	F	F	F
19	BB1	100 000	7	2.174e-20	0.671
	CBB	100 000	7	1.238e-13	0.495
	CBB(0.8)	100 000	7	2.815e-16	0.594
	ABB	100 000	7	2.174e-20	0.542
	CABB	100 000	7	2.815e-16	0.498
	C(0.8)	100 000	7	1.385e-13	0.428
21	BB1	100 000	14	1.429e-17	0.710
	CBB	100 000	14	8.728e-18	0.572
	CBB(0.8)	100 000	14	1.258e-21	0.696
	ABB	100 000	14	1.429e-17	0.641
	CABB	100 000	14	1.258e-21	0.578
	C(0.8)	100 000	14	8.728e-18	0.530

25	BB1	100 000	310	1.821e-8	6.317
	CBB	100 000	811	3.951e-8	13.138
	CBB(0.8)	100 000	191	1.391e-8	3.072
	ABB	100 000	278	3.588e-8	4.338
	CABB	100 000	250	3.810e-8	4.254
	C(0.8)	100 000	239	2.458e-8	3.734
29	BB1	1000	210	482.5	0.067
	CBB	1000	113	482.5	0.041
	CBB(0.8)	1000	140	482.5	0.052
	ABB	1000	213	−123.9	0.070
	CABB	1000	205	−123.9	0.066
	C(0.8)	1000	220	−123.9	0.070
31	BB1	10 000	927	3.610e-08	1.027
	CBB	10 000	870	4.727e-08	0.9875
	CBB(0.8)	10 000	700	3.506e-08	0.730
	ABB	10 000	1145	2.931e-08	1.192
	CABB	10 000	1277	4.012e-08	1.315
	C(0.8)	10 000	1271	3.502e-08	1.682
37	BB1	10 000	1431	−1	1.335
	CBB	10 000	1305	−1	0.969
	CBB(0.8)	10 000	1413	−1	1.043
	ABB	10 000	940	−1	0.658
	CABB	10 000	1146	−1	0.772
	C(0.8)	10 000	991	−1	0.681
41	BB1	100	F	F	F
	CBB	100	2684	1.740e-12	0.197
	CBB(0.8)	100	1213	6.330e-14	0.109
	ABB	100	F	F	F
	CABB	100	F	F	F
	C(0.8)	100	F	F	F

43	BB1	1000	2108	1.931e-13	0.360
	CBB	1000	1729	3.355e-15	0.349
	CBB(0.8)	1000	1546	2.914e-14	0.251
	ABB	1000	1140	8.831e-18	0.199
	CABB	1000	1130	3.008e-13	0.221
	C(0.8)	1000	1058	2.544e-15	0.185
49	BB1	1000	F	F	F
	CBB	1000	1551	−781.9	0.428
	CBB(0.8)	1000	F	F	F
	ABB	1000	F	F	F
	CABB	1000	F	F	F
	C(0.8)	1000	F	F	F
53	BB1	1000	61	1.070e-14	0.026
	CBB	1000	62	4.818e-15	0.025
	CBB(0.8)	1000	46	4.611e-15	0.021
	ABB	1000	F	F	F
	CABB	1000	F	F	F
	C(0.8)	1000	47	2.729e-14	0.019
55	BB1	1000	447	1121e-13	0.111
	CBB	1000	382	1.629e-13	0.099
	CBB(0.8)	1000	519	2.033e-13	0.141
	ABB	1000	327	1.013e-13	0.087
	CABB	1000	321	1.381e-13	0.087
	C(0.8)	1000	365	1.001e-13	0.090

5. Conclusions

In this paper, we have proposed a new adaptive and composite BB step size, which is an optimal weighted mean of the classical BB step sizes. Combining the steepest descent direction and this new step size, the so-called CBB algorithm has been developed.

We have established the global convergence of the CBB and CABB methods, as they are applied to solve the quadratic minimization problems. The R-linear convergence theorem has been established for the CBB and CABB methods.

Numerical experiments demonstrate that the developed algorithm in this paper outperforms the similar ones available in the literature, since it is used to solve ill-posed large-scale quadratic minimization problems or generic nonlinear test problems. In summary, the new step size is useful to improve the numerical efficiency of algorithms, especially for solving ill-posed or large-scale optimization problems.

Acknowledgements

The authors would like to express their thanks to the anonymous referees for their constructive comments on the paper, which have greatly improved its presentation. This research was supported by the National Natural Science Foundation of China (Grant No. 71671190).

References

[1] N. Andrei, "An unconstrained optimization test functions collection", *Adv. Model. Optim.* **10** (2008) 147–161; doi:10.1021/es702781x.

[2] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods", *IMA J. Numer. Anal.* **8** (1988) 141–148; doi:10.1093/imanum/8.1.141.

[3] E. G. Birgin, J. M. Martínez and M. Raydan, "Nonmonotone spectral projected gradient methods on convex sets", *SIAM J. Optim.* **10** (2000) 1196–1211; doi:10.1137/S1052623497330963.

[4] S. Bonettini, R. Zanella and L. Zanni, "A scaled gradient projection method for constrained image deblurring", *Inverse Problems* **25** (2009) 015002; doi:10.1088/0266-5611/25/1/015002.

[5] X. R. Chen, Y. M. Liu and Z. Wan, "Optimal decision-making for the online and offline retailers under BOPS model", *ANZIAM J.* **58** (2016) 187–208; doi:10.1017/S1446181116000201.

[6] W. Cheng and D. H. Li, "A derivative-free nonmonotone line search and its application to the spectral residual method", *IMA J. Numer. Anal.* **29** (2009) 814–825; doi:10.1093/imanum/drn019.

[7] Y. H. Dai, "Alternate step gradient method", *Optimization* **52** (2003) 395–415; doi:10.1080/02331930310001611547.

[8] Y. H. Dai, M. Al-Baali and X. Yang, "A positive Barzilai–Borwein-like stepsize and an extension for symmetric linear systems", in: *Numerical Analysis and Optimization*, Volume 431 of *Springer Proceedings in Mathematics and Statistics Series* (Springer, Cham, 2015) 59–75.

[9] Y. H. Dai and R. Fletcher, "Projected Barzilai–Borwein methods for large-scale box-constrained quadratic programming", *Numer. Math.* **100** (2005) 21–47; doi:10.1007/s00211-004-0569-y.

[10] Y. H. Dai and L. Z. Liao, "R-linear convergence of the Barzilai and Borwein gradient method", *IMA J. Numer. Anal.* **22** (2002) 1–10; doi:10.1093/imanum/22.1.1.

[11] S. Deng and Z. Wan, "A three-term conjugate gradient algorithm for large-scale unconstrained optimization problems", *Appl. Numer. Math.* **92** (2015) 70–81; doi:10.1016/j.apnum.2015.01.008.

[12] J. E. Dennis and J. J. Moré, "Quasi-Newton methods, motivation and theory", *Siam Rev.* **19** (1977) 46–89; doi:10.1137/1019005.

[13] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles", *Math. Program.* **91** (2002) 201–213; doi:10.1007/s101070100263.

[14] M. A. Figueiredo, R. D. Nowak and S. J. Wright, "Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems", *IEEE J. Sel. Top. Signal Process.* **1** (2007) 586–597; doi:10.1109/jstsp.2007.910281.

[15] Y. Huang, H. Liu and S. Zhou, "Quadratic regularization projected Barzilai–Borwein method for nonnegative matrix factorization", *Data Min. Knowl. Discov.* **29** (2014) 1665–1684; doi:10.1007/s10618-014-0390-x.

[16] Y. Huang, H. Liu and S. Zhou, "An efficient monotone projected Barzilai–Borwein method for nonnegative matrix factorization", *Appl. Math. Lett.* **45** (2015) 12–17; doi:10.1016/j.aml.2015.01.003.

[17] S. Huang and Z. Wan, "A new nonmonotone spectral residual method for nonsmooth nonlinear equations", *J. Comput. Appl. Math.* **313** (2017) 82–101; doi:10.1016/j.cam.2016.09.014.

[18] S. Huang, Z. Wan and J. Zhang, “An extended nonmonotone line search technique for large-scale unconstrained optimization”, *J. Comput. Appl. Math.* **330** (2018) 586–604; doi:10.1016/j.cam.2017.09.026.

[19] W. La Cruz, J. M. Martínez and M. Raydan, “Spectral residual method without gradient information for solving large-scale nonlinear systems of equations”, *Math. Comput.* **75** (2006) 1429–1448; doi:10.1090/s0025-5718-06-01840-0.

[20] Y. X. Li and Z. Wan, “Bi-level programming approach to optimal strategy for VMI problems under random demand”, *ANZIAM J.* **59** (2017) 247–270; doi:10.1017/S1446181117000384.

[21] M. Raydan, “On the Barzilai and Borwein choice of steplength for the gradient method”, *IMA J. Numer. Anal.* **13** (1993) 321–326; doi:10.1093/imanum/13.3.321.

[22] K. Sopyła and P. Drozda, “Stochastic gradient descent with Barzilai–Borwein update step for SVM”, *Inform. Sci.* **316** (2015) 218–233; doi:10.1016/j.ins.2015.03.073.

[23] W. Sun and Y. X. Yuan, *Optimization theory and methods: nonlinear programming* (Springer, New York, 2006); doi:10.1007/b106451.

[24] Z. Wan, Y. Chen, S. Huang and D. Feng, “A modified nonmonotone BFGS algorithm for solving smooth nonlinear equations”, *Optim. Lett.* **8** (2014) 1845–1860; doi:10.1007/s11590-013-0678-6.

[25] Z. Wan, J. Guo, J. Liu and W. Liu, “A modified spectral conjugate gradient projection method for signal recovery”, *Signal Image Video Process.* **12** (2018) 1455–1462; doi:10.1007/s11760-018-1300-2.

[26] H. Wu and Z. Wan, “A multiobjective optimization model and an orthogonal design-based hybrid heuristic algorithm for regional urban mining management problems”, *J. Air Waste Manag. Assoc.* **68** (2017) 146–169; doi:10.1080/10962247.2017.1386141.

[27] Y. Yuan, “Gradient methods for large scale convex quadratic functions”, in: *Optimization and regularization for computational inverse problems and applications* (Springer, Berlin, Heidelberg, 2010) 141–155; doi:10.1007/978-3-642-13742-6_7.

[28] X. B. Zhang, S. Huang and Z. Wan, “Optimal pricing and ordering in global supply chain management with constraints under random demand”, *Appl. Math. Model.* **40** (2016) 10105–10130; doi:10.1016/j.apm.2016.06.054.

[29] X. B. Zhang, S. Huang and Z. Wan, “Stochastic programming approach to global supply chain management under random additive demand”, *Oper. Res.* **18** (2018) 389–420; doi:10.1007/s12351-016-0269-2.

[30] B. Zhou, L. Gao and Y. H. Dai, “Gradient methods with adaptive step-sizes”, *Comput. Optim. Appl.* **35** (2006) 69–86; doi:10.1007/s10589-006-6446-0.