Abstract A new stepsize is derived for the gradient method. It is shown that this stepsize asymptotically converges to the reciprocal of the largest eigenvalue of the Hessian of the objective function. Based on this spectral property, we develop a monotone gradient method that takes a certain number of steps with the asymptotic optimal stepsize proposed by Dai and Yang (Computational Optimization and Applications, 2006, 33(1): 73-88) followed by some short steps associated with the new stepsize. By employing one step retard of the asymptotic optimal stepsize, a nonmonotone variant is suggested. $R$-linear convergences are established for the proposed methods. Numerical comparisons with other recent successful methods demonstrate the efficiency of our methods.

Keywords gradient method · spectral property · convergence · quadratic optimization
1 Introduction

We consider the problem of minimizing a convex quadratic function

$$\min f(x) = \frac{1}{2} x^T A x - b^T x,$$

(1)

where $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n \times n}$ is symmetric positive definite with eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and condition number $\kappa = \frac{\lambda_n}{\lambda_1}$. This problem is one of the simplest non-trivial non-linear programming problems. Solving problem (1) is usually a pre-requisite for a method to be generalized to solve more general problems. In addition, various optimization problems arising in many applications including machine learning [6], sparse reconstruction [18], nonnegative matrix factorization [25,27] can be formulated as the form of (1), possibly with the addition of regularization or bound constraints.

The simplest method for solving (1) is the gradient method which updates iterates by

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

(2)

where $g_k = \nabla f(x_k)$ and $\alpha_k > 0$ is the stepsize determined in some way.

The classic steepest descent (SD) method can be dated back to Cauchy [5], who suggested to compute the stepsize by exact line search:

$$\alpha_{SD}^k = \arg \min_{\alpha} f(x_k - \alpha g_k) = \frac{g_k^T g_k}{g_k^T A g_k}.$$

(3)

It has been shown that the method converges linearly [1] with $Q$-linear factor $\frac{\kappa - 1}{\kappa + 1}$. Thus, the SD method can be very slow especially when the condition number is large. Further analysis shows that the gradients will asymptotically produce zigzags between two orthogonal directions associated with the two eigenvectors corresponding to $\lambda_1$ and $\lambda_n$, see [20,29] for more details.

In 1988, from the point of view of quasi-Newton method, Barzilai and Borwein [2] designed the following two ingenious stepsizes that significantly improve the performance of gradient methods:

$$\alpha_{BB1}^k = \frac{s_k^T s_{k-1}}{s_{k-1}^T s_{k-1}} \alpha_{BB2}^k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}},$$

(4)

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. The BB method was shown to be global convergent for general $n$-dimensional strictly convex quadratics [30] and the convergence rate is $R$-linear [11]. Recently, the BB method and its variants have successfully been extended to general unconstrained problems [31], to constrained optimization problems [3,24] and to various applications [25,26,28], see also [4,10,16,19,33] and references therein.

Let $\{\xi_1, \xi_2, \ldots, \xi_n\}$ be the orthonormal eigenvectors associated with the eigenvalues. Denote the components of $g_k$ along the eigenvectors $\xi_i$ by $\mu_i^k$, $i = 1, \ldots, n$, i.e.,

$$g_k = \sum_{i=1}^n \mu_i^k \xi_i,$$
which together with the update rule (2) gives that
\[ g_{k+1} = g_k - \alpha_k A g_k = \prod_{j=1}^{k} (I - \alpha_j A) g_1 = \sum_{i=1}^{n} \mu^{k+1}_i \xi_i, \]
where
\[ \mu^{k+1}_i = \mu^k_i (1 - \alpha_k \lambda_i) = \mu^1_i \prod_{j=1}^{k} (1 - \alpha_j \lambda_i), \]
which implies that the closer \( \alpha_k \) to 1, the smaller \( |\mu^{k+1}_i| \). In addition, if \( \mu^k_i = 0 \), the corresponding component vanishes at all subsequent iterations.

Since the SD method will asymptotically zigzag between \( \xi_1 \) and \( \xi_n \), a natural way to break the zigzagging pattern is eliminating the component \( \mu^n_k \) or \( \mu^n_k \) which can be achieved by employing a stepsize approximates \( \frac{1}{\lambda_1} \) or \( \frac{1}{\lambda_n} \).

One pioneer work in this line is due to Yuan [14,32] who derived the following stepsize by imposing finite termination for two-dimensional convex quadratics:
\[ \alpha^Y_k = \frac{2}{\sqrt{(1/\alpha^{SD}_{k-1} - 1/\alpha^{SD}_k)^2 + 4 \|g_k\|^2 / (\alpha^{SD}_{k-1} \|g_{k-1}\|^2) + (1/\alpha^{SD}_{k-1} + 1/\alpha^{SD}_k)}}. \]  
(5)

Dai and Yuan [14] further suggested a new gradient method whose stepsize is given by
\[ \alpha^{DY}_k = \begin{cases} \alpha^{SD}_k, & \text{if } \text{mod}(k,4) < 2; \\ \alpha^Y_t, & \text{otherwise}. \end{cases} \]  
(6)

The DY method (6) keeps monotonicity and appears better than the non-monotone BB method, see [14]. It was shown by De Asmundis et al. [15] that the stepsize \( \alpha^X_k \) converges to \( \frac{1}{\lambda_n} \) if the SD method is applied to problem (1).

That is, occasionally employing the stepsize \( \alpha^Y_k \) along the SD method will enhance the elimination of the component \( \mu^k_i \). Recently, Gonzaga and Schneider [23] suggested a monotone method where all stepsizes are of the form (3). Particularly, their method approximates \( \frac{1}{\lambda_n} \) by a short stepsize calculated by replacing \( g_k \) in (3) with \( \tilde{g} = (I - \eta A) g_k \), where \( \eta \) is a large number.

Nonmonotone gradient methods exploit spectral properties have been developed as well. Frassoldati et al. [21] developed a new short stepsize by maximizing the next SD stepsize \( \alpha^{SD}_{k+1} \) according to the current stepsize. They further suggested the ABB\textsubscript{min2} method which tries to exploit BB1 stepsizes close to \( \frac{1}{\lambda_1} \) by using short stepsizes to eliminate those components associated with large eigenvalues to force the gradient to be dominated by components corresponding to small eigenvalues. More recently, based on the favourable property of \( \alpha^Y_k \), De Asmundis et al. [15] suggested to reuse it in a cyclic fashion after a certain number of SD steps. Precisely, their approach, referred to as the SDC method, employs the stepsize
\[ \alpha^{SDC}_k = \begin{cases} \alpha^{SD}_k, & \text{if } \text{mod}(k, h + s) < h; \\ \alpha^Y_t, & \text{otherwise, with } t = \max\{i \leq k : \text{mod}(i, h + s) = h\}, \end{cases} \]  
(7)
where $h \geq 2$ and $s \geq 1$. They also proposed a monotone version of (7) by imposing safeguards on the stepsizes.

One common characteristic of the aforementioned methods is making use of spectral properties of the stepsizes. The recent study in [16] points out that gradient methods using long and short stepsizes that attempt to exploit their spectral properties show better numerical behaviour both in the quadratic and general cases. For more works on gradient methods, see [7,9,15,16,21,23,33,34] and references therein.

Recently, Dai and Yang [12] introduced a gradient method with a new stepsize that possesses similar spectral property as $\alpha_k^Y$. In particular, their stepsize is calculated by

$$\alpha_k^{AOPT} = \frac{\|g_k\|}{\|Ag_k\|}, \quad (8)$$

which asymptotically converges to $\frac{2}{\lambda_1 + \lambda_n}$ that is optimal in the sense that it minimizes $\|I - \alpha A\|$, see for example [12,17]. Since $\alpha_k^{AOPT} \leq \alpha_k^{SD}$, the Dai-Yang method (8) is monotone but without exact line searches. In addition, the method converges Q-linearly and enjoys the same rate as the SD method. More importantly, it is able to recover the eigenvectors $\xi_1$ and $\xi_n$.

In this paper, based on the Dai-Yang method (8), we propose a new stepsize to exploit the spectral property. Particularly, our new stepsize is given by

$$\bar{\alpha}_k = \frac{d_k^T d_k}{d_k^T Ad_k}, \quad (9)$$

where

$$d_k = \frac{g_{k-1}}{\|g_{k-1}\|} - \frac{g_k}{\|g_k\|}. \quad (10)$$

We show that the stepsize $\bar{\alpha}_k$ asymptotically converges to $\frac{1}{\lambda_n}$ if the Dai-Yang method (8) is applied to problem (1). Therefore, the stepsize $\bar{\alpha}_k$ is helpful in eliminating the component $\mu_n$. Thanks to this desired property, we are able to develop a new efficient gradient method by taking a certain number of steps with the asymptotic optimal stepsize $\alpha_k^{AOPT}$ followed by some short steps which are determined by the smaller one of $\alpha_k^{AOPT}$ and $\bar{\alpha}_{k-1}$. Thus, our method is monotone without exact line searches. We also consider a nonmonotone variant of the method which is same as the monotone one except using the stepsize $\alpha_k^{AOPT}$ with one step retard. $R$-linear convergences of the proposed methods are established. Preliminary numerical comparisons with the DY, ABB$_{min 2}$ and SDC methods are presented to demonstrate the efficiency of our methods.

The paper is organized as follows. In Section 2 we analyze asymptotic spectral property of the stepsize $\bar{\alpha}_k$ and present the new method as well as its nonmonotone variant. In Section 3 we show the proposed methods are $R$-linearly convergent. In Section 4 we report some numerical comparisons of our methods and the DY, ABB$_{min 2}$ and SDC methods. Finally, a brief discussion is presented.
2 Proposed methods

In this section we first analyze the spectral property of the stepsize $\bar{\alpha}_k$ and then present our new gradient methods.

2.1 Spectral property of $\bar{\alpha}_k$

We recall important properties of the method (8).

**Lemma 1** [12] For any starting point $x_1$ satisfying

$$\mu_1^1 \neq 0, \quad \mu_n^1 \neq 0,$$

let $\{x_k\}$ be the iterations generated by the method (8). Then we have that

$$\lim_{k \to \infty} \alpha_k^{AOP} = \frac{2}{\lambda_1 + \lambda_n}.$$

Furthermore,

$$\lim_{k \to \infty} \frac{\mu_i^{2k-1}}{\sum_{j=1}^{n} (\mu_j^{2k-1})^2} = \begin{cases} \text{sign}(\mu_1^1)\sqrt{c_1}, & \text{if } i = 1; \\ 0, & \text{if } i = 2, \ldots, n-1; \\ \text{sign}(\mu_n^1)\sqrt{c_2}, & \text{if } i = n, \end{cases}$$

and

$$\lim_{k \to \infty} \frac{\mu_i^{2k}}{\sum_{j=1}^{n} (\mu_j^{2k})^2} = \begin{cases} \text{sign}(\mu_1^1)\sqrt{c_1}, & \text{if } i = 1; \\ 0, & \text{if } i = 2, \ldots, n-1; \\ -\text{sign}(\mu_n^1)\sqrt{c_2}, & \text{if } i = n, \end{cases}$$

which indicates that

$$\lim_{k \to \infty} \frac{g_{k-1}}{\|g_k\|} + \frac{g_k}{\|g_k\|} = 2\text{sign}(\mu_1^1)\sqrt{c_1}\xi_1$$

and

$$\lim_{k \to \infty} \frac{g_{k-1}}{\|g_k\|} - \frac{g_k}{\|g_k\|} = \pm 2\sqrt{c_2}\xi_n,$$

where

$$c_1 = \frac{\lambda_1 + 3\lambda_n}{4(\lambda_1 + \lambda_n)}, \quad c_2 = \frac{3\lambda_1 + \lambda_n}{4(\lambda_1 + \lambda_n)}.$$

Lemma 1 indicates that the method (8) asymptotically reduces its search in the two-dimensional subspace spanned by $\xi_1$ and $\xi_n$. In order to accelerate the method, we can employ some stepsize approximates $\frac{1}{\lambda_1}$ or $\frac{1}{\lambda_n}$ to eliminate the component $\mu_1^k$ or $\mu_n^k$. Note that the vector $d_k$ tends to align the eigenvector $\xi_n$. If we take some gradient steps with $\alpha_k^{AOP}$ such that $d_k \approx \pm 2\sqrt{c_2}\xi_n$, the stepsize $\bar{\alpha}_k$ will be an approximation of $\frac{1}{\lambda_n}$. The next theorem provides theoretical evidence for this.
Theorem 1 Under the conditions in Lemma 1, let \( \{g_k\} \) be the sequence generated by applying the method (8) to problem (1). Then we have

\[
\lim_{k \to \infty} \bar{\alpha}_k = \frac{1}{\lambda_n}.
\]

Proof From the definition (10) of \( d_k \), we have

\[
d_k^T d_k = 2 - 2 \frac{g_k^T g_{k-1}}{\|g_{k-1}\| \|g_k\|}
\]

and

\[
d_k^T A d_k = \frac{g_k^T A g_{k-1}}{\|g_{k-1}\|^2} + \frac{g_k^T A g_k}{\|g_k\|^2} - 2 \frac{g_k^T A g_{k-1}}{\|g_{k-1}\| \|g_k\|}.
\]

which indicate that

\[
\lim_{k \to \infty} d_k^T d_k = 2 - 2 \lim_{k \to \infty} \sum_{j=1}^{n} \frac{\mu_i^{2k-1} \mu_j^{2k}}{\sqrt{\sum_{j=1}^{n} (\mu_j^{2k-1})^2} \sqrt{\sum_{j=1}^{n} (\mu_j^{2k})^2}}
\]

\[
= 2 - 2(c_1 - c_2)
\]

and

\[
\lim_{k \to \infty} d_k^T A d_k = \lim_{k \to \infty} \sum_{j=1}^{n} \frac{\mu_i^{2k-1} \mu_j^{2k}}{\sqrt{\sum_{j=1}^{n} (\mu_j^{2k-1})^2} \sqrt{\sum_{j=1}^{n} (\mu_j^{2k})^2}}
\]

\[
= 2(\lambda_1 c_1 + \lambda_n c_2) - 2(\lambda_1 c_1 - \lambda_n c_2)
\]

\[
= 4\lambda_n c_2.
\]

It follows from (13), (14) and the definition (9) of \( \bar{\alpha}_k \) that

\[
\lim_{k \to \infty} \bar{\alpha}_k = \lim_{k \to \infty} \frac{d_k^T d_k}{d_k^T A d_k} = \frac{2 - 2(c_1 - c_2)}{4\lambda_n c_2}
\]

\[
= \frac{4(\lambda_1 + \lambda_n) - (2\lambda_n - 2\lambda_1)}{2\lambda_n(3\lambda_1 + \lambda_n)} = \frac{1}{\lambda_n}.
\]

This completes the proof. \( \square \)

Using the same argument as the one in Theorem 1, we get the following result.
**Theorem 2** Under the conditions in Lemma 1, let \( \{g_k\} \) be the sequence generated by applying the method (8) to problem (1), we have

\[
\lim_{k \to \infty} \hat{\alpha}_k = \frac{1}{\lambda_1},
\]

where

\[
\hat{\alpha}_k = \frac{\hat{d}_k^T \hat{d}_k}{\hat{d}_k^T A \hat{d}_k}
\]

with

\[
\hat{d}_k = \frac{g_{k-1}}{\|g_{k-1}\|} + \frac{g_k}{\|g_k\|}.
\]

2.2 The algorithm

Theorems 1 and 2 in the former subsection provide us the possibility of employing the two stepsizes \( \hat{\alpha}_k \) and \( \bar{\alpha}_k \) to vanish the components \( \mu_1^k \) and \( \mu_n^k \). However, the following example shows negative aspects of using \( \hat{\alpha}_k \). Particularly, we applied the method (8) to a problem in the form of (1) with

\[
A = \text{diag}\{a_1, a_2, \ldots, a_n\}, \quad b = 0,
\]

where \( a_1 = 1, a_n = n \) and \( a_i \) is randomly generated in \((1, n)\), \( i = 2, \ldots, n - 1 \). Figure 1 presents the result of an instance with \( n = 1000 \). We can see that \( \bar{\alpha}_k \) approximates \( \frac{1}{\lambda_n} \) with satisfactory accuracy in few iterations. Although we did observe that after 1000 iterations the value of \( |\hat{\alpha}_k - \frac{1}{\lambda_1}| \) is reduced by a factor of 0.01, \( \hat{\alpha}_k \) converges to \( \frac{1}{\lambda_1} \) very slowly in the first few hundreds of iterations.

Now we give a rough explanation of the above phenomenon. Since the gradient norm decreases very slowly, by the update rule (2) we have that

\[
\frac{g_k^{(i)}}{\|g_k\|} = (1 - \alpha_k \lambda_i) \frac{g_{k-1}^{(i)}}{\|g_{k-1}\|} \approx (1 - \alpha_k \lambda_i) \frac{g_{k-1}^{(i)}}{\|g_{k-1}\|}.
\]

Suppose that \( \alpha_{k_0} \) satisfies \( |1 - \alpha_{k_0} \lambda_i| \leq 1 \) for all \( i = 1, 2, \ldots, n \) and \( k \geq k_0 \). This will be true since \( \alpha_k^{OPT} \) approximates \( \frac{2}{\lambda_1 + \lambda_n} \) as the iteration process goes on. Let \( d_k^{(i)} \) be the \( i \)-th component of \( d_k \), i.e.,

\[
d_k^{(i)} = \frac{g_k^{(i)}}{\|g_k\|} - \frac{g_{k-1}^{(i)}}{\|g_{k-1}\|}.
\]

For the stepsize \( \hat{\alpha}_k, k \geq k_0 \), we consider the following two cases:

**Case 1.** \( |1 - \alpha_k \lambda_i| \leq 0.6 \).

By trivial computation we know that after five steps the value of \( \frac{g_k^{(i)}}{\|g_k\|} \) is less than 8% of its initial value and keeps small at all subsequent iterations, thus \( d_k^{(i)} \) can be neglected.

**Case 2.** \( |1 - \alpha_k \lambda_i| > 0.6 \).
(i) If \(1 - \alpha_k \lambda_i \geq 0.9\), the value of \(\frac{g^{(i)}_k}{\|g_k\|}\) will not change much and thus \(d^{(i)}_k\) may not affect the value of \(\tilde{\alpha}_k\) too much which indicates that the component can also be neglected.

(ii) If \(0.6 < 1 - \alpha_k \lambda_i < 0.9\), i.e., \(0.1 < \alpha_k \lambda_i < 0.4\) which implies that \(d^{(i)}_k\) will be small in few iterations and is safe to be abandoned.

(iii) If \(1 - \alpha_k \lambda_i < -0.6\), the value of \(\frac{g^{(i)}_k}{\|g_k\|}\) changes signs and \(d^{(i)}_k\) may increase.

The above analysis shows that \(\tilde{\alpha}_k\) will only be dominated by the components corresponding to those eigenvalues in (iii) of Case 2. Notice that the required inequality in (iii) implies that \(\lambda_i > \frac{4(\lambda_1 + \lambda_n)}{5}\). If \(A\) has many large eigenvalues satisfy the condition, \(\tilde{\alpha}_k\) will be an estimation of the reciprocal of one of those eigenvalues which is also an approximation of \(\frac{1}{\lambda_n}\). Once \(A\) has few such large eigenvalues, \(\tilde{\alpha}_k\) will be dominated by the components corresponding to the first few largest eigenvalues which also yields a good estimation of \(\frac{1}{\lambda_n}\). In other words, \(\tilde{\alpha}_k\) will approximate \(\frac{1}{\lambda_n}\) with satisfactory accuracy in few iterations. This coincides our observation in Figure 1.

For the stepsize \(\hat{\alpha}_k\), when \(1 - \alpha_k \lambda_i > 0\), the value of \(\frac{g^{(i)}_k}{\|g_k\|} + \frac{g^{(i)}_{k-1}}{\|g_{k-1}\|}\) may increase even if \(1 - \alpha_k \lambda_i \leq 0.1\). That is, most of the components of the gradient corresponding to those eigenvalues less than \(\frac{1}{\alpha_k}\) will affect the value of \(\hat{\alpha}_k\). Thus, \(\hat{\alpha}_k\) would not be a good approximation of \(\frac{1}{\lambda_i}\) until those components become very small. Moreover, a rough estimation of \(\frac{1}{\lambda_i}\) may yield a large step which will increase most of the components of the gradient. As a result, it may impractical to use \(\hat{\alpha}_k\) for eliminating the component \(\mu^k_1\).
Based on the above observations, our method only combines \( \bar{\alpha}_k \) with \( \alpha_{k}^{AOPT} \). Note that, for quadratics, the BB1 stepsize \( \alpha_k^{BB1} \) is the former SD stepsize (3). Motivated by this and the good performance of the BB method, the stepsize \( \bar{\alpha}_k \) with one step delay is employed for the proposed method. As \( \bar{\alpha}_{k-1} \) is expected to be short to approximate \( \frac{1}{\lambda_i} \), we resort to \( \alpha_k^{AOPT} \) if \( \bar{\alpha}_{k-1} \) becomes large. More precisely, our method takes \( h \) steps with \( \alpha_k^{AOPT} \) and then \( s \) short steps, that is,

\[
\alpha_k = \begin{cases} 
\alpha_{k}^{AOPT}, & \text{if } \text{mod}(k, h + s) < h; \\
\min\{\alpha_{k}^{AOPT}, \bar{\alpha}_{k-1}\}, & \text{otherwise}.
\end{cases}
\]  

(16)

Remark 1 Although our method (16) looks like the SDC method (7), they differ in the following ways: (i) the method (16) does not use exact line searches which are necessary for the SDC method; (ii) the method (16) does not reuse any stepsize while the SDC method uses the same Yuan’s stepsize \( \alpha_k^{Y} \) for \( s \) steps; (iii) the method (16) is monotone while the SDC method is nonmonotone and its monotone version is obtained by using a safeguard with \( 2\alpha_k^{SD} \).

![Fig. 2](image-url) 

**Fig. 2** Problem (15): history of the sequence \( \{\bar{\alpha}_k\} \) for the first 100 iterations of the gradient method with \( \alpha_{k-1}^{AOPT} \).

The analysis in the beginning of this subsection indicates that \( d_k \) will be small if \( |1 - \alpha_k \lambda_i| \) is small. Thus, any stepsize approximates some \( \frac{1}{\lambda_i} \) is expected to enforce a small \( \bar{\alpha}_k \). It is natural to consider the retard stepsize \( \alpha_{k-1}^{AOPT} \) which has been analyzed in [8]. We can see from Figure 2 that, when the gradient method with \( \alpha_{k-1}^{AOPT} \) is applied to problem (15), the stepsize \( \bar{\alpha}_k \)
approximates $\frac{1}{\lambda_n}$ with high accuracy if its value is small. Motivated by this observation, we suggest the following nonmonotone variant:

$$\alpha_k = \begin{cases} 
\alpha_{k-1}^{OPT}, & \text{if } \text{mod}(k, h + s) < h; \\
\min\{\alpha_{k-1}^{OPT}, \bar{\alpha}_{k-1}\}, & \text{otherwise.}
\end{cases}$$ (17)

3 Convergence

In this section, we establish the $R$-linear convergence of our method (16) and its nonmonotone variant (17).

Since the gradient method (2) is invariant under translations and rotations when applying to problem (1), we make the following assumption throughout the analysis.

**Assumption 1.** The matrix $A$ is diagonal, i.e.,

$$A = \text{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_n\},$$ (18)

with $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_n$.

In order to give a uniform analysis of the methods (16) and (17), we resort to the following property proposed by Dai [7].

**Property (A)** [7]. Suppose that there exist an integer $m$ and positive constants $M_1 \geq \lambda_1$ and $M_2$ such that

(i) $\lambda_1 \leq \alpha_k^{-1} \leq M_1$;

(ii) for any integer $l \in [1, n-1]$ and $\epsilon > 0$, if $G(k-j, l) \leq \epsilon$ and $(g_{k-j}^{(l+1)})^2 \geq M_2 \epsilon$

hold for $j \in [0, \min\{k, m\} - 1]$, then $\alpha_k^{-1} \geq \frac{2}{3} \lambda_{l+1}$.

Here,

$$G(k, l) = \sum_{i=1}^{l} (g_k^{(i)})^2.$$

Dai [7] has proved that if $A$ has the form (18) with $1 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and the stepsizes of gradient method (2) have the Property (A), then either $g_k = 0$ for some finite $k$ or the sequence $\{\|g_k\|\}$ converges to zero $R$-linearly. Therefore, we show $R$-linear convergence of the proposed methods by proving that (16) and (17) satisfy Property (A).

**Theorem 3** Suppose that the sequence $\{\|g_k\|\}$ is generated by any of the method (16) or (17) applied to $n$-dimensional quadratics with the matrix $A$ has the form (18) and $1 = \lambda_1 < \lambda_2 < \cdots < \lambda_n$ then either $g_k = 0$ for some finite $k$ or the sequence $\{\|g_k\|\}$ converges to zero $R$-linearly.

**Proof** We show that the stepsize $\alpha_k$ has Property (A) with $m = 2$, $M_1 = \lambda_n$ and $M_2 = 2$.

Clearly, $\lambda_1 \leq \alpha_k^{-1} \leq \lambda_n$ for all $k \geq 1$. Thus, (i) of Property (A) holds with $M_1 = \lambda_n$. 


Notice that
\[(\alpha_k^{AOPT})^{-1} \geq (\alpha_k^{SD})^{-1}\]
and
\[(\alpha_{k-1}^{AOPT})^{-1} \geq (\alpha_{k-1}^{BB})^{-1} = (\alpha_{k-1}^{SD})^{-1}.
\]

We are suffice to show (ii) holds for \(\alpha_k^{BB}\). If \(G(k-j,l) \leq \epsilon\) and \((g_{k-j}^{(l+1)})^2 \geq 2\epsilon\) hold for \(j \in [0, \min\{k, m\} - 1]\), we have
\[(\alpha_k^{BB})^{-1}(\alpha_{k-j}^{SD})^{-1} = \sum_{i=1}^{n} \lambda_i (g_{k-j}^{(i)})^2 \geq \frac{\lambda_{l+1}}{\epsilon_l/2\epsilon_l + 1} \geq \frac{2}{3} \lambda_{l+1}.
\]

This completes the proof. \(\square\)

4 Numerical results

In this section, we compare the performance of our methods (16) and (17) with the DY method (6) in [14], with the ABB\textsubscript{min2} method in [21], and with the SDC method (7) in [15]. Since the SDC method performs better than its monotone counterpart for most problems, we only run SDC. All codes were written in Matlab (v.9.0-R2016a). All the runs were carried out on a laptop with an Intel Core i7, 2.9 GHz processor and 8 GB of RAM running Windows 10 system.

For all the test problems, the iteration was stopped if
\[
\|g_k\| \leq \epsilon \|g_1\|
\]
is satisfied with a given \(\epsilon\) or the iteration number exceeds 20,000. Based on the observation from Figures 1 and 2, we tested \(h\) with values 10 and 20 for our methods. As in [21], the parameter \(\tau\) used by the ABB\textsubscript{min2} method is set to 0.9 for all the problems.

Firstly, the compared methods were applied to 1000 dimensional quadratic problems with different spectral distributions. In particular, the problem used in [10,13,22,34] is employed, where \(A = QVQ^T\) with
\[
Q = (I - 2w_3w_3^T)(I - 2w_2w_2^T)(I - 2w_1w_1^T),
\]
and \(w_1, w_2,\) and \(w_3\) are unitary random vectors, \(V = \text{diag}(v_1, \ldots, v_n)\) is a diagonal matrix where \(v_1 = 1\) and \(v_n = \kappa\), and \(v_j\) is randomly generated between 1 and \(\kappa\) for \(j = 2, \ldots, n - 1\). The entries of \(b\) were randomly generated between -10 and 10. Five sets of different spectral distributions presented in
Table 1 were carried out. For each problem set, three different values of condition number $\kappa$ as well as three accuracies $\epsilon$ were tested. For each value of $\kappa$ or $\epsilon$, 10 instances were randomly generated. Table 2 presents the number of iterations averaged over those instances using the starting point $x_1 = (1, \ldots, 1)^T$ where the parameter pair $(h, s)$ used for the SDC method was set to $(8, 6)$ which is more efficient than other choices in our test.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Spectrum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{v_2, \ldots, v_{n-1}} \subset (1, \kappa)</td>
</tr>
<tr>
<td>2</td>
<td>{v_2, \ldots, v_{n/5}} \subset (1, 100) &lt;br/&gt;{v_{n/5+1}, \ldots, v_{n-1}} \subset (\frac{\kappa}{2}, \kappa)</td>
</tr>
<tr>
<td>3</td>
<td>{v_2, \ldots, v_{n/2}} \subset (1, 100) &lt;br/&gt;{v_{n/2+1}, \ldots, v_{n-1}} \subset (\frac{\kappa}{2}, \kappa)</td>
</tr>
<tr>
<td>4</td>
<td>{v_2, \ldots, v_{4n/5}} \subset (1, 100) &lt;br/&gt;{v_{4n/5+1}, \ldots, v_{n-1}} \subset (\frac{\kappa}{2}, \kappa)</td>
</tr>
<tr>
<td>5</td>
<td>{v_2, \ldots, v_{n/5}} \subset (1, 100) &lt;br/&gt;{v_{n/5+1}, \ldots, v_{4n/5}} \subset (100, \frac{\kappa}{2}) &lt;br/&gt;{v_{4n/5+1}, \ldots, v_{n-1}} \subset (\frac{\kappa}{2}, \kappa)</td>
</tr>
</tbody>
</table>

We can see from Table 2 that, our method (16) is very competitive with the DY, ABB$\text{min}_2$ and SDC methods. For a fixed $h$, larger values of $s$ seem to be preferable for our method (16). In addition, different settings of $s$ lead to comparable results, with differences of less than 10% in the number of iterations for most of the test problems. For the first problem set, our method (16) outperforms the DY and SDC methods although the ABB$\text{min}_2$ method is the fastest one among the compared methods. Particularly, when a high accuracy is required, the method (16) with $(h, s) = (20, 100)$ takes less than $\frac{1}{6}$ and $\frac{1}{4}$ iterations needed by the DY and SDC methods, respectively. As for the second to fourth problem sets, the method (16) with different settings performs better than the DY and ABB$\text{min}_2$ methods. Moreover, it is comparable to and even better than the SDC method if proper $h$ and $s$ are selected. Our method (16) also show evidence over the DY and SDC methods on the last problem set. The total number of iterations indicates the effectiveness of our method (16) as well. We have to point out that our method (16) and the DY method are monotone while the ABB$\text{min}_2$ and SDC methods are not. Moreover, the per-iteration cost of the ABB$\text{min}_2$ method is higher because except one matrix-vector it needs four vector-vector multiplications to compute its stepsize at each iteration while our method (16) only takes three.
Gradient methods exploiting spectral properties

Table 2 Number of averaged iterations of the methods (16), DY, ABB_{min 2} and SDC on problems in Table 1.

<table>
<thead>
<tr>
<th>Problem</th>
<th>( \epsilon )</th>
<th>((h, s)) for the method (16)</th>
<th>DY</th>
<th>ABB_{min 2}</th>
<th>SDC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10^{-5}</td>
<td>10^{-6}</td>
<td>10^{-10}</td>
<td>10^{-20}</td>
<td>10^{-5}</td>
</tr>
<tr>
<td>1</td>
<td>332.9</td>
<td>327.6</td>
<td>340.1</td>
<td>360.3</td>
<td>326.6</td>
</tr>
<tr>
<td>2</td>
<td>2325.4</td>
<td>1176.3</td>
<td>875.2</td>
<td>931.3</td>
<td>1326.9</td>
</tr>
<tr>
<td>3</td>
<td>6328.8</td>
<td>3797.9</td>
<td>2047.7</td>
<td>1348.0</td>
<td>1345.6</td>
</tr>
</tbody>
</table>

Table 3 presents averaged number of iterations of our method (17) on problems in Table 1. For comparison purposes, the results of the DY, ABB_{min 2} and SDC methods are restated here. Similar performance as the method (16) shown in Table 2 can be observed. In particular, for each accuracy level, our method (17) takes around 30% less total iterations than the ABB_{min 2} method and even less than those by the DY and SDC methods. Notice that problems of the last set are difficult for the compared methods since they require more iterations than other four sets. However, our method (17) always dominates the compared three methods except with the pair \((h, s) = (10, 20)\). As compared with the method (16), the retard strategy used in the method (17) tends to improve the performance when \(h = 10\). For the case \(h = 20\), the method (17) is comparable to and even better than (16) in terms of total iterations.

Secondly, we compared the methods on two large-scale real problems Laplacel(a) and Laplacel(b) described in [19]. Of the problems require the solution of an elliptic system of linear equations arising from a 3D Laplacian on a box, discretized using a standard 7-point finite difference stencil. The solution is fixed by a Gaussian function whose center is \((\alpha, \beta, \gamma)\), multiplied by \(x(x-1)\). A parameter \(\sigma\) is used to control the rate of decay of the Gaussian. Both Laplacel(a) and Laplacel(b) have \(n = N^3\) variables with \(N\) being the interior nodes taken in each coordinate direction and a highly sparse Hessian matrix with condition number \(10^{3.61}\). We refer the readers to [19] for more details on these problems. In our tests, the associated parameters are set as
Table 3 Number of averaged iterations of the methods (17), DY, ABB_{min 2} and SDC on problems in Table 1.

<table>
<thead>
<tr>
<th>problem</th>
<th>( \epsilon )</th>
<th>((h,s)) for the method (17)</th>
<th>DY</th>
<th>ABB_{min 2}</th>
<th>SDC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10^{-6}</td>
<td>(10, 20)</td>
<td>(10, 30)</td>
<td>(10, 50)</td>
<td>(10, 80)</td>
</tr>
<tr>
<td>1</td>
<td>10^{-6}</td>
<td>345.2</td>
<td>338.9</td>
<td>351.7</td>
<td>330.9</td>
</tr>
<tr>
<td>2</td>
<td>10^{-6}</td>
<td>1833.5</td>
<td>1537.8</td>
<td>1406.0</td>
<td>989.7</td>
</tr>
<tr>
<td>3</td>
<td>10^{-6}</td>
<td>1277.2</td>
<td>1247.4</td>
<td>1214.4</td>
<td>1216.1</td>
</tr>
<tr>
<td>4</td>
<td>10^{-6}</td>
<td>272.8</td>
<td>275.9</td>
<td>284.6</td>
<td>291.7</td>
</tr>
<tr>
<td>5</td>
<td>10^{-6}</td>
<td>1367.1</td>
<td>1356.7</td>
<td>1312.8</td>
<td>1350.1</td>
</tr>
<tr>
<td>total</td>
<td>10^{-6}</td>
<td>1969.7</td>
<td>1985.0</td>
<td>2034.7</td>
<td>2023.6</td>
</tr>
</tbody>
</table>

follows:

(a) \( \sigma = 20, \alpha = \beta = \gamma = 0.5 \);
(b) \( \sigma = 50, \alpha = 0.4, \beta = 0.7, \gamma = 0.5 \).

We used the null vector as the initial point. The number of iterations required by the compared methods on the two problems Laplace1(a) and Laplace1(b) are listed in Tables 4 and 5.

From Table 4 we can see that, for the problem Laplace1(a), our method (16) with a large \( s \) outperforms the DY and SDC methods when the accuracy level is high. Moreover, with proper \((h, s)\), it performs better than the ABB_{min 2} method which dominates the DY and SDC methods. For the problem Laplace1(b), our method (16) is still competitive with the compared methods.

Table 5 shows similar tendency as the former one. However, for the problem Laplace1(b), our method (17) clearly outperforms the DY and SDC methods especially the requirement of the accuracy is high. Furthermore, our method (17) is very competitive with the ABB_{min 2} method in terms of total iterations.

5 Discussion

Based on the asymptotic optimal stepsize, we have proposed a new monotone gradient method which employs a new stepsize that converges to the reciprocal of the largest eigenvalue of the Hessian of the objective function. A nonmonotone variant has been presented as well. R-linear convergences of the proposed methods have been established. Our numerical experiments showed the proposed methods are very competitive with other recent successful gradient methods.
Gradient methods exploiting spectral properties

Table 4 Number of iterations of the methods (16), DY, ABB_{min 2} and SDC on the 3D Laplacian problem.

<table>
<thead>
<tr>
<th>α</th>
<th>ε</th>
<th>(λ, ε) for the method (16)</th>
<th>DY</th>
<th>ABB_{min 2}</th>
<th>SDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>60^3</td>
<td>10^{-6}</td>
<td>271</td>
<td>241</td>
<td>243</td>
<td>397</td>
</tr>
<tr>
<td>10^{-9}</td>
<td>348</td>
<td>257</td>
<td>421</td>
<td>357</td>
<td>334</td>
</tr>
<tr>
<td>10^{-12}</td>
<td>451</td>
<td>394</td>
<td>437</td>
<td>452</td>
<td>441</td>
</tr>
<tr>
<td>80^3</td>
<td>10^{-6}</td>
<td>315</td>
<td>441</td>
<td>362</td>
<td>303</td>
</tr>
<tr>
<td>10^{-9}</td>
<td>436</td>
<td>480</td>
<td>481</td>
<td>510</td>
<td>340</td>
</tr>
<tr>
<td>10^{-12}</td>
<td>602</td>
<td>548</td>
<td>601</td>
<td>631</td>
<td>451</td>
</tr>
<tr>
<td>100^3</td>
<td>10^{-6}</td>
<td>506</td>
<td>482</td>
<td>301</td>
<td>371</td>
</tr>
<tr>
<td>10^{-9}</td>
<td>691</td>
<td>639</td>
<td>541</td>
<td>527</td>
<td>565</td>
</tr>
<tr>
<td>10^{-12}</td>
<td>826</td>
<td>908</td>
<td>649</td>
<td>608</td>
<td>771</td>
</tr>
<tr>
<td>Total</td>
<td>1086</td>
<td>1164</td>
<td>956</td>
<td>971</td>
<td>1191</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>1475</td>
<td>1376</td>
<td>1443</td>
<td>1394</td>
<td>1348</td>
</tr>
<tr>
<td>10^{-9}</td>
<td>1879</td>
<td>1842</td>
<td>1687</td>
<td>1891</td>
<td>1663</td>
</tr>
</tbody>
</table>

As mentioned in Section 2, any stepsize approximates some $\alpha_k$ is expected to enforce a small $\alpha_k$. Thus, many stepsizes can be selected such as $\alpha_k^{BB1}$ and $\alpha_k^{BB2}$. We trivially combined the method (16) with the two BB stepizes, that is,

$$
\alpha_k = \begin{cases} 
\alpha_k^{BB1}, & \text{if mod}(k, h + s) < h; \\
\min\{\alpha_k^{BB1}, \alpha_k^{BB2}\}, & \text{otherwise},
\end{cases}
$$

and

$$
\alpha_k = \begin{cases} 
\alpha_k^{BB2}, & \text{if mod}(k, h + s) < h; \\
\min\{\alpha_k^{BB2}, \alpha_k^{BB1}\}, & \text{otherwise},
\end{cases}
$$

Promising numerical results of the methods (20) and (21) have been observed. Sophisticated numerical results exploit the favorable properties of the BB method and our new stepsize $\alpha_k$ will be expected.

An interesting question is how to approximate $\frac{1}{\lambda_i}$ to eliminate the corresponding components. This will be challenge because it needs a large stepsize that may increase those elements related to small eigenvalues and deteriorate the performance. Another future concern is to apply the proposed method to general unconstrained optimization and to constrained optimization.
Table 5 Number of iterations of the methods (17), DY, ABB_{min 2} and SDC on the 3D Laplacian problem.

<table>
<thead>
<tr>
<th>n</th>
<th>ε</th>
<th>(a, ε) for the method (17)</th>
<th>DY</th>
<th>ABB_{min 2}</th>
<th>SDC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(10, 20)</td>
<td>(10, 30)</td>
<td>(10, 50)</td>
<td>(10, 100)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60³</td>
<td>10⁻⁶</td>
<td>279</td>
<td>208</td>
<td>241</td>
<td>209</td>
</tr>
<tr>
<td>10⁻⁹</td>
<td>421</td>
<td>361</td>
<td>308</td>
<td>326</td>
<td>331</td>
</tr>
<tr>
<td>10⁻¹²</td>
<td>486</td>
<td>424</td>
<td>342</td>
<td>380</td>
<td>361</td>
</tr>
<tr>
<td>80³</td>
<td>10⁻⁶</td>
<td>301</td>
<td>335</td>
<td>271</td>
<td>290</td>
</tr>
<tr>
<td>10⁻⁹</td>
<td>489</td>
<td>490</td>
<td>421</td>
<td>376</td>
<td>450</td>
</tr>
<tr>
<td>10⁻¹²</td>
<td>676</td>
<td>552</td>
<td>596</td>
<td>551</td>
<td>524</td>
</tr>
<tr>
<td>100³</td>
<td>10⁻⁹</td>
<td>474</td>
<td>361</td>
<td>361</td>
<td>541</td>
</tr>
<tr>
<td>10⁻¹²</td>
<td>721</td>
<td>441</td>
<td>459</td>
<td>576</td>
<td>561</td>
</tr>
<tr>
<td>total</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>10⁻⁶</td>
<td>1631</td>
<td>1292</td>
<td>3168</td>
<td>1278</td>
<td>1341</td>
</tr>
<tr>
<td>10⁻⁹</td>
<td>1973</td>
<td>1866</td>
<td>1506</td>
<td>1716</td>
<td>1683</td>
</tr>
</tbody>
</table>

References