EQUIPPING BARZILAI-BORWEIN METHOD WITH TWO DIMENSIONAL QUADRATIC TERMINATION PROPERTY*

YAKUI HUANG†, YU-HONG DAI‡, AND XIN-WEI LIU§

Abstract. A novel gradient stepsize is derived at the motivation of equipping the Barzilai-Borwein (BB) method with two dimensional quadratic termination property. A remarkable feature of the novel stepsize is that its computation only depends on the BB stepsizes in previous iterations and does not require any exact line search or the Hessian, and hence it can easily be extended for nonlinear optimization. By adaptively taking long BB steps and some short steps associated with the new stepsize, we develop an efficient gradient method for quadratic optimization and general unconstrained optimization and extend it to solve extreme eigenvalues problems. The proposed method is further extended for box-constrained optimization and singly linearly box-constrained optimization by incorporating gradient projection techniques. Numerical experiments demonstrate that the proposed method outperforms the most successful gradient methods in the literature.

Key words. Barzilai-Borwein method, quadratic termination property, unconstrained optimization, extreme eigenvalues problem, box-constrained optimization, singly linearly box-constrained optimization

AMS subject classifications. 90C20, 90C25, 90C30

1. Introduction.

To minimize a smooth function $f(x) : \mathbb{R}^n \to \mathbb{R}$, the gradient method updates iterates as follows

$$x_{k+1} = x_k - \alpha_k g_k, \tag{1.1}$$

where $g_k = \nabla f(x_k)$ and α_k is the stepsize. Cauchy's (monotone) gradient method [7] (also known as the steepest descent (SD) method) chooses the stepsize, say α_k^{SD}, by the exact line search. Although the SD method converges Q-linearly, it performs poorly in many problems due to zigzag behaviors [2, 24]. By asking the stepsize to satisfy certain secant equations in the sense of least squares, Barzilai and Borwein [3] introduced the following long and short choices,

$$\alpha_k^{BB1} = \arg\min_{\alpha\in\mathbb{R}} \|\alpha^{-1}s_{k-1} - y_{k-1}\|_2 = \frac{s_{k-1}^{\mathsf{T}} s_{k-1}}{s_{k-1}^{\mathsf{T}} y_{k-1}} \tag{1.2}$$

and

$$\alpha_k^{BB2} = \arg\min_{\alpha\in\mathbb{R}} \|s_{k-1} - \alpha y_{k-1}\|_2 = \frac{s_{k-1}^{\mathsf{T}} y_{k-1}}{y_{k-1}^{\mathsf{T}} y_{k-1}}, \tag{1.3}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. Such genius stepsizes bring a surprisingly R-superlinear convergence in the two-dimensional strictly convex quadratic

*This work was supported by the National Natural Science Foundation of China (Grant Nos. 11701137, 11631013, 12071108, 11671116, 11991021, 12021001), the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No. XDA27000000), Natural Science Foundation of Hebei Province (Grant No. A2021202010), and Beijing Academy of Artificial Intelligence (BAAI).

†Institute of Mathematics, Hebei University of Technology, Tianjin 300401, China (hyk@hebut.edu.cn).

‡Corresponding author. LSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China; School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China (dyh@lsec.cc.ac.cn, http://lsec.cc.ac.cn/~dyh/).

§Institute of Mathematics, Hebei University of Technology, Tianjin 300401, China (mathlxw@hebut.edu.cn).

function [3]. For any dimensional strictly convex quadratic functions, the Barzilai-Borwein (BB) method is shown to be globally convergent [37] and the convergence rate is R-linear [16]. See Li and Sun [35] for an interesting improved R-linear convergence result of the BB method. By cooperating with the nonmonotone line search by Grippo et al. [28], Raydan [38] first extended the BB method for unconstrained optimization, yielding a very efficient gradient method called GBB. Birgin et al. [4] further extended the BB method to solve constrained optimization problems. Another significant work of the gradient method is due to Yuan [42, 43], who suggested to calculate the stepsize such that, with the previous and coming steps using SD stepsizes, the minimizer of a two dimensional strictly convex quadratic function is achieved in three iterations. By modifying Yuan's stepsize and alternating it with the SD stepsize in a suitable way, Dai and Yuan [17] proposed the so-called Dai-Yuan (monotone) gradient method, which performs even better than the (nonmonotone) BB method.

In this paper, we shall introduce a new mechanism for the gradient method to achieve two dimensional quadratic termination. Here we call a method has the property of two dimensional quadratic termination if it can give the exact minimizer of a two dimensional convex quadratic function within finite iterations. Interestingly, the aforementioned Yuan stepsize is a special example deduced from the mechanism. Furthermore, based on the mechanism, we derive a novel stepsize (2.15) such that the BB method equipped with the stepsize has the two dimensional quadratic termination property. A distinguished feature of the novel stepsize is that it is computed by BB stepsizes in two consecutive iterations and does not use any exact line search or the Hessian. Hence it can easily be extended to solve a wide class of unconstrained and constrained optimization problems.

By adaptively taking long BB steps and some short steps associated with the novel stepsize, we develop an efficient gradient method, the method (3.1), for quadratic optimization. Numerical experiments on minimizing quadratic functions show that the method (3.1) performs much better than many successful gradient methods developed recently including BB1 [3], Dai-Yuan (DY) [17], ABB [45], ABBmin1 [25], ABBmin2 [25] and SDC [20]. The combination of the method (3.1) and the Gripp-Lampariello-Lucidi (GLL) nonmonotone line search [28] yields an efficient gradient method, Algorithm 3.1, for unconstrained optimization. Numerical experiments on unconstrained problems from the CUTEst collection [27] show that Algorithm 3.1 outperforms GBB [38] and ABBmin [21, 25].

Furthermore, with suitable modifications of the BB stepsizes and the use of the Dai-Fletcher nonmonotone line search [12], the method (3.1) is extended for extreme eigenvalues problems, yielding Algorithm 4.1. Numerical experiments demonstrate the advantage of Algorithm 4.1 over EigUncABB [34]. By incorporating gradient projection techniques and taking constraints into consideration, the method (3.1) is further generalized for box-constrained optimization and singly linearly box-constrained (SLB) optimization, yielding Algorithm 5.1. Numerical experiments on box-constrained problems from the CUTEst collection [27] show that Algorithm 5.1 outperforms SPG [4, 5] and BoxVABBmin [9]. Meanwhile, numerical experiments with random SLB problems and SLB problems arising in support vector machine [8] highly suggest the potential benefit of Algorithm 5.1 comparing the Dai-Fletcher method [13] and the EQ-VABBmin method [10].

The paper is organized as follows. In Section 2, we introduce the new mechanism for the gradient method to achieve two dimensional quadratic termination. A novel stepsize (2.15) is derived such that the BB method equipped with the stepsize has such a property. Based on the novel stepsize, Section 3 presents an efficient gradi-

ent method, the method (3.1), for quadratic optimization and an efficient gradient algorithm, Algorithm 3.1, for unconstrained optimization. Furthermore, Section 4 provides an efficient gradient algorithm, Algorithm 4.1, for solving extreme eigenvalues problems and Section 5 provides an efficient gradient projection algorithm, Algorithm 5.1, for box-constrained optimization and singly linearly box-constrained optimization. Conclusion and discussion are made in Section 6.

2. A mechanism and a novel stepsize for the gradient method. In this section, we consider the unconstrained quadratic optimization

$$\min_{x\in\mathbb{R}^n} \; q(x) := \frac{1}{2}x^{\mathsf{T}}Ax - b^{\mathsf{T}}x, \tag{2.1}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $b \in \mathbb{R}^n$. At first, we shall provide a mechanism for the gradient method to achieve the two dimensional quadratic termination. Then we utilize the mechanism to derive a novel stepsize such that the BB method can have the two dimensional quadratic termination by equipping with such stepsize.

2.1. A mechanism for achieving two dimensional quadratic termination. Our mechanism starts from the following observation. Suppose that the gradient method is such that the gradient g_{k+1} is an eigenvector of the Hessian A; namely,

$$Ag_{k+1} = \lambda g_{k+1}, \tag{2.2}$$

where λ is some eigenvalue of A. In the unconstrained quadratic case, it is easy to see from (1.1) and (2.2) that

$$g_{k+2} = (I - \alpha_{k+1}A)g_{k+1} = (1 - \alpha_{k+1}\lambda)g_{k+1}, \tag{2.3}$$

which means that g_{k+2} is parallel to g_{k+1} and $Ag_{k+2} = \lambda g_{k+2}$. Notice that for many stepsize formulae in the gradient method, the stepsize α_{k+2} is the reciprocal of the Rayleigh quotient of the Hessian A with respect to some vector in the form of $A^\mu g_{k+1}$ or $A^\mu g_{k+2}$, where μ is some real number; namely,

$$\alpha_{k+2} = \frac{(A^\mu g_{k+i})^{\mathsf{T}}(A^\mu g_{k+i})}{(A^\mu g_{k+i})^{\mathsf{T}}A(A^\mu g_{k+i})}, \quad \text{where } i = 1 \text{ or } 2. \tag{2.4}$$

Thus if neither g_{k+1} nor g_{k+2} vanish, we will have that $\alpha_{k+2} = 1/\lambda$ and hence $g_{k+3} = (1 - \alpha_{k+2}\lambda)g_{k+2} = 0$. Therefore we must have that $g_{k+i} = 0$ for some $1 \le i \le 3$, yielding the finite termination.

As the relation (2.2) is difficult to reach in the general case, we consider some other conditions in the following. To this aim, let λ_1 and λ_n be the smallest and largest eigenvalues of A, respectively, and assume that ψ is a real analytic function on $[\lambda_1, \lambda_n]$ and can be expressed by Laurent series $\psi(z) = \sum_{k=-\infty}^{\infty} c_k z^k$ where $c_k \in \mathbb{R}$ is such that $0 < \sum_{k=-\infty}^{\infty} c_k z^k < +\infty$ for all $z \in [\lambda_1, \lambda_n]$. Then the relation (2.2) implies that

$$g_{\nu(k)}^{\mathsf{T}}\psi(A)g_{k+1} = \psi(\lambda)g_{\nu(k)}^{\mathsf{T}}g_{k+1}, \tag{2.5}$$

where $\nu(k) \in \{1, \ldots, k\}$. Furthermore, assume that ψ_1, ψ_2, ψ_3, ψ_4 are real analytic functions on $[\lambda_1, \lambda_n]$ with the same property of ψ, satisfying $\psi_1(z)\psi_2(z) = \psi_3(z)\psi_4(z)$ for $z \in \mathbb{R}$. It follows from (2.5) that

$$g_{\nu_1(k)}^{\mathsf{T}}\psi_1(A)g_{k+1} \cdot g_{\nu_2(k)}^{\mathsf{T}}\psi_2(A)g_{k+1} = g_{\nu_1(k)}^{\mathsf{T}}\psi_3(A)g_{k+1} \cdot g_{\nu_2(k)}^{\mathsf{T}}\psi_4(A)g_{k+1}, \tag{2.6}$$

where $\nu_1(k), \nu_2(k) \in \{1, \ldots, k\}$. Again, notice that in the unconstrained quadratic case, the gradient method (1.1) gives $g_{k+1} = (I - \alpha_k A)g_k$. The relation (2.6) provides a quadratic equation in stepsize α_k as follows.

$$\begin{aligned} & g_{\nu_1(k)}^{\mathsf{T}} \psi_1(A)(I - \alpha_k A)g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_2(A)(I - \alpha_k A)g_k \\ & = g_{\nu_1(k)}^{\mathsf{T}} \psi_3(A)(I - \alpha_k A)g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_4(A)(I - \alpha_k A)g_k. \end{aligned} \tag{2.7}$$

In the following, we shall show that if $n = 2$, and if the gradient method preserves the invariance property under orthogonal transformations, we can derive the relation (2.2) from the relation (2.7) and hence realizes a mechanism for achieving the two dimensional quadratic termination.

To proceed, we rewrite the quadratic equation (2.7) as follows,

$$\phi_1 \alpha_k^2 - \phi_2 \alpha_k + \phi_3 = 0, \tag{2.8}$$

where the coefficients are

$$\begin{aligned} \phi_1 &= g_{\nu_1(k)}^{\mathsf{T}} \psi_1(A) A g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_2(A) A g_k - g_{\nu_1(k)}^{\mathsf{T}} \psi_3(A) A g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_4(A) A g_k, \\ \phi_2 &= g_{\nu_1(k)}^{\mathsf{T}} \psi_1(A) g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_2(A) A g_k + g_{\nu_1(k)}^{\mathsf{T}} \psi_1(A) A g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_2(A) g_k \\ &\quad - g_{\nu_1(k)}^{\mathsf{T}} \psi_3(A) g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_4(A) A g_k - g_{\nu_1(k)}^{\mathsf{T}} \psi_3(A) A g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_4(A) g_k, \\ \phi_3 &= g_{\nu_1(k)}^{\mathsf{T}} \psi_1(A) g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_2(A) g_k - g_{\nu_1(k)}^{\mathsf{T}} \psi_3(A) g_k \cdot g_{\nu_2(k)}^{\mathsf{T}} \psi_4(A) g_k. \end{aligned}$$

Then the two solutions of (2.7) or equivalently (2.8) are

$$\alpha_k = \frac{\phi_2 \pm \sqrt{\phi_2^2 - 4\phi_1\phi_3}}{2\phi_1} = \frac{2}{\frac{\phi_2}{\phi_3} \mp \sqrt{\left(\frac{\phi_2}{\phi_3}\right)^2 - 4\frac{\phi_1}{\phi_3}}}. \tag{2.9}$$

Now we are ready to give the following basic theorem for the two dimensional quadratic termination property.

THEOREM 2.1. *Consider the gradient method (1.1) with the invariance property under orthogonal transformations for the problem (2.1) with $n = 2$. Then if the stepsize solves the quadratic equation (2.7) with $\psi_1(z)\psi_2(z) = \psi_3(z)\psi_4(z)$ and if α_{k+2} is of the form (2.4), we must have that $g_{k+i} = 0$ for some $1 \le i \le 3$.*

Proof. Due to the invariance property under orthogonal transformations and since $n = 2$, we can assume without loss of generality that $A = \mathrm{diag}\{1, \lambda\}$ with $\lambda > 0$.

Denote $g_k^{(i)}$, $g_{\nu_1(k)}^{(i)}$ and $g_{\nu_2(k)}^{(i)}$ to be the i-th components of g_k, $g_{\nu_1(k)}$ and $g_{\nu_2(k)}$, respectively, for $i = 1, 2$. Notice that $\psi_1(z)\psi_2(z) = \psi_3(z)\psi_4(z)$. Direct calculations show that the coefficients in (2.8) are equal to

$$\phi_1 = \lambda\Delta, \quad \phi_2 = (1 + \lambda)\Delta, \quad \phi_3 = \Delta,$$

where

$$\begin{aligned} \Delta =& g_{\nu_1(k)}^{(1)} g_{\nu_2(k)}^{(2)} g_k^{(1)} g_k^{(2)} (\psi_1(1)\psi_2(\lambda) - \psi_3(1)\psi_4(\lambda)) \\ & + g_{\nu_1(k)}^{(2)} g_{\nu_2(k)}^{(1)} g_k^{(1)} g_k^{(2)} (\psi_1(\lambda)\psi_2(1) - \psi_3(\lambda)\psi_4(1)). \end{aligned}$$

Thus, by (2.9), the two roots of (2.7) are

$$\alpha_{k,1} = 1 \quad \text{and} \quad \alpha_{k,2} = \frac{1}{\lambda}.$$

If $\alpha_{k,1}$ is applied, by the update rule (1.1), we get that $g_{k+1}^{(1)} = (1-\alpha_{k,1})g_k^{(1)} = 0$ and hence $Ag_{k+1} = \lambda g_{k+1}$. Thus if neither g_{k+1} nor g_{k+2} vanish, and if α_{k+2} is of the form (2.4), we will have that $\alpha_{k+2} = 1/\lambda$ and hence $g_{k+3} = (1-\alpha_{k+2}\lambda)g_{k+2} = 0$. Similar conclusion can be drawn for $\alpha_{k,2}$. So we must have that $g_{k+i} = 0$ for some $1 \le i \le 3$. This completes the proof. ∎

Remark 2.2. By Theorem 2.1, to achieve the two dimensional quadratic termination property for the stepsize α_k calculated from (2.7), the choices of $\nu_1(k), \nu_2(k)$, ψ_1, ψ_2, ψ_3, and ψ_4 can be arbitrary, provided that $\psi_1(z)\psi_2(z) = \psi_3(z)\psi_4(z)$ for $z \in \mathbb{R}$. Consequently, in the two dimensional case, such a stepsize always leads both the SD and BB methods terminate in finite iterations, since the stepsizes in the methods are clearly of the form (2.4).

The Dai-Yuan gradient method [17] employs the following stepsize

$$\alpha_k^{DY} = \frac{2}{\frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}} + \sqrt{\left(\frac{1}{\alpha_{k-1}^{SD}} - \frac{1}{\alpha_k^{SD}}\right)^2 + \frac{4\|g_k\|_2^2}{(\alpha_{k-1}^{SD}\|g_{k-1}\|_2)^2}}}, \tag{2.10}$$

which is such that the SD method enjoys the two dimensional quadratic termination. Here we notice that the stepsize (2.10) is a variant of the stepsize by Yuan [42]. Now we show that α_k^{DY} is a special solution of (2.7).

THEOREM 2.3. *Suppose that $\alpha_{k-1} = \alpha_{k-1}^{SD}$ and $I - \alpha_{k-1}A$ is invertible. Then the stepsize α_k^{DY} is a solution of the equation (2.7) corresponding to $\nu_1(k) = k-1$, $\nu_2(k) = k$, $\psi_1(A) = \psi_4(A) = (I-\alpha_{k-1}A)^{-1}$ and $\psi_2(A) = \psi_3(A) = I$.*

Proof. From (2.7) and the choices of $\nu_1(k)$, $\nu_2(k)$, ψ_1, ψ_2, ψ_3, and ψ_4, we have

$$g_{k-1}^{\mathsf{T}}(I-\alpha_k A)g_{k-1} \cdot g_k^{\mathsf{T}}(g_k - \alpha_k A g_k) = g_{k-1}^{\mathsf{T}}(g_k - \alpha_k A g_k) \cdot g_k^{\mathsf{T}}(I-\alpha_k A)g_{k-1}.$$

Denote $\zeta_{k,j} = g_k^{\mathsf{T}} A^j g_k$ for $j = 0, 1$. It follows that

$$(\zeta_{k-1,0} - \alpha_k\zeta_{k-1,1})(\zeta_{k,0} - \alpha_k\zeta_{k,1}) = (g_{k-1}^{\mathsf{T}}g_k - g_{k-1}^{\mathsf{T}}Ag_k)^2. \tag{2.11}$$

Since $\alpha_{k-1} = \alpha_{k-1}^{SD}$ and $g_k = (I-\alpha_{k-1}A)g_{k-1}$, we obtain $g_{k-1}^{\mathsf{T}}g_k = 0$ and

$$g_{k-1}^{\mathsf{T}}Ag_k = \frac{1}{\alpha_{k-1}^{SD}}(g_{k-1}-g_k)^{\mathsf{T}}g_k = -\frac{1}{\alpha_{k-1}^{SD}}\zeta_{k,0},$$

which together with (2.11) gives the equation (2.8) with

$$\phi_1 = \zeta_{k-1,1}\zeta_{k,1} - \frac{1}{(\alpha_{k-1}^{SD})^2}\zeta_{k,0}^2,\ \phi_2 = \zeta_{k,0}\zeta_{k-1,1} + \zeta_{k-1,0}\zeta_{k,1},\ \phi_3 = \zeta_{k-1,0}\zeta_{k,0}.$$

Consequently, we have that

$$\frac{\phi_1}{\phi_3} = \frac{\zeta_{k-1,1}\zeta_{k,1} - \frac{1}{(\alpha_{k-1}^{SD})^2}\zeta_{k,0}^2}{\zeta_{k-1,0}\zeta_{k,0}} = \frac{1}{\alpha_{k-1}^{SD}\alpha_k^{SD}} - \frac{\|g_k\|_2^2}{\left(\alpha_{k-1}^{SD}\|g_{k-1}\|_2\right)^2} \tag{2.12}$$

and

$$\frac{\phi_2}{\phi_3} = \frac{\zeta_{k,0}\zeta_{k-1,1} + \zeta_{k-1,0}\zeta_{k,1}}{\zeta_{k-1,0}\zeta_{k,0}} = \frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}}, \tag{2.13}$$

where the last equality in (2.12) is due to the definition of $\zeta_{k,0}$. Thus we conclude from (2.9) that the stepsize α_k^{DY} is the smaller root of (2.7). This completes the proof. ∎

Remark 2.4. The stepsize α_k^{DY} can only lead the SD method achieves the two dimensional quadratic termination because it uses the relation $g_{k-1}^{\mathsf{T}} g_k = 0$ to eliminate some terms of the equation (2.7).

Remark 2.5. For the family of gradient methods with

$$\alpha_k = \frac{g_{k-1}^{\mathsf{T}} \psi(A) g_{k-1}}{g_{k-1}^{\mathsf{T}} \psi(A) A g_{k-1}},$$

Huang et al. [32] presented a stepsize, called $\tilde{\alpha}_k^H$, such that the family enjoys the two dimensional quadratic termination. In particular,

$$\tilde{\alpha}_k^H = \frac{2}{H_k^{(11)} + H_k^{(22)} + \sqrt{(H_k^{(11)} - H_k^{(22)})^2 + 4(H_k^{(12)})^2}},$$

where, for $r \in \mathbb{R}$, $H_k^{(ij)}$ is the (i,j) element of

$$H_k = \begin{pmatrix} \frac{q_{k-1}^{\mathsf{T}} \psi^{2r}(A) A q_{k-1}}{\|\psi^r(A) q_{k-1}\|^2} & \frac{q_{k-1}^{\mathsf{T}} \psi(A) A g_k}{\|\psi^r(A) q_{k-1}\| \|\psi^{1-r}(A) g_k\|} \\ \frac{q_{k-1}^{\mathsf{T}} \psi(A) A g_k}{\|\psi^r(A) q_{k-1}\| \|\psi^{1-r}(A) g_k\|} & \frac{g_k^{\mathsf{T}} \psi^{2(1-r)}(A) A g_k}{\|\psi^{1-r}(A) g_k\|^2} \end{pmatrix}$$

with q_{k-1} being a vector satisfying $(I - \alpha_{k-2}A) q_{k-1} = g_{k-2}$. Using the same way, under the assumption that $I - \alpha_{k-1}A$ and $I - \alpha_{k-2}A$ are invertible, we can deduce from Theorem 2.3 that for the same $r \in \mathbb{R}$, the stepsize $\tilde{\alpha}_k^H$ is a solution of the equation (2.7) corresponding to $\nu_1(k) = k-2$, $\nu_2(k) = k$, $\psi_1(A) = (I - \alpha_{k-2}A)^{-3}(I - \alpha_{k-1}A)^{-1}\psi^{2r}(A)$, $\psi_2(A) = \psi^{2(1-r)}(A)$, $\psi_3(A) = (I - \alpha_{k-2}A)^{-1}\psi(A)$, and $\psi_4(A) = (I - \alpha_{k-2}A)^{-2}(I - \alpha_{k-1}A)^{-1}\psi(A)$. The drawback of this stepsize $\tilde{\alpha}_k^H$ is that the Hessian is involved in its calculation.

As shown in Theorem 2.1, for a two dimensional strictly convex quadratic function, the two roots of (2.7) are reciprocals of the largest and smallest eigenvalues of the Hessian A. This is also true, in the sense of limitation for the stepsize α_k^{DY} and the corresponding larger root when $n > 2$, see [30]. The reason why the smaller stepsize is preferable is that the larger one is generally not a good approximation of the reciprocal of the largest eigenvalue and may significantly increase the objective value, see [30, 31] for details.

2.2. A novel stepsize equipped for the BB method. Now we shall equip the BB method with a new stepsize via the quadratic equation (2.7) to achieve the two dimensional quadratic termination.

To this aim, suppose that $I - \alpha_{k-2}A$ and $I - \alpha_{k-1}A$ are invertible and let $\nu_1(k) = k-2$ and $\nu_2(k) = k-1$. Furthermore, consider the following ψ_i $(i = 1, \ldots, 4)$,

$$\begin{aligned} &\psi_1(A) = (I - \alpha_{k-2}A)^{-1}, \\ &\psi_2(A) = (I - \alpha_{k-1}A)^{-1}, \\ &\psi_3(A) = (I - \alpha_{k-2}A)^{-1}(I - \alpha_{k-1}A)^{-1}, \\ &\psi_4(A) = I, \end{aligned}$$

which satisfy $\psi_1(z)\psi_2(z) = \psi_3(z)\psi_4(z)$ for $z \in \mathbb{R}$. In this case, the equation (2.7) becomes

$$g_{k-2}^{\mathsf{T}}(I - \alpha_{k-2}A)^{-1}(I - \alpha_k A) g_k \cdot g_{k-1}^{\mathsf{T}}(I - \alpha_{k-1}A)^{-1}(I - \alpha_k A) g_k$$

$$= g_{k-2}^{\mathsf{T}}(I-\alpha_{k-2}A)^{-1}(I-\alpha_{k-1}A)^{-1}(I-\alpha_k A)g_k \cdot g_{k-1}^{\mathsf{T}}(I-\alpha_k A)g_k. \tag{2.14}$$

In order to get the formula of the novel stepsize, we assume for the moment that the equation (2.14) has a solution. This issue will be specified by Theorem 2.9.

THEOREM 2.6. *Suppose that $\alpha_{k-1}^{BB1} \neq \alpha_k^{BB1}$. Then the following stepsize α_k^{new} is a solution of the quadratic equation (2.14),*

$$\alpha_k^{new} = \frac{2}{\frac{\phi_2}{\phi_3} + \sqrt{\left(\frac{\phi_2}{\phi_3}\right)^2 - 4\,\frac{\phi_1}{\phi_3}}}, \tag{2.15}$$

where

$$\frac{\phi_1}{\phi_3} = \frac{\alpha_{k-1}^{BB2} - \alpha_k^{BB2}}{\alpha_{k-1}^{BB2}\alpha_k^{BB2}(\alpha_{k-1}^{BB1} - \alpha_k^{BB1})} \quad \textit{and} \quad \frac{\phi_2}{\phi_3} = \frac{\alpha_{k-1}^{BB1}\alpha_{k-1}^{BB2} - \alpha_k^{BB1}\alpha_k^{BB2}}{\alpha_{k-1}^{BB2}\alpha_k^{BB2}(\alpha_{k-1}^{BB1} - \alpha_k^{BB1})}. \tag{2.16}$$

Proof. Denote $\zeta_{k,j} = g_k^{\mathsf{T}} A^j g_k$ for $j = 0, 1, 2$ as before. It follows from the definitions of α_k^{BB1} and α_k^{BB2} that $\alpha_k^{BB1} = \zeta_{k-1,0}/\zeta_{k-1,1}$ and $\alpha_k^{BB2} = \zeta_{k-1,1}/\zeta_{k-1,2}$. By the update rule (1.1), we know that $g_{k+1} = (I - \alpha_k A)g_k$ holds for all $k \geq 1$ in the context of unconstrained quadratic optimization. Combining this with the fact that $I - \alpha_{k-i}A$ is invertible for $i = 1, 2$ gives

$$\begin{aligned}
&g_{k-2}^{\mathsf{T}}(I-\alpha_{k-2}A)^{-1}(I-\alpha_k A)g_k \\
&\quad = g_{k-2}^{\mathsf{T}}(I-\alpha_k A)(I-\alpha_{k-1}A)g_{k-2} \\
&\quad = \zeta_{k-2,0} - \alpha_{k-1}\zeta_{k-2,1} - \alpha_k(\zeta_{k-2,1} - \alpha_{k-1}\zeta_{k-2,2}) \\
&\quad = \zeta_{k-2,1}(\alpha_{k-1}^{BB1} - \alpha_{k-1}) - \alpha_k\zeta_{k-2,2}(\alpha_{k-1}^{BB2} - \alpha_{k-1}),
\end{aligned}$$

$$\begin{aligned}
&g_{k-1}^{\mathsf{T}}(I-\alpha_{k-1}A)^{-1}(I-\alpha_k A)g_k \\
&\quad = g_{k-1}^{\mathsf{T}}(I-\alpha_k A)g_{k-1} \\
&\quad = \zeta_{k-1,1}(\alpha_k^{BB1} - \alpha_k),
\end{aligned}$$

$$\begin{aligned}
&g_{k-2}^{\mathsf{T}}(I-\alpha_{k-2}A)^{-1}(I-\alpha_{k-1}A)^{-1}(I-\alpha_k A)g_k \\
&\quad = g_{k-2}^{\mathsf{T}}(I-\alpha_k A)g_{k-2} \\
&\quad = \zeta_{k-2,1}(\alpha_{k-1}^{BB1} - \alpha_k),
\end{aligned}$$

$$\begin{aligned}
&g_{k-1}^{\mathsf{T}}(I-\alpha_k A)g_k \\
&\quad = g_{k-1}^{\mathsf{T}}(I-\alpha_k A)(I-\alpha_{k-1}A)g_{k-1} \\
&\quad = \zeta_{k-1,1}(\alpha_k^{BB1} - \alpha_{k-1}) - \alpha_k\zeta_{k-1,2}(\alpha_k^{BB2} - \alpha_{k-1}).
\end{aligned}$$

The equation (2.14) can be written in the form (2.8) with

$$\begin{aligned}
\phi_1 &= \zeta_{k-2,2}\zeta_{k-1,1}(\alpha_{k-1}^{BB2} - \alpha_{k-1}) - \zeta_{k-2,1}\zeta_{k-1,2}(\alpha_k^{BB2} - \alpha_{k-1}) \\
&= \zeta_{k-2,2}\zeta_{k-1,2}\left[\alpha_k^{BB2}(\alpha_{k-1}^{BB2} - \alpha_{k-1}) - \alpha_{k-1}^{BB2}(\alpha_k^{BB2} - \alpha_{k-1})\right] \\
&= \zeta_{k-2,2}\zeta_{k-1,2}\alpha_{k-1}(\alpha_{k-1}^{BB2} - \alpha_k^{BB2}),
\end{aligned}$$

$$\begin{aligned}
\phi_2 &= \zeta_{k-2,2}\zeta_{k-1,1}\alpha_k^{BB1}(\alpha_{k-1}^{BB2}-\alpha_{k-1}) + \zeta_{k-2,1}\zeta_{k-1,1}(\alpha_{k-1}^{BB1}-\alpha_{k-1}) \\
&\quad -\zeta_{k-2,1}\zeta_{k-1,2}\alpha_{k-1}^{BB1}(\alpha_k^{BB2}-\alpha_{k-1}) - \zeta_{k-2,1}\zeta_{k-1,1}(\alpha_k^{BB1}-\alpha_{k-1}) \\
&= \zeta_{k-2,2}\zeta_{k-1,2}\big[\alpha_k^{BB1}\alpha_k^{BB2}(\alpha_{k-1}^{BB2}-\alpha_{k-1}) - \alpha_{k-1}^{BB1}\alpha_{k-1}^{BB2}(\alpha_k^{BB2}-\alpha_{k-1}) \\
&\quad +\alpha_{k-1}^{BB2}\alpha_k^{BB2}(\alpha_{k-1}^{BB1}-\alpha_k^{BB1})\big] \\
&= \zeta_{k-2,2}\zeta_{k-1,2}\alpha_{k-1}(\alpha_{k-1}^{BB1}\alpha_{k-1}^{BB2}-\alpha_k^{BB1}\alpha_k^{BB2}), \\
\phi_3 &= \zeta_{k-2,1}\zeta_{k-1,1}\alpha_{k-1}(\alpha_{k-1}^{BB1}-\alpha_k^{BB1}).
\end{aligned}$$

Therefore we obtain

$$\begin{aligned}
\frac{\phi_1}{\phi_3} &= \frac{\zeta_{k-2,2}\zeta_{k-1,2}\alpha_{k-1}(\alpha_{k-1}^{BB2}-\alpha_k^{BB2})}{\zeta_{k-2,1}\zeta_{k-1,1}\alpha_{k-1}(\alpha_{k-1}^{BB1}-\alpha_k^{BB1})} \\
&= \frac{\alpha_{k-1}^{BB2}-\alpha_k^{BB2}}{\alpha_{k-1}^{BB2}\alpha_k^{BB2}(\alpha_{k-1}^{BB1}-\alpha_k^{BB1})}, \\
\frac{\phi_2}{\phi_3} &= \frac{\zeta_{k-2,2}\zeta_{k-1,2}\alpha_{k-1}(\alpha_{k-1}^{BB1}\alpha_{k-1}^{BB2}-\alpha_k^{BB1}\alpha_k^{BB2})}{\zeta_{k-2,1}\zeta_{k-1,1}\alpha_{k-1}(\alpha_{k-1}^{BB1}-\alpha_k^{BB1})} \\
&= \frac{\alpha_{k-1}^{BB1}\alpha_{k-1}^{BB2}-\alpha_k^{BB1}\alpha_k^{BB2}}{\alpha_{k-1}^{BB2}\alpha_k^{BB2}(\alpha_{k-1}^{BB1}-\alpha_k^{BB1})}.
\end{aligned}$$

This completes the proof by noting that α_k^{new} is the smaller solution of (2.8). ∎

Remark 2.7. Although the derivation of α_k^{new} is based on unconstrained quadratic optimization, it can be applied for general nonlinear optimization as well because the formula is only related to the BB stepsizes computed in the last two iterations.

Remark 2.8. Interestingly enough, by the proof of Theorem 2.6, the stepsize α_k^{new} is independent of the gradient method. In other words, it satisfies the equation (2.14) for any gradient method such as the SD, BB and alternate BB [12] methods.

Since the equation (2.14) is a special case of (2.7), we know by Theorems 2.1 and 2.6 that the stepsize α_k^{new} yields both the BB1 and BB2 methods achieve the desired two dimensional quadratic termination. For the purpose of numerical verification, we applied the two methods with α_3 replaced with α_3^{new} to the unconstrained quadratic minimization problem (2.1) with

$$A = \mathrm{diag}\{1,\, \lambda\} \quad \text{and} \quad b = 0. \tag{2.17}$$

In particular, we used $\alpha_1 = \alpha_1^{SD}$ and $\alpha_3 = \alpha_3^{new}$ for both methods, and $\alpha_k = \alpha_k^{BB1}$ and $\alpha_k = \alpha_k^{BB2}$ for the BB1 and BB2 methods, respectively, when $k \neq 3$. The algorithm was run for five iterations using ten random starting points. Table 1 presents averaged values of $\|g_6\|_2$ and $f(x_6)$. It can be observed that for different λ, the values of $\|g_6\|_2$ and $f(x_6)$ obtained by the BB1 and BB2 methods with α_3^{new} are numerically very close to zero, whereas those values obtained by the unmodified BB1 method are far away from zero.

The next theorem indicates that the quadratic equation (2.14) always has a solution. It also provides upper and lower bounds for α_k^{new}. A simple way of computing ϕ_2/ϕ_3 is given in the proof of the theorem. Here we do not consider the trivial case $\phi_3 = 0$.

TABLE 1
Averaged results on problem (2.17) with different condition numbers.

λ	BB1		BB1 with α_3^{new}		BB2 with α_3^{new}	
	$\|g_6\|_2$	$f(x_6)$	$\|g_6\|_2$	$f(x_6)$	$\|g_6\|_2$	$f(x_6)$
10	6.9873e-01	7.5641e-02	9.3863e-18	1.7612e-32	7.5042e-20	5.1740e-33
100	6.3834e+00	1.8293e+00	1.3555e-17	1.1388e-31	8.6044e-17	3.2604e-31
1000	1.5874e+00	6.6377e-03	8.8296e-16	3.2198e-30	3.7438e-28	9.5183e-31
10000	2.9710e+01	2.1038e-01	8.3267e-17	3.2828e-30	2.0988e-31	8.0889e-30

THEOREM 2.9. *The stepsize α_k^{new} in Theorem 2.6 is well defined. Moreover, if $\phi_1/\phi_3 \geq 0$ and $\phi_2 \neq 0$, we have that*

$$\phi_3/\phi_2 \leq \alpha_k^{new} \leq \min\{\alpha_k^{BB2}, \alpha_{k-1}^{BB2}\}; \tag{2.18}$$

if $\phi_1/\phi_3 < 0$ and $\phi_2 \neq 0$, we have that

$$\max\{\alpha_k^{BB2}, \alpha_{k-1}^{BB2}\} \leq \alpha_k^{new} \leq |\phi_3/\phi_2|\,. \tag{2.19}$$

Proof. If $\phi_1/\phi_3 < 0$, we see from (2.15) that α_k^{new} is well defined. Now we consider the case that $\phi_1/\phi_3 \geq 0$. Notice that ϕ_2/ϕ_3 can be rewritten as

$$\begin{aligned}\frac{\phi_2}{\phi_3} &= \frac{\alpha_{k-1}^{BB2}(\alpha_{k-1}^{BB1} - \alpha_k^{BB1}) + \alpha_k^{BB1}(\alpha_{k-1}^{BB2} - \alpha_k^{BB2})}{\alpha_{k-1}^{BB2}\alpha_k^{BB2}(\alpha_{k-1}^{BB1} - \alpha_k^{BB1})} \\ &= \frac{1}{\alpha_k^{BB2}} + \frac{\phi_1}{\phi_3}\alpha_k^{BB1}. \end{aligned} \tag{2.20}$$

Since $\alpha_k^{BB1} \geq \alpha_k^{BB2}$, by (2.20) and (2.16), we obtain

$$\begin{aligned}\left(\frac{\phi_2}{\phi_3}\right)^2 &\geq \left(\frac{\phi_2}{\phi_3}\right)^2 - 4\,\frac{\phi_1}{\phi_3} \\ &= \frac{1}{(\alpha_k^{BB2})^2} + \left(\frac{\phi_1}{\phi_3}\right)^2 (\alpha_k^{BB1})^2 + 2\frac{\phi_1}{\phi_3}\frac{\alpha_k^{BB1}}{\alpha_k^{BB2}} - 4\,\frac{\phi_1}{\phi_3} \\ &\geq \frac{1}{(\alpha_k^{BB2})^2} + \left(\frac{\phi_1}{\phi_3}\right)^2 (\alpha_k^{BB1})^2 - 2\,\frac{\phi_1}{\phi_3}\frac{\alpha_k^{BB1}}{\alpha_k^{BB2}} \\ &= \left(\frac{1}{\alpha_k^{BB2}} - \frac{\phi_1}{\phi_3}\alpha_k^{BB1}\right)^2, \end{aligned}$$

which with (2.15) indicates that α_k^{new} is well defined and

$$\frac{\phi_3}{\phi_2} \leq \alpha_k^{new} \leq \min\left\{\alpha_k^{BB2},\ \frac{1}{\alpha_k^{BB1}}\frac{\phi_3}{\phi_1}\right\} \leq \alpha_k^{BB2}. \tag{2.21}$$

Similarly to (2.20), we can get that

$$\frac{\phi_2}{\phi_3} = \frac{1}{\alpha_{k-1}^{BB2}} + \frac{\phi_1}{\phi_3}\alpha_{k-1}^{BB1}. \tag{2.22}$$

Hence, we obtain

$$\frac{\phi_3}{\phi_2} \leq \alpha_k^{new} \leq \min\left\{\alpha_{k-1}^{BB2},\ \frac{1}{\alpha_{k-1}^{BB1}}\frac{\phi_3}{\phi_1}\right\} \leq \alpha_{k-1}^{BB2}. \tag{2.23}$$

Combining (2.21) and (2.23) yields the desired inequalities (2.18). This completes the proof of (2.18).

To prove (2.19), by $\phi_1/\phi_3 < 0$ and (2.20), we have that

$$\left(\frac{\phi_2}{\phi_3}\right)^2 \le \left(\frac{\phi_2}{\phi_3}\right)^2 - 4\,\frac{\phi_1}{\phi_3} \le \left(\frac{1}{\alpha_k^{BB2}} - \frac{\phi_1}{\phi_3}\alpha_k^{BB1}\right)^2,$$

which with the fact $\alpha_k^{BB1} \ge 0$ implies that $\alpha_k^{BB2} \le \alpha_k^{new} \le |\phi_3/\phi_2|$. Similarly, by (2.22), we can obtain $\alpha_{k-1}^{BB2} \le \alpha_k^{new} \le |\phi_3/\phi_2|$. Then the relation (2.19) follows immediately. ∎

As will be seen in the next section, the above theorem stimulates us to provide a new choice for the short stepsize in the algorithmic design of the gradient method. Finally, we give an asymptotic spectral property of α_k^{new} within the SD method.

THEOREM 2.10. *When applying the SD method to n-dimensional unconstrained quadratic problem* (2.1)*, we have that* $\lim_{k\to\infty} \alpha_k^{new} = 1/\lambda_n$.

Proof. By Theorem 1 of [30], when applying the SD method to problem (2.1), we obtain

$$\begin{aligned}
\lim_{k\to\infty} \alpha_{2k}^{BB1} &= \lim_{k\to\infty} \alpha_{2k-1}^{SD} = \frac{1+c^2}{\lambda_1(\kappa+c^2)},\\
\lim_{k\to\infty} \alpha_{2k+1}^{BB1} &= \lim_{k\to\infty} \alpha_{2k}^{SD} = \frac{1+c^2}{\lambda_1(1+c^2\kappa)},\\
\lim_{k\to\infty} \alpha_{2k}^{BB2} &= \frac{\kappa+c^2}{\lambda_1(\kappa^2+c^2)},\\
\lim_{k\to\infty} \alpha_{2k+1}^{BB2} &= \frac{1+c^2\kappa}{\lambda_1(1+c^2\kappa^2)},
\end{aligned}$$

where $\kappa = \lambda_n/\lambda_1$ is the condition number of A and c is a constant. It follows from (2.15) and (2.16) that

$$\begin{aligned}
\lim_{k\to\infty} \frac{\phi_1}{\phi_3} &= \lambda_1\lambda_n,\\
\lim_{k\to\infty} \frac{\phi_2}{\phi_3} &= \lambda_1+\lambda_n,
\end{aligned}$$

which yield $\lim_{k\to\infty} \alpha_k^{new} = 1/\lambda_n$. This completes our proof. ∎

3. A new gradient method for unconstrained optimization. In this section, by making use of the new stepsize α_k^{new}, we develop an efficient gradient method for solving unconstrained optimization problems.

3.1. Quadratic optimization. To begin with, we consider the quadratic optimization (2.1) again, which is often used for constructing and analyzing optimization methods and plays an important role in nonlinear optimization.

Extensive studies show that using the long BB stepsize α_k^{BB1} (since $\alpha_k^{BB1} \ge \alpha_k^{BB2}$) and some short stepsize in an alternate or adaptive manner is numerically better than the original BB method, see for example [10, 11, 12, 14, 15, 25, 26, 32, 39, 45]. If $\phi_1/\phi_3 \ge 0$ and $\phi_2 \ne 0$, we know from Theorem 2.9 that the new stepsize α_k^{new} is shorter than the two short stepsizes α_k^{BB2} and α_{k-1}^{BB2}. On the other hand, if $\phi_1/\phi_3 < 0$ and $\phi_2 \ne 0$, the stepsize α_k^{new} is longer than both α_k^{BB2} and α_{k-1}^{BB2}. Combining the two cases, we shall replace α_k^{new} by $\min\{\alpha_{k-1}^{BB2}, \alpha_k^{BB2}, \alpha_k^{new}\}$, which is

the shortest stepsize among α_{k-1}^{BB2}, α_k^{BB2} and α_k^{new}, in the algorithmic design of new gradient methods.

Motivated by the success of adaptive schemes [6, 45], we now suggest the gradient method (1.1) with the stepsizes $\{\alpha_k : k \geq 3\}$ given by

$$\alpha_k = \begin{cases} \min\{\alpha_{k-1}^{BB2}, \alpha_k^{BB2}, \alpha_k^{new}\}, & \text{if } \alpha_k^{BB2}/\alpha_k^{BB1} < \tau_k; \\ \alpha_k^{BB1}, & \text{otherwise,} \end{cases} \tag{3.1}$$

where $\tau_k > 0$. In addition, we take $\alpha_1 = \alpha_1^{SD}$ and $\alpha_2 = \alpha_2^{BB1}$ for quadratic optimization. The simplest way to update τ_k in (3.1) is setting it to some constant $\tau \in (0, 1)$ for all k (see [45]). For this fixed scheme, the performance of the method (3.1) may heavily depend on the value of τ. Another strategy is to update τ_k dynamically by

$$\tau_{k+1} = \begin{cases} \tau_k/\gamma, & \text{if } \alpha_k^{BB2}/\alpha_k^{BB1} < \tau_k; \\ \tau_k\gamma, & \text{otherwise,} \end{cases} \tag{3.2}$$

for some $\gamma > 1$, see [6] for example. Clearly, the fixed scheme is a special case of (3.2) with $\gamma = 1$. In what follows, we will present an intuitive comparison between the fixed and dynamic schemes.

We tested the new method (3.1) on some randomly generated quadratic problems [42]. The objective function is given by

$$f(x) = (x - x^*)^{\mathsf{T}}\text{diag}\{v_1, \ldots, v_n\}(x - x^*), \tag{3.3}$$

where x^* was randomly generated with components in $[-10, 10]$, $v_1 = 1$, $v_n = \kappa$, and v_j, $j = 2, \ldots, n-1$, were generated between 1 and κ by the *rand* function in Matlab. The null vector was employed as the starting point. We terminated the method if either the number of iteration exceeds 20000 or $\|g_k\|_2 \leq \epsilon\|g_1\|_2$, where ϵ is a given tolerance. For each problem, we tested three different values of tolerances $\epsilon = 10^{-6}$, 10^{-9}, 10^{-12} and condition numbers $\kappa = 10^4$, 10^5, 10^6. We randomly generated 10 instances of the problem for each value of κ or ϵ.

TABLE 2
The average number of iterations required by the method (3.1) with fixed and dynamic schemes on quadratic problem (3.3).

n	ϵ	Fixed scheme (τ)					Dynamic scheme (τ_1)				
		0.1	0.2	0.5	0.7	0.9	0.1	0.2	0.5	0.7	0.9
1000	1e-06	211.4	202.2	198.7	194.8	194.4	203.6	194.3	198.1	202.4	195.7
	1e-09	616.2	688.7	767.0	888.0	1097.5	563.3	548.7	598.5	566.4	576.9
	1e-12	898.0	954.0	1100.0	1127.8	1824.8	811.9	813.6	845.7	830.9	831.1
10000	1e-06	263.2	261.3	232.8	233.9	244.9	251.1	245.6	250.1	250.0	243.3
	1e-09	1223.9	1188.0	1520.6	1537.7	1925.6	1092.3	1206.6	1196.4	1240.6	1229.3
	1e-12	1954.9	1793.6	2085.1	2198.6	3230.3	1949.5	1932.3	2034.2	2055.1	2055.2

For the fixed scheme, the parameter τ varies from $\{0.1, 0.2, 0.5, 0.7, 0.9\}$. The dynamic scheme also uses these values for τ_1 and takes the value $\gamma = 1.02$. Table 2 presents the average number of iterations of the two schemes. Clearly, when $n = 1000$, the dynamic scheme outperforms the fixed scheme for most values of τ_1. When $n = 10000$, the dynamic scheme is better than or comparable to the fixed one. Moreover, performance of the dynamic scheme is less dependent on the value of τ_1. Hence, in what follows, we will concentrate on the dynamic scheme unless otherwise specified.

The R-linear global convergence of the new method (3.1) can be established by using the Property (A) in [11]. We say that the stepsize α_k has Property (A) if 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 $\alpha_k^{-1} \geq \frac{2}{3}\lambda_{l+1}$, where $G(k,l) = \sum_{i=1}^{l}(g_k^{(i)})^2$.

THEOREM 3.1. *Suppose that the sequence $\{\|g_k\|\}$ is generated by the method (3.1) applied to solve problem (2.1) with $A = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ and $1 = \lambda_1 < \lambda_2 < \ldots < \lambda_n$. Further assume that $|\alpha_{k-1}^{BB1} - \alpha_k^{BB1}| > \tilde{\epsilon}$ for some $\tilde{\epsilon} > 0$ and $k > 1$. Then either $g_k = 0$ for some finite k or the sequence $\{\|g_k\|\}$ converges to zero R-linearly.*

Proof. By Theorem 4.1 in [11], we are suffice to show the stepsize (3.1) satisfies Property (A).

When $\alpha_k = \min\{\alpha_{k-1}^{BB2}, \alpha_k^{BB2}\}$ or $\alpha_k = \alpha_k^{BB1}$, we have that $\lambda_1 \leq (\alpha_k^{BB1})^{-1} \leq \alpha_k^{-1} \leq \lambda_n$. For the case $\alpha_k = \alpha_k^{new}$, by (2.18) and (2.20) we obtain that

$$\lambda_1 \leq (\alpha_k^{BB2})^{-1} \leq \alpha_k^{-1} \leq \frac{\phi_2}{\phi_3} = \frac{1}{\alpha_k^{BB2}} + \frac{\phi_1}{\phi_3}\alpha_k^{BB1} \leq \lambda_n + \frac{2\lambda_n^2}{\tilde{\epsilon}\lambda_1^2}.$$

Thus, condition (i) always holds with $M_1 = \lambda_n + \frac{2\lambda_n^2}{\tilde{\epsilon}\lambda_1^2}$.

By noting that $\alpha_k^{-1} \geq (\alpha_k^{BB1})^{-1} = (\alpha_{k-1}^{SD})^{-1}$, and choosing $m = 2$ and $M_2 = 2$, we can prove condition (ii) holds in the same way as Theorem 3.1 in [31]. This completes the proof. ∎

To further illustrate the efficiency of the new method (3.1) for quadratic optimization, we compared it with the following gradient methods:

(i) BB1 [3]: the original BB method using α_k^{BB1};
(ii) DY [17]: the Dai-Yuan monotone gradient method;
(iii) ABB [45]: the adaptive BB method;
(iv) ABBmin1 [25]: a gradient method adaptively uses α_k^{BB1} and $\min\{\alpha_j^{BB2} : j = \max\{1, k-m\}, \ldots, k\}$;
(v) ABBmin2 [25]: a gradient method adaptively uses α_k^{BB1} and a short stepsize in [25];
(vi) SDC [20]: a gradient method takes h SD iterates followed by s steps with the same α_k^{DY}.

Firstly, we tested the methods on the problem (3.3). We employed the same settings for initial points, condition numbers and tolerances as before. Five different distributions of the diagonal elements v_j, $j = 2, \ldots, n-1$, were generated (see Table 3). The problem dimension was set to $n = 10000$.

TABLE 3
Distributions of $\{v_j : j = 2, \ldots, n-1\}$.

Set	Spectrum
1	$\{v_2, \ldots, v_{n-1}\} \subset (1, \kappa)$
2	$\{v_2, \ldots, v_{n/5}\} \subset (1, 100)$ $\{v_{n/5+1}, \ldots, v_{n-1}\} \subset (\frac{\kappa}{2}, \kappa)$
3	$\{v_2, \ldots, v_{n/2}\} \subset (1, 100)$ $\{v_{n/2+1}, \ldots, v_{n-1}\} \subset (\frac{\kappa}{2}, \kappa)$
4	$\{v_2, \ldots, v_{4n/5}\} \subset (1, 100)$ $\{v_{4n/5+1}, \ldots, v_{n-1}\} \subset (\frac{\kappa}{2}, \kappa)$
5	$\{v_2, \ldots, v_{n/5}\} \subset (1, 100)$ $\{v_{n/5+1}, \ldots, v_{4n/5}\} \subset (100, \frac{\kappa}{2})$ $\{v_{4n/5+1}, \ldots, v_{n-1}\} \subset (\frac{\kappa}{2}, \kappa)$

For the SDC method, the parameter pair (h, s) was set to $(8, 6)$, which is more

efficient than other choices. As suggested in [25], $\tau = 0.15$ was used for the ABB method. We tested the new method (3.1), the ABBmin1 and ABBmin2 methods using fixed ($\gamma = 1$) and dynamic ($\gamma > 1$) choices of τ. It is observed that ABBmin1 using fixed τ performs better than the dynamic scheme. Thus, we chose fixed τ of ABBmin1 from $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ such that ABBmin1 achieves the best performance for each given $m \in \{3, 5, 9\}$. In particular, the pair (m, τ) for ABBmin1 was set to $(3, 0.9), (5, 0.8)$ and $(9, 0.7)$. For ABBmin2, we chose τ_1 from the above set to get the best performance for each given $\gamma \in \{1, 1.02, 1.1\}$, and set the pair (γ, τ_1) to $(1, 0.4), (1.02, 0.3), (1.1, 0.7)$. Similarly, the pair (γ, τ_1) used by the method (3.1) was set to $(1, 0.2), (1.02, 0.2), (1.1, 0.4)$. Table 4 presents the average number of iterations required by the compared methods for given tolerances. It can be seen that the new method (3.1) clearly outperforms the the BB1, DY, SDC, ABB and ABBmin1 methods. As compared with ABBmin2, our method (3.1) often performs better than it for the second to fourth problem sets, and is very competitive with it for the first and last problem sets.

TABLE 4

The average number of iterations required by the method (3.1), the BB1, DY, SDC, ABB, ABBmin1 and ABBmin2 methods on quadratic problem (3.3) with spectral distributions in Table 3.

ϵ	BB1	DY	SDC	ABB	ABBmin1 (m,τ)			ABBmin2 (γ,τ_1)			method (3.1) (γ,τ_1)		
					(3,0.9)	(5,0.8)	(9,0.7)	(1,0.4)	(1.02,0.3)	(1.1,0.7)	(1,0.2)	(1.02,0.2)	(1.1,0.4)
problem set 1													
10^{-6}	279.7	252.4	245.7	265.0	268.7	256.1	280.0	249.6	242.8	251.5	247.0	242.9	264.4
10^{-9}	2584.2	2228.3	2010.4	1418.8	2004.0	1910.6	1703.9	1016.4	1266.1	1503.9	1256.8	1145.0	1330.8
10^{-12}	5849.8	6026.8	4132.6	2030.4	4259.1	4442.0	3536.9	1615.0	2409.7	2771.6	1778.4	1899.5	2180.9
problem set 2													
10^{-6}	314.9	273.9	174.7	283.7	175.4	156.6	156.9	97.6	99.1	97.4	105.9	102.6	98.4
10^{-9}	1641.8	1320.7	737.7	1446.2	786.1	673.3	579.0	520.9	411.1	391.4	391.0	378.4	374.8
10^{-12}	2950.6	2532.2	1274.2	2769.9	1363.0	1250.1	1026.5	932.6	706.0	674.6	664.0	654.5	637.9
problem set 3													
10^{-6}	403.7	326.6	210.1	349.6	214.1	205.6	183.3	133.9	128.5	125.4	131.7	134.5	122.7
10^{-9}	1728.4	1610.5	755.7	1562.6	819.8	721.7	624.7	597.2	447.8	428.7	425.5	415.3	403.2
10^{-12}	2994.1	2748.5	1254.4	2764.9	1486.8	1273.5	1062.3	1013.8	751.0	693.0	713.5	676.6	669.1
problem set 4													
10^{-6}	511.6	408.6	255.2	429.6	281.0	233.2	223.1	151.5	151.1	154.6	153.7	148.5	144.2
10^{-9}	1907.7	1636.0	850.9	1737.5	927.6	804.0	672.8	607.0	475.2	455.6	442.2	451.9	435.2
10^{-12}	3150.2	2800.0	1399.4	2911.8	1569.7	1295.0	1060.5	1049.0	786.3	736.3	708.8	724.5	693.4
problem set 5													
10^{-6}	848.3	689.0	678.0	702.3	726.4	694.8	707.0	628.3	621.5	648.2	649.4	656.1	665.8
10^{-9}	4026.1	3973.2	3276.1	2989.0	3524.1	3465.4	3350.2	2607.0	2789.6	2777.8	2680.5	2713.0	2891.9
10^{-12}	7163.7	7795.7	6113.8	5201.9	5847.2	5615.3	5695.9	4573.9	4960.8	5101.9	4593.6	4720.9	4707.8

Furthermore, we compared the above methods on the non-rand quadratic problem [20] with A being a diagonal matrix given by

$$A_{jj} = 10^{\frac{\log_{10}\kappa}{n-1}(n-j)}, \quad j = 1, \ldots, n, \tag{3.4}$$

and $b = 0$. The problem dimensional was also set as $n = 10000$. The pair (h, s) used for the SDC method was set to (30, 2), which sounds to provide better performance than some other choices. The ABB method employed the same parameter settings as above. We chose parameters in the same way as before for our method (3.1) and the ABBmin1 and ABBmin2 methods. In particular, the pair (m, τ) for ABBmin1 was set to $(3, 0.4), (5, 0.7), (9, 0.5)$ and the pair (γ, τ_1) for ABBmin2 and our method (3.1) was set to $(1, 0.7), (1.02, 0.9), (1.1, 0.2)$ and $(1, 0.4), (1.02, 0.2), (1.1, 0.8)$, respectively. The average number of iterations over 10 different starting points with entries randomly generated in $[-10, 10]$ are presented in Table 5. We see that our method (3.1)

again wins the BB1, DY, SDC, ABB and ABBmin1 methods. Moreover, the method (3.1) usually outperforms ABBmin2 using the pairs $(1.02, 0.9)$ and $(1.1, 0.2)$, and is comparable to ABBmin2 using the pair $(1, 0.7)$. We would like to emphasize that the stepsize used by ABBmin2 is associated with the Hessian of the objective which makes it difficult to apply ABBmin2 to minimize general non-quadratic functions.

TABLE 5
The average number of iterations required by the method (3.1), the BB1, DY, SDC, ABB, ABBmin1 and ABBmin2 methods on problem (3.4) with $n = 10000$.

ϵ	BB1	DY	SDC	ABB	ABBmin1 (m, τ)			ABBmin2 (γ, τ_1)			method (3.1) (γ, τ_1)		
					(3,0.4)	(5,0.7)	(9,0.5)	(1,0.7)	(1.02,0.9)	(1.1,0.2)	(1,0.4)	(1.02,0.2)	(1.1,0.8)
$\kappa = 10^4$													
10^{-6}	718.2	528.1	568.7	522.1	572.9	556.3	522.9	478.6	486.9	507.3	487.7	499.4	522.5
10^{-9}	1264.9	985.2	990.8	941.7	1059.1	968.8	952.8	858.3	917.0	910.4	895.3	915.1	933.3
10^{-12}	1927.9	1437.0	1423.9	1350.7	1519.3	1394.7	1433.7	1251.1	1342.1	1344.8	1279.9	1327.9	1295.8
$\kappa = 10^5$													
10^{-6}	1407.1	1198.5	1244.4	1275.7	1173.4	1346.2	1189.5	1100.0	1065.6	1197.5	1108.7	1122.2	1121.7
10^{-9}	3801.1	3174.3	2661.9	2625.9	2903.2	2782.0	2991.4	2425.2	2562.8	2666.0	2435.5	2486.8	2592.4
10^{-12}	5606.6	4923.8	4039.4	3839.0	4390.2	4237.6	4336.7	3607.7	3874.0	3958.2	3708.8	3801.5	3882.7
$\kappa = 10^6$													
10^{-6}	2549.0	2063.6	2040.4	2512.6	2051.4	2414.8	2269.6	2220.0	1900.9	1964.3	2012.9	2051.0	1916.2
10^{-9}	10573.5	10685.9	7163.6	8314.9	8358.2	8212.0	7644.2	6723.8	7099.5	7040.1	6589.5	6635.5	6825.5
10^{-12}	17026.3	18415.4	11463.8	12473.9	14568.9	13446.8	13295.1	10787.0	11234.4	11436.2	10420.5	10890.3	10915.5

3.2. Unconstrained optimization. To extend the new method (3.1) for minimizing a general smooth function $f(x)$,

$$\min_{x\in\mathbb{R}^n} f(x), \tag{3.5}$$

we usually need to incorporate some line search to ensure global convergence. As pointed out by Fletcher [23], one important feature of BB-like methods is the inherent nonmonotone property. Some nonmonotone line search is often employed to gain good numerical performance [12, 18, 38, 44]. Here we would like to adopt the Grippo-Lampariello-Lucidi (GLL) nonmonotone line search [28], which accepts λ_k when it satisfies

$$f(x_k + \lambda_k d_k) \le f_r + \sigma\lambda_k g_k^{\mathsf{T}} d_k, \tag{3.6}$$

where σ is a positive parameter and the reference value f_r is the maximal function value in recentest available M iterations; namely, $f_r = \max_{0\le i\le \min\{k,M-1\}} f(x_{k-i})$. This strategy was firstly incorporated for unconstrained optimization in the global BB (GBB) method by Raydan [38] and performs well in practice.

The combination of the gradient method with the stepsize formula (3.1) and the GLL nonmonotone line search yields a new algorithm, Algorithm 3.1, for unconstrained optimization. Under standard assumptions, global convergence of Algorithm 3.1 can be established in the same way as Theorem 2.2 in [4]. Moreover, the convergence rate is R-linear for strongly convex objective functions, see Theorem 3.4 in [33].

We compared Algorithm 3.1 with the GBB [38] and ABBmin [21, 25] methods for unconstrained optimization problems from the CUTEst collection [27] with dimension less than or equal to 10000. We deleted the problem if either it can not be solved in 200000 iterations by any of the algorithms or the function evaluation exceeds one million, and 147 problems are left.

Algorithm 3.1 A gradient method for unconstrained optimization

Initialization: $x_1 \in \mathbb{R}^n$, $\alpha_{\max} \geq \alpha_{\min} > 0$, $\alpha_1 \in [\alpha_{\min}, \alpha_{\max}]$, $\tau_1 > 0$, $\gamma > 1$, $\epsilon, \sigma, \delta \in (0,1)$, $M \in \mathbb{N}$, $k := 1$.
while $\|g_k\|_\infty > \epsilon$ **do**
  $d_k = -g_k$, $\lambda_k = \alpha_k$
  $f_r = \max_{0 \leq i \leq \min\{k, M-1\}} f(x_{k-i})$
  while the condition (3.6) does not hold **do**
    $\lambda_k = \delta \lambda_k$
  end while
  $x_{k+1} = x_k + \lambda_k d_k$
  if $s_k^\mathsf{T} y_k > 0$ **then**
    if $\alpha_k^{BB2}/\alpha_k^{BB1} < \tau_k$ and $s_{k-1}^\mathsf{T} y_{k-1} > 0$ **then**
      if $\alpha_{k+1}^{new} > 0$ **then**
        $\alpha_{k+1} = \min\{\alpha_k^{BB2}, \alpha_{k+1}^{BB2}, \alpha_{k+1}^{new}\}$
      else
        $\alpha_{k+1} = \min\{\alpha_k^{BB2}, \alpha_{k+1}^{BB2}\}$
      end if
      $\tau_{k+1} = \tau_k/\gamma$
    else
      $\alpha_{k+1} = \alpha_{k+1}^{BB1}$
      $\tau_{k+1} = \tau_k \gamma$
    end if
  else
    $\alpha_{k+1} = \min\{1/\|g_k\|_\infty, \|x_k\|_\infty/\|g_k\|_\infty\}$
  end if
  Chop extreme values of the stepsize such that $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$
  $k = k + 1$
end while

To implement Algorithm 3.1, we use $\alpha_1 = \|x_1\|_\infty/\|g_1\|_\infty$ if $x_1 \neq 0$ and otherwise $\alpha_1 = 1/\|g_1\|_\infty$. In addition, we set $\alpha_2 = \alpha_2^{BB1}$ for the case $\alpha_2^{BB1} > 0$ and otherwise chose $\alpha_2 = \min\{1/\|g_2\|_\infty, \|x_2\|_\infty/\|g_2\|_\infty\}$. The following parameters were employed for Algorithm 3.1:

$$\alpha_{\min} = 10^{-10},\ \alpha_{\max} = 10^6,\ M = 10,\ \sigma = 10^{-4},\ \delta = 0.5,\ \tau_1 = 0.2,\ \gamma = 1.02.$$

Default parameter settings were used for the GBB and ABBmin methods. The stopping condition $\|g_k\|_\infty \leq 10^{-6}$ was adopted for all the three methods.

The number of iterations, the total number of function evaluations and the CPU time in seconds of the compared methods are reported in the Appendix, which show that both Algorithm 3.1 and ABBmin are faster than GBB. Performance profiles [22] of Algorithm 3.1 and ABBmin using iteration and CPU time metrics are plotted in Figure 1. For each method, the vertical axis of the figure shows the percentage of problems the method solves within the factor ρ of the minimum value of the metric. From Figure 1, it can be seen that Algorithm 3.1 performs better than ABBmin.

4. Extreme eigenvalues computation. In this section, we consider the problem of computing several extreme eigenvalues of large-scale real symmetric matrices.

For a given $n \times n$ real symmetric positive definite matrix A, we are interested in the first $r \ll n$ largest/smallest eigenvalues and their corresponding eigenvectors, which

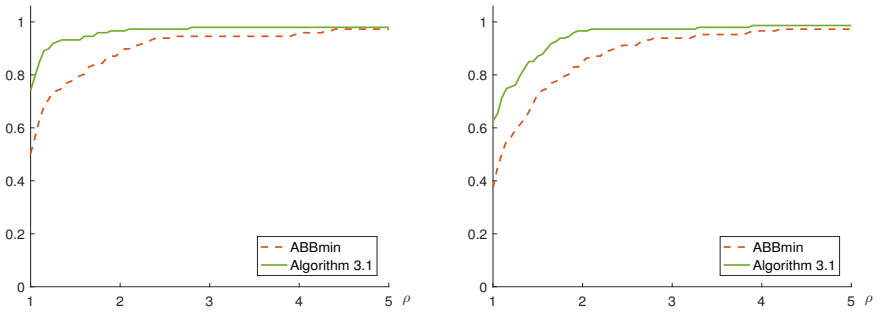

FIG. 1. *Performance profiles of Algorithm 3.1 and ABBmin on 147 unconstrained problems from CUTEst, iteration (left) and CPU time (right) metrics.*

has important applications in scientific and engineering computing such as principal component analysis [19] and electronic structure calculation [29]. This problem can be formulated as an unconstrained optimization problem [1]

$$\min_{X\in\mathbb{R}^{n\times r}} \quad \mathrm{tr}(X^{\mathsf{T}}AX(X^{\mathsf{T}}X)^{-1}) \tag{4.1}$$

or a constrained optimization problem with orthogonality constraints [40, 41]

$$\min_{X\in\mathbb{R}^{n\times r}} \quad \mathrm{tr}(X^{\mathsf{T}}AX) \quad \text{s.t.} \quad X^{\mathsf{T}}X = I_r, \tag{4.2}$$

where I_r denotes the $r \times r$ identity matrix. However, it is not easy to calculate the inverse or orthogonalization of a matrix, especially in the case of large dimension. To avoid these difficulties, Jiang et al. [34] provides the following equivalent unconstrained reformulation

$$\min_{X\in\mathbb{R}^{n\times r}} P_\mu(X) = \frac{1}{4}tr(X^{\mathsf{T}}XX^{\mathsf{T}}X) + \frac{1}{2}tr(X^{\mathsf{T}}(A-\mu I_n)X), \tag{4.3}$$

where $\mu > 0$ is a scaling parameter. They proposed the so-called EigUncABB method for solving (4.3), which is very competitive with the Matlab function EIGS and other recent methods.

In order to apply Algorithm 3.1 to problem (4.3), we replace the two BB stepsizes α_k^{BB1} and α_k^{BB2} in the calculations of α_k^{new} and (3.1) by

$$\alpha_k^{MBB1} = \frac{\mathrm{tr}(S_{k-1}^{\mathsf{T}}S_{k-1})}{\mathrm{tr}(S_{k-1}^{\mathsf{T}}Y_{k-1})} \quad \text{and} \quad \alpha_k^{MBB2} = \frac{\mathrm{tr}(S_{k-1}^{\mathsf{T}}Y_{k-1})}{\mathrm{tr}(Y_{k-1}^{\mathsf{T}}Y_{k-1})},$$

respectively, where $S_{k-1} = X_k - X_{k-1}$ and $Y_{k-1} = \nabla P_\mu(X_k) - \nabla P_\mu(X_{k-1})$. Notice that the above two modified BB stepsizes are different from those employed by EigUncABB, which uses $|\alpha_k^{MBB1}|$ and $|\alpha_k^{MBB2}|$ in an alternate manner. We shall refer to the modified α_k^{new} as α_k^{Mnew}.

Since the evaluation of $P_\mu(X)$ is expensive for large-dimension X, we shall adopt the Dai-Fletcher nonmonotone line search [12] to reduce the number of function evaluations. In particular, it uses f_{best}, f_c, m and M to update the reference value f_r in (3.6), where $f_{best} = \min_{1\le j\le k} f(x_j)$ is the current best function value, f_c is the

maximum value of the objective function since the value of f_{best} was found, m is the number of iterations since the value of f_{best} was obtained, and M is a preassigned number. The value of f_r is unchanged if the method finds a better function value in M iterations. Otherwise, if $m = M$, f_r is reset to f_c and f_c is reset to the current function value. See [12] for details on this line search.

Our method for problem (4.3) is formally presented in Algorithm 4.1. The global convergence can be established using the same arguments as the one in Theorem 3.2 of [18].

Algorithm 4.1 A gradient method for extreme eigenvalues problems

Initialization: $X_1 \in \mathbb{R}^{n\times r}$, $\tau_1 > 0$, $\alpha_{\max} \geq \alpha_{\min} > 0$, $\alpha_1 \in [\alpha_{\min}, \alpha_{\max}]$, $\tau_1 > 0$, $\gamma > 1$, $\epsilon, \sigma, \delta \in (0, 1)$, $M \in \mathbb{N}$, $m = 0$, $f_r = f_{best} = f_c = P_\mu(X_1)$, $k := 1$.
while $\|\nabla P_\mu(X_k)\|_F > \epsilon$ **do**
  if $P_\mu(X_k) < f_{best}$ **then**
    $f_{best} = P_\mu(X_k)$, $f_c = P_\mu(X_k)$, $m = 0$
  else
    $f_c = \max\{f_c, P_\mu(X_k)\}$, $m = m + 1$
    if $m = M$ **then**
      $f_r = P_\mu(X_k)$, $f_c = P_\mu(X_k)$, $m = 0$
    end if
  end if
  $d_k = -\nabla P_\mu(X_k)$, $\lambda_k = \alpha_k$
  while the condition (3.6) does not hold **do**
    $\lambda_k = \delta\lambda_k$
  end while
  $X_{k+1} = X_k + \lambda_k d_k$
  if $\mathrm{tr}(S_k^{\mathsf{T}} Y_k) > 0$ **then**
    if $\alpha_k^{MBB2}/\alpha_k^{MBB1} < \tau_k$ and $\mathrm{tr}(S_{k-1}^{\mathsf{T}} Y_{k-1}) > 0$ **then**
      if $\alpha_{k+1}^{Mnew} > 0$ **then**
        $\alpha_{k+1} = \min\{\alpha_k^{MBB2}, \alpha_{k+1}^{MBB2}, \alpha_{k+1}^{Mnew}\}$
      else
        $\alpha_{k+1} = \min\{\alpha_k^{MBB2}, \alpha_{k+1}^{MBB2}\}$
      end if
      $\tau_{k+1} = \tau_k/\gamma$
    else
      $\alpha_{k+1} = \alpha_{k+1}^{MBB1}$
      $\tau_{k+1} = \tau_k\gamma$
    end if
  else
    $\alpha_{k+1} = |\alpha_{k+1}^{MBB1}|$
  end if
  Chop extreme values of the stepsize such that $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$
  $k = k + 1$
end while

We compared Algorithm 4.1 with EigUncABB on extreme eigenvalues problems which involve a $16,000 \times 16,000$ matrix, say A, generated by the *laplacian* function in Matlab. The matrix A can be viewed as the 3D negative Laplacian on a rectangular finite difference grid.

For Algorithm 4.1, we choose $\alpha_1 = \|\nabla P_\mu(X_1)\|_F^{-1}$ and $\alpha_2 = |\alpha_2^{MBB1}|$. The other parameters in Algorithm 4.1 were chosen in the same way as in the above section and default parameters for EigUncABB were employed. As suggested in [34], μ was initialized to $1.01 \times \lambda_{\bar{r}}(X_1^\mathsf{T} A X_1)$, where $\bar{r} = \max\{\lfloor 1.1r \rfloor, 10\}$ with $\lfloor\cdot\rfloor$ denoting the nearest integer less than or equal to the corresponding element. Moreover, it is updated by $\mu = 1.01\lambda_{\bar{r}}(X_k^\mathsf{T} A X_k)$ when $\|\nabla P_\mu(X_k)\|_F \le 0.1^j \|\nabla P_\mu(X_1)\|_F$ and $j \le j_{\max}$ for some integers j and $j_{\max}$. Specifically, the value of $j_{\max}$ was set to 3. The initial value of j was set to 1 and it will be increased by one if μ is updated. We generated initial points by the following Matlab codes

$$\text{seed} = 100;\ \text{rng(seed,'twister')};\ X_1 = \text{randn}(n, r);\ X_1 = \text{orth}(X_1).$$

Both methods were terminated provided $\|\nabla P_\mu(X_k)\|_F \le 10^{-3}$.

To measure the quality of computed solutions, we calculate the relative eigenvalue error and residual error of the i-th eigenpair as

$$\text{err}_i = \frac{|\bar{\lambda}_i - \lambda_i|}{\max\{1, |\lambda_i|\}} \quad \text{and} \quad \text{resi}_i = \frac{\|A\bar{u}_i - \bar{\lambda}_i \bar{u}_i\|_2}{\max\{1, |\bar{\lambda}_i|\}},$$

respectively. Here λ_i is the true i-th eigenvalue, $\bar{u}_i$ and $\bar{\lambda}_i$ are the i-th eigenvector and eigenvalue obtained by the compared algorithms, respectively.

TABLE 6
Results of EigUncABB and Algorithm 4.1 on extreme eigenvalues problems.

	EigUncABB					Algorithm 4.1				
r	resi	err	iter	nfe	time	resi	err	iter	nfe	time
20	1.4e-05	9.8e-09	140	160	2.25	4.2e-06	3.3e-09	136	144	2.10
50	1.6e-05	1.8e-08	146	170	4.58	2.3e-05	2.1e-07	128	133	3.90
100	7.1e-05	2.0e-09	167	183	10.39	1.4e-04	1.4e-08	137	146	8.70
200	3.8e-06	5.2e-10	192	216	28.97	1.1e-06	5.4e-10	168	175	24.26
300	4.1e-06	8.1e-11	160	188	44.98	4.9e-06	2.4e-11	149	163	39.91
400	6.9e-07	6.7e-13	148	168	64.40	4.0e-06	3.7e-11	166	178	67.94
500	3.2e-06	2.9e-11	201	227	122.32	6.8e-07	7.7e-14	171	188	101.95
600	3.2e-06	1.6e-12	204	238	169.76	4.4e-06	6.4e-12	172	190	137.86
700	6.0e-07	1.1e-12	185	212	199.48	2.6e-06	3.3e-12	166	185	177.61
800	2.4e-06	1.9e-11	208	234	270.53	1.3e-05	4.7e-11	194	211	244.68
900	3.5e-06	7.3e-12	171	194	268.97	1.0e-07	1.9e-13	164	175	243.93
1000	1.3e-06	2.1e-12	211	240	388.06	1.1e-06	9.7e-13	194	208	341.42

In Table 6, "resi" denotes mean values of resi_i, "err" denotes mean values of err_i, $i = 1, \cdots, r$, and "iter" is the number of iterations, "nfe" is the total number of function evaluations and "time" is the CPU time in seconds. From Table 6, we can see that Algorithm 4.1 is comparable to EigUncABB in the sense of relative eigenvalue error and residual error. Moreover, Algorithm 4.1 outperforms EigUncABB in terms of iterations, function evaluations and CPU time for most values of r.

5. Special constrained optimization. In this section, we extend Algorithm 3.1 to solve two special constrained optimization problems of the form

$$\min_{x \in \Omega} \quad f(x), \tag{5.1}$$

where $\Omega \subseteq \mathbb{R}^n$ is a closed convex set and f is a Lipschitz continuously differentiable function on Ω.

Our algorithm for solving problem (5.1) falls into the gradient projection category, which calculates the search direction by

$$d_k = P_\Omega(x_k - \alpha_k g_k) - x_k, \tag{5.2}$$

with $P_\Omega(\cdot)$ being the Euclidean projection onto Ω and α_k being the stepsize. For a general closed convex set Ω, the projection $P_\Omega(\cdot)$ may not be easy to compute. However, for the box-constrained optimization, where $\Omega = \{x \in \mathbb{R}^n |\ l \le x \le u\}$, we have $P_\Omega(x) = \max\{l, \min\{x, u\}\}$. Here, $l \le x \le u$ means componentwise; namely, $l_i \le x_i \le u_i$ for all $i = 1, \ldots, n$. In addition, for singly linearly box-constrained optimization, the projection to its feasible set $\Omega = \{x \in \mathbb{R}^n |\ l \le x \le u,\ a^\mathsf{T} x = b\}$ with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$ can efficiently be computed by for example the secant-like algorithm developed in [13]. In what follows, we will focus on box-constrained optimization and singly linearly box-constrained optimization.

For the calculation of the stepsize α_k, we can simply utilize the formula (3.1). However, this unmodified stepsize cannot capture the feature of the constraints well in the context of constrained optimization. Therefore we will employ the following stepsize

$$\alpha_k = \begin{cases} \min\{\bar\alpha_{k-1}^{BB2}, \bar\alpha_k^{BB2}, \bar\alpha_k^{new}\}, & \text{if } \bar\alpha_k^{BB2}/\bar\alpha_k^{BB1} < \tau_k; \\ \bar\alpha_k^{BB1}, & \text{otherwise,} \end{cases} \tag{5.3}$$

where τ_k is updated by the rule (3.2), $\bar\alpha_k^{BB1}$ and $\bar\alpha_k^{BB2}$ are two modified BB stepsizes specified in the following subsections, and the stepsize $\bar\alpha_k^{new}$ is defined by replacing α_k^{BB1} and α_k^{BB2} in α_k^{new} with $\bar\alpha_k^{BB1}$ and $\bar\alpha_k^{BB2}$, respectively.

We present our method for problem (5.1) in Algorithm 5.1. For the first two stepsizes in Algorithm 5.1, we employ the same choices as Algorithm 3.1 but replace $\|g_j\|_\infty$ by $\|P_\Omega(x_j - g_j) - x_j\|_\infty$ for $j = 1, 2$ and α_2^{BB1} by $\bar\alpha_2^{BB1}$, respectively. Under standard assumptions, global convergence of Algorithm 5.1 can be established as Theorem 2.2 in [4], and the rate is R-linear for strongly convex objective functions, see Theorem 3.4 in [33].

5.1. Box-constrained optimization. As pointed out by [31, 32], for the box-constrained problem, a projected gradient method often eventually solves an unconstrained problem in the subspace corresponding to free variables. So, it is useful to modify the stepsize by considering those free variables only. To this end, we consider to replace the gradients of f in the two BB stepsizes by those of the Lagrangian function

$$\mathcal{L}(x, \delta, \zeta) = f(x) - \delta^\mathsf{T}(x - l) - \zeta^\mathsf{T}(u - x),$$

where $\delta, \zeta \in \mathbb{R}^n$ are Lagrange multipliers. That is,

$$\bar\alpha_k^{BB1} = \frac{s_{k-1}^\mathsf{T} s_{k-1}}{s_{k-1}^\mathsf{T} \bar y_{k-1}} \quad \text{and} \quad \bar\alpha_k^{BB2} = \frac{s_{k-1}^\mathsf{T} \bar y_{k-1}}{\bar y_{k-1}^\mathsf{T} \bar y_{k-1}}, \tag{5.4}$$

where

$$\begin{aligned} \bar y_{k-1} &= \nabla_x \mathcal{L}(x_k, \delta_k, \zeta_k) - \nabla_x \mathcal{L}(x_{k-1}, \delta_{k-1}, \zeta_{k-1}) \\ &= y_{k-1} - (\delta_k - \delta_{k-1}) + (\zeta_k - \zeta_{k-1}). \end{aligned} \tag{5.5}$$

Algorithm 5.1 Projected gradient method for problem (5.1)

Initialization: $x_1 \in \mathbb{R}^n$, $\alpha_{\max} \geq \alpha_{\min} > 0$, $\alpha_1 \in [\alpha_{\min}, \alpha_{\max}]$, $\tau_1 > 0$, $\gamma > 1$, $\epsilon, \sigma, \delta \in (0,1)$, $M \in \mathbb{N}$, $k := 1$.
while $\|P_\Omega(x_k - g_k) - x_k\|_\infty > \epsilon$ **do**
  Compute the search direction d_k by (5.2), $\lambda_k = \alpha_k$
  $f_r = \max_{0 \leq i \leq \min\{k, M-1\}} f(x_{k-i})$
  while the condition (3.6) does not hold **do**
    $\lambda_k = \delta \lambda_k$
  end while
  $x_{k+1} = x_k + \lambda_k d_k$
  if $s_k^{\mathsf{T}} \bar{y}_k > 0$ **then**
    if $\bar{\alpha}_k^{BB2} / \bar{\alpha}_k^{BB1} < \tau_k$ and $s_{k-1}^{\mathsf{T}} \bar{y}_{k-1} > 0$ **then**
      if $\bar{\alpha}_{k+1}^{new} > 0$ **then**
        $\alpha_{k+1} = \min\{\bar{\alpha}_k^{BB2}, \bar{\alpha}_{k+1}^{BB2}, \bar{\alpha}_{k+1}^{new}\}$
      else
        $\alpha_{k+1} = \min\{\bar{\alpha}_k^{BB2}, \bar{\alpha}_{k+1}^{BB2}\}$
      end if
      $\tau_{k+1} = \tau_k / \gamma$
    else
      $\alpha_{k+1} = \bar{\alpha}_{k+1}^{BB1}$
      $\tau_{k+1} = \tau_k \gamma$
    end if
  else
    $\alpha_{k+1} = \min\{1/\|P_\Omega(x_k - g_k) - x_k\|_\infty, \|x_k\|_\infty / \|P_\Omega(x_k - g_k) - x_k\|_\infty\}$
  end if
  Chop extreme values of the stepsize such that $\alpha_{k+1} \in [\alpha_{\min}, \alpha_{\max}]$
  $k = k + 1$
end while

In this way, the above two modified BB stepsizes take the constraints into consideration. Use the sets $\mathcal{A}_k$ and $\mathcal{I}_k = \mathcal{N} \setminus \mathcal{A}_k$ to estimate the active and inactive sets at x_k, respectively, where $\mathcal{N} = \{1, 2, \ldots, n\}$. Based on the above analysis, we simply set $\bar{y}_{k-1}^{(i)} = 0$ for $i \in \mathcal{A}_k$. Notice that, by the first-order optimality conditions of problem (5.1), the Lagrange multipliers for free variables are zeros. Then we set $\delta_k^{(i)} - \delta_{k-1}^{(i)} = 0$ and $\zeta_k^{(i)} - \zeta_{k-1}^{(i)} = 0$ for $i \in \mathcal{I}_k$. Hence, $\bar{y}_{k-1}$ can be written as

$$\bar{y}_{k-1}^{(i)} = \begin{cases} 0, & \text{if } i \in \mathcal{A}_k; \\ g_k^{(i)} - g_{k-1}^{(i)}, & \text{otherwise.} \end{cases} \tag{5.6}$$

In our test, we set $\mathcal{A}_k = \{i \in \mathcal{N} \mid s_{k-1}^{(i)} = 0\}$, which is suitable for box-constrained optimization. The above two modified BB stepsizes often yield better performance than the original BB stepsizes, see for example [31, 32]. Here we mention that similar modified BB stepsizes are presented in [9] for box-constrained quadratic programming.

Now we compare Algorithm 5.1 with SPG[1] [4, 5] and BoxVABBmin [9] for box-constrained problems from the CUTEst collection [27] with dimension larger than 50. Notice that SPG is a generalization of GBB with the long BB stepsize α_k^{BB1} and

[1] codes available at https://www.ime.usp.br/~egbirgin/tango/codes.php

BoxVABBmin is a variant of ABBmin with the modified BB stepsizes (5.4) using a different $\mathcal{A}_k$. Three of the problems were deleted since none of the three algorithms can solve them successfully and hence 47 problems are left in our test.

We adopted default parameters for SPG and BoxVABBmin and used the same settings for Algorithm 5.1 as for the unconstrained case. The algorithms were terminated if either $\|P_\Omega(x_k - g_k) - x_k\|_\infty \leq 10^{-6}$ or the number of iterations exceeds 200000.

The function evaluations, the number of iterations and CPU time in seconds of the compared methods are listed in Table 7. Clearly, Algorithm 5.1 and BoxVABBmin perform much better than SPG while Algorithm 5.1 outperforms BoxVABBmin for most of the problems.

TABLE 7

Results of Algorithm 5.1, SPG and BoxVABBmin on 47 box-constrained problems from the CUTEst collection.

problem	n	SPG			BoxVABBmin			Algorithm 5.1		
		nfe	iter	time	nfe	iter	time	nfe	iter	time
CHEBYQAD	100	38612	65566	198.79	6735	7464	24.29	6521	7220	23.72
DECONVB	63	4438	6977	2.81	745	810	0.03	506	586	0.04
GENROSEB	500	142	143	0.10	181	185	0.01	160	161	0.02
LINVERSE	1999	10233	16742	12.25	1209	1422	0.52	623	654	0.26
NONSCOMP	5000	46	47	0.15	43	45	0.06	47	49	0.07
BIGGSB1	5000	48380	79820	143.51	4151	4779	7.02	3307	3786	4.51
BQPGABIM	50	37	50	0.06	29	41	0.00	26	38	0.00
BQPGASIM	50	34	47	0.03	32	44	0.00	37	50	0.00
BQPGAUSS	2003	200000	345240	213.47	32561	37859	9.66	52073	60174	14.98
CHENHARK	5000	34598	58036	98.90	2948	3380	5.82	3282	3799	5.09
CVXBQP1	10000	1	2	0.01	1	2	0.00	1	2	0.00
DEGDIAG	100001	2	3	0.02	1	2	0.02	1	2	0.01
DEGTRID	100001	156	210	2.15	92	101	1.06	102	108	1.42
DEGTRID2	100001	3	4	0.07	1	2	0.03	1	2	0.03
DIXON3DQ	10000	57368	98708	227.16	5198	6068	12.95	4571	5311	9.82
DQDRTIC	5000	26	27	0.08	17	18	0.20	14	15	0.11
HARKERP2	1000	126	189	0.62	52	53	0.18	34	35	0.12
JNLBRNG1	10000	1732	2639	10.92	494	564	2.39	477	533	2.22
JNLBRNG2	10000	1979	3232	13.22	558	642	2.79	666	759	3.36
JNLBRNGA	10000	975	1519	6.24	339	379	1.76	307	352	1.24
JNLBRNGB	10000	14072	23080	89.40	2462	2858	11.63	2385	2757	12.65
NCVXBQP1	10000	1	2	0.01	2	3	0.02	2	3	0.01
NCVXBQP2	10000	71	75	0.49	74	79	0.22	69	76	0.15
NCVXBQP3	10000	120	128	0.42	122	126	0.23	114	121	0.22
NOBNDTOR	5476	533	813	3.06	223	263	1.05	209	249	0.64
OBSTCLAE	10000	691	985	4.70	237	258	1.05	294	325	1.26
OBSTCLAL	10000	304	404	2.09	154	170	0.63	158	178	0.65
OBSTCLBL	10000	369	509	2.41	168	179	0.68	179	201	0.73
OBSTCLBM	10000	319	443	2.07	137	152	0.61	151	167	0.70
OBSTCLBU	10000	385	541	2.15	192	205	1.14	170	188	0.71
PENTDI	5000	1	4	0.00	1	3	0.00	1	3	0.00
QUDLIN	5000	2	3	0.01	2	3	0.00	2	3	0.00
TESTQUAD	5000	176486	305876	460.61	4845	5666	7.14	6459	7374	9.09
TOINTQOR	50	57	58	0.05	50	51	0.00	48	49	0.00
TORSION1	5476	607	895	2.94	221	247	1.11	211	255	0.73
TORSION2	5476	540	792	3.56	214	237	0.92	207	246	0.63
TORSION3	5476	164	219	0.69	87	97	0.25	103	123	0.33
TORSION4	5476	121	155	0.56	95	101	0.29	102	116	0.31
TORSION5	5476	54	64	0.58	52	60	0.15	47	54	0.14
TORSION6	5476	68	80	0.40	65	68	0.29	56	62	0.17
TORSIONA	5476	459	668	2.35	201	229	0.88	170	201	0.57
TORSIONB	5476	500	744	2.51	218	248	1.20	170	200	0.65
TORSIONC	5476	194	266	1.29	94	106	0.34	99	123	0.35
TORSIOND	5476	184	242	1.31	91	96	0.45	102	113	0.44
TORSIONE	5476	49	58	0.27	58	66	0.28	47	54	0.18
TORSIONF	5476	73	81	0.36	59	63	0.19	56	62	0.23
TRIDIA	5000	12247	20902	32.96	2538	2925	4.50	2322	2659	2.69

5.2. Singly linearly box-constrained optimization. Now we shall consider the solution of the singly linearly box-constrained (SLB) optimization problem. To improve the BB stepsizes (5.4) by taking the constraints into consideration, similarly to (5.5), denote

$$\begin{aligned}\bar{y}_{k-1} &= \nabla_x \mathcal{L}(x_k, \delta_k, \zeta_k, t_k) - \nabla_x \mathcal{L}(x_{k-1}, \delta_{k-1}, \zeta_{k-1}, t_{k-1}) \\ &= y_{k-1} - (\delta_k - \delta_{k-1}) + (\zeta_k - \zeta_{k-1}) - (t_k - t_{k-1})a,\end{aligned} \tag{5.7}$$

where

$$\mathcal{L}(x, \delta, \zeta, t) = f(x) - \delta^{\mathsf{T}}(x - l) - \zeta^{\mathsf{T}}(u - x) - t(a^{\mathsf{T}}x - b).$$

Similarly to the box-constrained case, we set $\bar{y}_{k-1}^{(i)} = 0$ for $i \in \mathcal{A}_k$, and $\delta_k^{(i)} - \delta_{k-1}^{(i)} = 0$ and $\zeta_k^{(i)} - \zeta_{k-1}^{(i)} = 0$ for $i \in \mathcal{I}_k$. Again by the first-order optimality conditions of problem (5.1), $\nabla_x \mathcal{L}$ is zero at the solution, which yields

$$\bar{y}_{k-1}^{(\mathcal{I}_k)} = y_{k-1}^{(\mathcal{I}_k)} - (t_k - t_{k-1})a^{(\mathcal{I}_k)} = 0.$$

It is necessary to estimate the Lagrange multipliers t_k and t_{k-1} for computing $\bar{y}_{k-1}^{(\mathcal{I}_k)}$. A simple way is to estimate $t_k - t_{k-1}$ directly by

$$t_k - t_{k-1} = \frac{(a^{(\mathcal{I}_k)})^{\mathsf{T}} y_{k-1}^{(\mathcal{I}_k)}}{(a^{(\mathcal{I}_k)})^{\mathsf{T}} a^{(\mathcal{I}_k)}}.$$

Thus, we have

$$\bar{y}_{k-1}^{(i)} = \begin{cases} 0, & \text{if } i \in \mathcal{A}_k; \\ y_{k-1}^{(\mathcal{I}_k)} - \frac{(a^{(\mathcal{I}_k)})^{\mathsf{T}} y_{k-1}^{(\mathcal{I}_k)}}{(a^{(\mathcal{I}_k)})^{\mathsf{T}} a^{(\mathcal{I}_k)}} a^{(\mathcal{I}_k)}, & \text{otherwise.} \end{cases} \tag{5.8}$$

This together with (5.4) provides us the modified BB stepsizes for SLB problems. For the case that

$$\mathcal{A}_k = \{i \in \mathcal{N} \mid x_k^{(i)} = x_{k-1}^{(i)} = l_i \ \text{ or } \ x_k^{(i)} = x_{k-1}^{(i)} = u_i\} \tag{5.9}$$

(see [10]), the stepsizes in (5.4) are derived in a different way.

In what follows, we compare Algorithm 5.1 with the Dai-Fletcher method [13] and the EQ-VABBmin method [10] for random SLB problems and SLB problems arising in support vector machines. The methods were terminated once $\|x_k - x_{k-1}\|_2 \le \epsilon$ or the total number of iterations exceeds 100000. We set $\tau_1 = 0.5$ and $\gamma = 1.3$ for Algorithm 5.1 and used default parameters for the Dai-Fletcher and EQ-VABBmin methods.

5.2.1. Random SLB problems. We employ the procedure in [13] to generate random SLB problems, which is based on the generation of random SPD box-constrained quadratic test problems in [36]. In particular, it uses five parameters n, *ncond*, *ndeg*, $na(x^*)$ and $na(x_1)$ to determine the number of variables, condition number of the Hessian, amount of near-degeneracy, active variables at the solution x^* and active variables at the starting point x_1, respectively. It generates SLB problems in the following form

$$\begin{aligned} \min \quad & \frac{1}{2}x^{\mathsf{T}}Ax - c^{\mathsf{T}}x \\ s.t. \quad & l \le x \le u \\ & a^{\mathsf{T}}x = b, \end{aligned}$$

where $A = PDP^\mathsf{T}$ with $P = (I-2p_3p_3^\mathsf{T})(I-2p_2p_2^\mathsf{T})(I-2p_1p_1^\mathsf{T})$, D is a diagonal matrix whose i-th component is defined by

$$\log d_i = \frac{i-1}{n-1} ncond,\ i = 1, \ldots, n,$$

and p_i, $i = 1, 2, 3$, are random vectors whose elements are sampled from a uniform distribution in $(-1, 1)$. See [13] for more details on the generation of the problems.

In our test, we set $n = 10000$ and chose other parameters from

$$ncond \in \{4, 5, 6\},\ ndeg \in \{1, 3, 5, 7, 9\},\ na(x^*) \in \{1000, 5000, 9000\}.$$

We randomly generated one problem for each case and then got 45 problems in total. Four different starting points with $na(x_1) \in \{0, 1000, 5000, 9000\}$ were generated for each problem. The tolerance parameter ϵ was set to 10^{-8} for this test.

TABLE 8
Results of Algorithm 5.1, Dai-Fletcher and EQ-VABBmin on 45 random SLB problems.

problem	Dai-Fletcher			EQ-VABBmin			Algorithm 5.1		
	nfe	iter	time	nfe	iter	time	nfe	iter	time
1	1533.0	1499.0	2.83	1122.0	988.3	1.86	1242.5	1058.3	1.92
2	1468.8	1418.5	2.24	1228.8	1073.3	2.27	1184.5	1008.3	2.10
3	1892.0	1774.5	3.53	1163.0	1037.3	2.38	1192.8	1020.5	2.37
4	9278.8	8470.8	11.50	3380.8	2974.3	4.98	3306.0	2811.8	4.69
5	9319.5	8315.0	18.12	4618.5	4197.0	11.44	3231.3	2754.0	7.46
6	11102.8	10305.8	23.70	3991.5	3546.5	9.33	3726.5	3267.8	8.66
7	58928.5	50309.0	91.70	10431.8	9258.8	21.27	8529.8	7373.5	15.74
8	65045.0	57488.8	114.16	20566.3	19257.3	47.97	10846.3	9408.8	23.96
9	86349.5	76291.0	174.98	18019.5	16168.3	41.75	13613.0	12062.3	31.58
10	1560.3	1501.0	2.34	1262.0	1117.5	1.96	1216.3	1029.8	1.85
11	1865.5	1790.8	2.96	1335.5	1167.8	2.50	1379.3	1169.8	2.55
12	2746.3	2621.8	6.02	1251.5	1125.3	3.56	1280.8	1126.3	3.20
13	8136.5	7332.5	11.00	3467.0	3046.8	5.32	3045.5	2608.3	4.55
14	13017.0	11867.3	20.82	3956.0	3548.3	7.78	3812.8	3293.8	7.77
15	20640.3	19141.3	42.46	4789.8	4319.3	11.14	3954.3	3551.0	9.10
16	59194.3	51170.3	77.75	9077.0	8079.3	14.26	8363.5	7219.0	13.27
17	102035.5	91391.8	172.48	14853.5	13486.0	31.32	10374.3	9116.5	20.87
18	154659.3	139044.5	290.37	26464.5	23928.8	59.50	14018.5	12513.3	30.44
19	1674.8	1643.3	2.40	1246.8	1096.8	1.95	1248.8	1068.8	1.86
20	2408.3	2295.5	3.98	1290.8	1144.8	2.50	1250.8	1068.5	2.36
21	3445.5	3323.5	6.99	1429.8	1281.0	3.25	1404.3	1237.8	3.14
22	9230.5	8436.5	12.77	3251.8	2873.8	5.20	3494.0	2979.8	5.35
23	17413.8	16018.5	29.99	3838.3	3440.5	8.00	3767.0	3255.5	7.69
24	23048.0	20999.0	51.75	4749.5	4301.8	11.71	4185.0	3707.5	10.21
25	56784.5	49131.3	69.77	9541.5	8463.3	14.42	7493.3	6522.3	11.06
26	142409.0	129826.8	220.09	18122.0	16399.8	35.04	10899.5	9612.3	20.36
27	148974.3	130906.3	291.37	19132.0	17318.0	46.19	10148.5	8963.5	22.82
28	1571.3	1524.8	2.28	1219.5	1088.5	1.86	1241.8	1056.0	1.75
29	2903.5	2755.5	5.74	1272.3	1130.8	2.84	1370.3	1163.8	2.96
30	3360.5	3208.0	6.96	1359.0	1223.3	3.27	1356.8	1178.5	3.05
31	7369.0	6800.0	10.01	3418.5	3003.0	5.47	3166.5	2718.0	5.05
32	20612.3	18987.5	36.24	3871.5	3488.5	7.94	3509.0	3049.3	6.74
33	24853.0	22808.0	48.72	4133.8	3740.0	9.53	3567.0	3164.3	7.90
34	68460.3	59711.3	83.84	9629.5	8558.8	15.47	8580.0	7398.0	12.89
35	164834.8	149315.3	277.48	18909.0	17113.0	40.25	8738.0	7701.0	17.31
36	179938.5	158597.0	370.88	16333.8	14804.3	41.31	8700.3	7746.3	20.76
37	1622.5	1577.0	2.30	1234.8	1103.3	1.93	1233.8	1061.8	1.84
38	3122.8	2994.3	5.56	1331.5	1182.0	2.76	1255.0	1089.8	2.55
39	3401.0	3184.0	7.07	1448.3	1295.8	3.49	1225.5	1063.3	2.82
40	9124.3	8251.8	12.11	3527.3	3107.5	5.39	2901.3	2495.0	4.33
41	24849.8	22809.5	42.15	4055.8	3632.8	8.25	3331.5	2910.0	6.52
42	25790.5	23405.5	49.07	4349.0	3943.8	9.93	3327.5	2956.8	7.44
43	67466.0	58745.8	78.26	9883.3	8791.3	14.01	8780.5	7599.0	12.19
44	178158.0	160234.5	288.10	16462.3	14895.3	32.82	9011.8	7980.0	16.93
45	180165.0	159855.0	360.46	13833.0	12531.3	34.58	8604.0	7670.3	20.51

Table 8 presents average results over the four starting points of the compared algorithms. We see that Algorithm 5.1 and EQ-VABBmin are usually much faster

than the Dai-Fletcher method. Moreover, Algorithm 5.1 costs smaller number of iterations and takes less CPU time than EQ-VABBmin for most of the problems.

5.2.2. Support vector machines. Support vector machines (SVMs) are popular models in machine learning, especially suitable for classification, which can be formulated as an SLB problem, see [8] for details. Suppose that we are given a training set of labeled examples

$$D = \{(z_i, w_i),\ i = 1, \ldots, n,\ z_i \in \mathbb{R}^m,\ w_i \in \{1, -1\}\}.$$

The SVM model employs some kernel function, say $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$, to classify new examples $z \in \mathbb{R}^m$ by a decision function $F : \mathbb{R}^m \to \{1, -1\}$ defined as

$$F(z) = \text{sign}\left(\sum_{i=1}^{n} x^* w_i K(z_i, z_j) + b^*\right),$$

where x^* solves

$$\begin{aligned} \min \quad & \frac{1}{2} x^\mathsf{T} G x - e^\mathsf{T} x \\ s.t. \quad & 0 \le x \le Ce \\ & w^\mathsf{T} x = 0. \end{aligned}$$

Here, G has entries $G_{ij} = w_i w_j K(z_i, z_j)$, $i, j = 1, \ldots, n$, C is a parameter of the SVM model, and e is the vector of all ones. The quantity $b^* \in \mathbb{R}$ is easily derived after x^* is computed.

Using the Gaussian kernel

$$K(z_i, z_j) = \exp\left(-\frac{\|z_i - z_j\|_2^2}{2\sigma^2}\right),$$

we have tested three real-world datasets for the binary classification: a9a, w8a, and ijcnn1, which can be downloaded from the LIBSVM website[2]. For each dataset, we randomly chose 1000 examples to generate the test problem. The parameters C and σ were set to 1 and $\sqrt{10}$, respectively.

Three different tolerance values were tested: $\epsilon = 10^{-3}, 10^{-6}, 10^{-9}$. The null vector was employed as the initial point for all the compared methods. Table 9 presents the required number of iterations and CPU time in seconds costed by the compared methods for different tolerance requirements. It can be seen from Table 9 that Algorithm 5.1 often performs better than the other two methods.

6. Conclusion and discussion. We have introduced a mechanism for the gradient method to achieve the two dimensional quadratic termination. Based on the mechanism, we derived a novel stepsize α_k^{new} (see the formula (2.15)) such that the Barzilai-Borwein (BB) method enjoys the two dimensional quadratic termination by equipping with α_k^{new}. This novel stepsize only makes use of the BB stepsizes in previous iterations and thus can easily be adopted to general unconstrained and constrained optimization. We developed a new efficient gradient method (see the method (3.1)) that adaptively takes long BB steps and short steps associated with α_k^{new} for unconstrained quadratic optimization. Then based on the method (3.1) and two

[2] www.csie.ntu.edu.tw/~cjlin/libsvmtools/

TABLE 9
Results of Algorithm 5.1, Dai-Fletcher and EQ-VABBmin on SVM problems with $n = 1000$.

methods	10^{-3}		10^{-6}		10^{-9}	
	iter	time	iter	time	iter	time
	a9a					
Dai-Fletcher	253	0.70	537	1.61	687	2.22
EQ-VABBmin	132	0.46	300	0.86	453	1.30
Algorithm 5.1	121	0.38	253	0.84	468	1.54
	w8a					
Dai-Fletcher	202	0.55	782	2.21	1028	2.73
EQ-VABBmin	107	0.32	454	1.33	679	1.92
Algorithm 5.1	78	0.24	284	0.81	550	1.59
	ijcnn1					
Dai-Fletcher	293	0.81	41784	145.69	84698	290.48
EQ-VABBmin	123	0.39	23928	82.99	29647	90.57
Algorithm 5.1	63	0.17	6487	20.44	21238	65.22

nonmonotone line searches, we were able to design efficient gradient algorithms for unconstrained optimization problems and extreme eigenvalues problems, see Algorithms 3.1 and 4.1. By incorporating the gradient projection technique and taking the constraints into consideration, we developed an efficient projected gradient algorithm, Algorithm 5.1, for both box-constrained optimization and singly linearly box-constrained (SLB) optimization problems. Our numerical experiments demonstrated the efficiency of these algorithms over many successful numerical methods in the literature.

The results achieved in this paper further emphasize the importance of the two dimensional quadratic optimization model in the analysis of gradient methods for optimization. We are wondering whether higher-dimensional quadratic optimization models will be helpful in the construction of more efficient gradient methods. This is an interesting issue worthwhile investigation.

Appendix. Additional experimental results. Tables 10, 11 and 12 present the results of Algorithm 3.1, GBB and ABBmin on 147 unconstrained problems from the CUTEst collection. It can be seen that Algorithm 3.1 performs better than GBB and ABBmin on most problems.

Acknowledgments. The authors would like to thank the associate editor and the anonymous referees for their valuable comments and suggestions.

REFERENCES

[1] P. ABSIL, R. MAHONY, R. SEPULCHRE, AND P. VAN DOOREN, *A Grassmann-Rayleigh quotient iteration for computing invariant subspaces*, SIAM Rev., 44 (2002), pp. 57–73.
[2] H. AKAIKE, *On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method*, Ann. Inst. Stat. Math., 11 (1959), pp. 1–16.
[3] J. BARZILAI AND J. M. BORWEIN, *Two-point step size gradient methods*, IMA J. Numer. Anal., 8 (1988), pp. 141–148.
[4] E. G. BIRGIN, J. M. MARTÍNEZ, AND M. RAYDAN, *Nonmonotone spectral projected gradient methods on convex sets*, SIAM J. Optim., 10 (2000), pp. 1196–1211.
[5] E. G. BIRGIN, J. M. MARTÍNEZ, AND M. RAYDAN, *Spectral projected gradient methods: review and perspectives*, J. Stat. Softw., 60 (2014), pp. 539–559.
[6] S. BONETTINI, R. ZANELLA, AND L. ZANNI, *A scaled gradient projection method for constrained image deblurring*, Inverse Probl., 25 (2008), 015002.
[7] A. CAUCHY, *Méthode générale pour la résolution des systemes di'équations simultanées*, Comp. Rend. Sci. Paris, 25 (1847), pp. 536–538.

TABLE 10

Results of Algorithm 3.1, GBB and ABBmin on 147 unconstrained problems from the CUTEst collection.

problem	n	GBB			ABBmin			Algorithm 3.1		
		nfe	iter	time	nfe	iter	time	nfe	iter	time
AKIVA	2	66	49	0.18	32	25	0.01	34	25	0.01
ALLINITU	4	16	16	0.01	18	18	0.00	16	16	0.01
ARGLINA	200	3	3	0.01	3	3	0.01	3	3	0.00
ARGTRIGLS	200	9469	5550	3.30	2083	2018	1.14	1380	1201	0.65
ARWHEAD	5000	4	4	0.02	4	4	0.01	4	4	0.02
BA-L1LS	57	43	32	0.01	43	32	0.00	43	32	0.00
BARD	3	284	174	0.01	171	165	0.02	85	73	0.02
BDQRTIC	5000	94	91	0.14	86	83	0.10	74	73	0.09
BEALE	2	57	52	0.00	37	35	0.00	40	40	0.00
BIGGS6	6	1691	995	0.05	899	839	0.04	464	410	0.03
BOX	10000	76	31	0.16	40	23	0.08	53	26	0.10
BOX3	3	42	38	0.00	47	40	0.00	38	31	0.00
BOXBODLS	2	8	5	0.00	28	5	0.00	9	8	0.01
BRKMCC	2	11	10	0.00	10	9	0.00	10	9	0.00
BROWNDEN	4	61	54	0.01	61	59	0.01	69	65	0.00
BROYDN3DLS	5000	127	119	0.19	85	84	0.08	90	88	0.09
BROYDN7D	5000	3629	2612	10.01	7179	2966	13.13	3209	2548	9.67
BROYDNBDLS	5000	95	88	0.20	45	43	0.09	50	49	0.11
BRYBND	5000	95	88	0.20	45	43	0.10	50	49	0.11
CHAINWOO	4000	86776	52782	81.00	87547	55599	81.21	644	560	0.76
CHNROSNB	50	2210	1401	0.08	1198	851	0.04	1039	883	0.04
CHNRSNBM	50	2830	1818	0.09	1808	1320	0.05	1305	1103	0.05
CHWIRUT2LS	3	1441	721	0.06	171	157	0.01	86	67	0.01
CLIFF	2	1369	433	0.03	117	38	0.01	115	36	0.00
CRAGGLVY	5000	217	176	0.47	168	157	0.74	157	150	0.40
CUBE	2	138	107	0.01	282	269	0.01	152	136	0.00
DECONVU	63	98988	56903	3.56	5839	5796	0.26	4431	3894	0.18
DENSCHNA	2	15	15	0.00	15	15	0.00	15	15	0.00
DENSCHNB	2	9	9	0.00	9	9	0.00	9	9	0.00
DENSCHNC	2	7	7	0.00	7	7	0.00	7	7	0.00
DENSCHND	3	184	127	0.01	126	125	0.01	128	121	0.01
DENSCHNE	3	100	72	0.00	99	40	0.00	71	49	0.00
DENSCHNF	2	29	19	0.00	71	21	0.00	18	18	0.00
DIXMAANA	3000	8	8	0.01	8	8	0.01	8	8	0.01
DIXMAANB	3000	8	8	0.02	8	8	0.01	8	8	0.02
DIXMAANC	3000	9	9	0.02	9	9	0.01	9	9	0.01
DIXMAAND	3000	10	10	0.02	10	10	0.02	10	10	0.01
DIXMAANE	3000	790	526	0.36	350	334	0.23	411	359	0.25
DIXMAANF	3000	901	580	0.44	285	273	0.35	308	277	0.30
DIXMAANG	3000	803	516	0.38	368	354	0.24	394	347	0.24
DIXMAANH	3000	983	653	0.50	299	291	0.22	316	282	0.21
DIXMAANI	3000	21288	12078	10.56	2811	2746	1.87	2581	2225	1.54
DIXMAANJ	3000	1160	728	0.58	321	312	0.23	330	292	0.21
DIXMAANK	3000	744	462	0.36	307	294	0.21	320	289	0.21
DIXMAANL	3000	734	463	0.39	253	244	0.17	260	237	0.18
DIXMAANM	3000	26329	14966	12.82	2995	2921	1.94	3105	2711	2.05
DIXMAANN	3000	3254	1962	1.55	656	641	0.45	904	785	0.58
DIXMAANO	3000	1636	1007	0.80	718	704	0.50	713	637	0.50
DIXMAANP	3000	3194	1875	1.48	752	728	0.49	748	667	0.46
DIXON3DQ	10000	82173	46678	104.65	7475	7404	14.72	6405	5487	11.10
DJTL	2	719	212	0.02	735	207	0.02	393	191	0.02
DQDRTIC	5000	29	29	0.04	14	14	0.03	15	15	0.04
DQRTIC	5000	50	50	0.05	50	50	0.03	50	50	0.04
ECKERLE4LS	3	13	5	0.00	13	5	0.00	13	5	0.00
EDENSCH	2000	40	40	0.01	42	42	0.03	38	38	0.02
EG2	1000	6	6	0.00	6	6	0.00	6	6	0.00
ENGVAL1	5000	36	30	0.05	28	25	0.08	28	25	0.04
ENGVAL2	3	161	102	0.01	273	241	0.02	106	93	0.01
ENSOLS	9	106	86	0.02	97	90	0.02	106	96	0.02
ERRINROS	50	100550	58958	2.45	26974	26736	0.97	2585	2313	0.09
EXPFIT	2	36	32	0.00	31	31	0.00	38	34	0.00
FBRAINLS	2	54	26	0.05	24	17	0.03	27	20	0.04
FLETBV3M	5000	66	65	0.19	206	105	0.48	59	58	0.22
FLETCBV2	5000	1	0	0.00	1	0	0.00	1	0	0.00
FLETCHCR	1000	1422	899	0.16	408	396	0.06	299	270	0.04
FMINSRF2	5625	1343	923	1.75	830	819	1.59	661	604	1.14
FMINSURF	5625	2288	1496	3.02	1007	992	1.90	935	846	1.61

TABLE 11

Results of Algorithm 3.1, GBB and ABBmin on 147 unconstrained problems from the CUTEst collection (continued).

problem	n	GBB			ABBmin			Algorithm 3.1		
		nfe	iter	time	nfe	iter	time	nfe	iter	time
GBRAINLS	2	73	33	0.07	25	17	0.04	25	17	0.03
GENHUMPS	5000	20699	14088	53.06	14273	5970	19.32	11438	9258	32.86
GENROSE	500	6269	3977	0.46	7295	3669	0.38	3918	3028	0.26
GROWTHLS	3	2	2	0.00	2	2	0.00	2	2	0.00
GULF	3	5893	2269	0.48	223	203	0.03	444	348	0.05
HAIRY	2	46	34	0.00	209	63	0.00	48	44	0.00
HATFLDD	3	203	100	0.00	194	149	0.01	50	37	0.00
HATFLDE	3	155	90	0.01	231	173	0.01	214	151	0.01
HATFLDFL	3	596	289	0.01	364	330	0.02	203	169	0.01
HEART8LS	8	866	458	0.02	1476	1126	0.05	943	694	0.03
HELIX	3	73	62	0.00	76	73	0.01	69	64	0.01
HIELOW	3	213	121	0.23	127	21	0.05	67	58	0.09
HILBERTA	2	8	8	0.00	8	8	0.00	8	8	0.00
HILBERTB	10	8	8	0.00	8	8	0.00	8	8	0.00
HIMMELBB	2	2	2	0.00	2	2	0.00	2	2	0.00
HIMMELBF	4	108383	51814	2.16	3560	3105	0.10	791	706	0.03
HIMMELBG	2	12	11	0.00	9	9	0.00	12	11	0.00
HIMMELBH	2	14	13	0.00	14	13	0.00	14	13	0.00
HUMPS	2	668	418	0.02	278	112	0.01	255	234	0.02
INTEQNELS	12	9	8	0.00	9	8	0.00	9	8	0.00
JENSMP	2	54	33	0.00	108	38	0.00	31	28	0.00
KOWOSB	4	320	208	0.01	320	294	0.02	178	139	0.01
LANCZOS1LS	6	9194	5256	0.26	4869	4748	0.19	2520	2330	0.10
LANCZOS2LS	6	9391	5368	0.33	4555	4426	0.24	3529	3220	0.15
LANCZOS3LS	6	8262	4713	0.24	5077	4944	0.20	4743	4387	0.18
LIARWHD	5000	112	61	0.10	43	43	0.08	51	48	0.07
LOGHAIRY	2	108	93	0.00	670	172	0.01	304	294	0.01
LSC1LS	3	189	163	0.01	207	91	0.00	830	582	0.04
LUKSAN11LS	100	6474	4150	0.21	11842	7647	0.35	5069	4080	0.17
LUKSAN13LS	98	390	288	0.02	245	203	0.02	232	204	0.01
LUKSAN14LS	98	364	262	0.02	196	187	0.01	193	177	0.01
LUKSAN15LS	100	34	32	0.01	34	32	0.02	34	32	0.02
LUKSAN16LS	100	41	38	0.00	38	35	0.00	41	38	0.01
LUKSAN17LS	100	1706	1084	0.34	526	504	0.11	518	473	0.10
LUKSAN21LS	100	3706	2220	0.12	1328	1239	0.05	2010	1638	0.08
MANCINO	100	15	15	0.08	15	15	0.06	15	15	0.06
MARATOSB	2	5268	1152	0.08	4797	2290	0.09	3038	1474	0.07
MEXHAT	2	479	80	0.01	52	45	0.00	49	37	0.00
MGH09LS	4	151	91	0.00	251	206	0.01	103	92	0.00
MNISTS0LS	494	2	2	0.32	2	2	0.25	2	2	0.15
MNISTS5LS	494	2	2	0.50	2	2	0.15	2	2	0.13
MOREBV	5000	148	98	0.09	115	98	0.10	119	95	0.12
MSQRTBLS	1024	32839	19067	23.57	4002	3917	4.24	2915	2576	2.99
NCB20	5010	34412	19916	92.55	2456	1940	8.52	510	453	2.03
NCB20B	5000	10708	6308	28.66	5418	5185	22.00	5316	2989	13.63
NONDIA	5000	13	11	0.01	30	27	0.03	25	25	0.04
NONDQUAR	5000	10390	5979	5.35	3999	3943	2.79	2863	2424	1.79
OSBORNEB	11	2850	1734	0.12	270	150	0.03	872	757	0.06
PALMER5C	6	28	28	0.01	26	26	0.00	28	28	0.00
PENALTY1	1000	19	18	0.01	653	176	0.04	45	45	0.01
POWELLSG	5000	500	302	0.28	226	196	0.14	252	198	0.14
POWER	10000	7540	1449	3.45	886	743	1.03	895	713	0.96
QUARTC	5000	50	50	0.03	50	50	0.04	50	50	0.03
RAT42LS	3	2	2	0.00	2	2	0.00	2	2	0.00
RAT43LS	4	4	4	0.00	4	4	0.00	4	4	0.00
ROSENBR	2	98	68	0.00	149	96	0.01	116	103	0.01
ROSENBRTU	2	436	213	0.01	1348	268	0.03	192	125	0.01
S308	2	14	14	0.00	14	14	0.00	14	14	0.00
SCHMVETT	5000	80	80	0.36	67	66	0.16	67	67	0.18
SENSORS	100	69	60	0.46	60	47	0.32	26	26	0.10
SINEVAL	2	269	101	0.01	353	271	0.02	237	168	0.01
SINQUAD	5000	214	125	0.34	227	124	0.34	85	76	0.19
SISSER	2	13	13	0.00	13	13	0.00	13	13	0.00
SNAIL	2	15	15	0.00	408	103	0.01	9	9	0.00

TABLE 12
Results of Algorithm 3.1, GBB and ABBmin on 147 unconstrained problems from the CUTEst collection (continued).

problem	n	GBB			ABBmin			Algorithm 3.1		
		nfe	iter	time	nfe	iter	time	nfe	iter	time
SPARSINE	5000	200132	113380	254.60	10564	10418	20.86	11523	9961	20.34
SPARSQUR	10000	32	32	0.22	32	32	0.20	32	32	0.18
SPMSRTLS	4999	769	503	0.97	395	354	0.57	757	667	1.09
SROSENBR	5000	25	21	0.03	19	18	0.02	18	17	0.03
TESTQUAD	5000	207683	117247	57.84	8469	8291	3.27	7714	6788	2.79
TOINTGOR	50	323	240	0.01	195	186	0.02	194	186	0.01
TOINTGSS	5000	2	2	0.01	2	2	0.00	2	2	0.00
TOINTPSP	50	434	312	0.03	192	184	0.02	209	189	0.01
TOINTQOR	50	58	58	0.00	53	53	0.00	49	49	0.00
TQUARTIC	5000	2361	717	0.91	43	27	0.03	50	27	0.04
TRIDIA	5000	19589	11351	7.66	1875	1821	1.04	2659	2323	1.38
VARDIM	200	12732	325	0.19	1924	192	0.04	1985	204	0.03
VAREIGVL	50	28	28	0.00	27	27	0.00	28	28	0.00
WATSON	12	9343	5460	0.35	3454	3368	0.22	1617	1412	0.06
WOODS	4000	890	523	0.50	691	434	0.36	268	235	0.18
ZANGWIL2	2	3	2	0.00	3	2	0.00	3	2	0.01

[8] C. CORTES AND V. VAPNIK, *Support-vector networks*, Mach. Learn., 20 (1995), pp. 273–297.
[9] S. CRISCI, V. RUGGIERO, AND L. ZANNI, *Steplength selection in gradient projection methods for box-constrained quadratic programs*, Appl. Math. Comput., 356 (2019), pp. 312–327.
[10] S. CRISCI, F. PORTA, V. RUGGIERO, AND L. ZANNI, *Spectral properties of Barzilai-Borwein rules in solving singly linearly constrained optimization problems subject to lower and upper bounds*, SIAM J. Optim., 30 (2020), pp. 1300–1326.
[11] Y.-H. DAI, *Alternate step gradient method*, Optimization, 52 (2003), pp. 395–415.
[12] Y.-H. DAI AND R. FLETCHER, *Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming*, Numer. Math., 100 (2005), pp. 21–47.
[13] Y.-H. DAI AND R. FLETCHER, *New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds*, Math. Program., 106 (2006), pp. 403–421.
[14] Y.-H. DAI, W. W. HAGER, K. SCHITTKOWSKI, AND H. ZHANG, *The cyclic Barzilai-Borwein method for unconstrained optimization*, IMA J. Numer. Anal., 26 (2006), pp. 604–627.
[15] Y.-H. DAI, Y. HUANG, AND X.-W. LIU, *A family of spectral gradient methods for optimization*, Comput. Optim. Appl., 74 (2019), pp. 43–65.
[16] Y.-H. DAI AND L.-Z. LIAO, *R-linear convergence of the Barzilai and Borwein gradient method*, IMA J. Numer. Anal., 22 (2002), pp. 1–10.
[17] Y.-H. DAI AND Y.-X. YUAN, *Analysis of monotone gradient methods*, J. Ind. Manag. Optim., 1 (2005), pp. 181–192.
[18] Y.-H. DAI AND H. ZHANG, *Adaptive two-point stepsize gradient algorithm*, Numer. Algorithms, 27 (2001), pp. 377–385.
[19] A. DASPREMONT, L. E. GHAOUI, M. I. JORDAN, AND G. R. G. LANCKRIET, *A direct formulation for sparse PCA using semidefinite programming*, SIAM Rev., 49 (2007), pp. 434–448.
[20] R. DE ASMUNDIS, D. DI SERAFINO, W. W. HAGER, G. TORALDO, AND H. ZHANG, *An efficient gradient method using the Yuan steplength*, Comp. Optim. Appl., 59 (2014), pp. 541–563.
[21] D. DI SERAFINO, V. RUGGIERO, G. TORALDO, AND L. ZANNI, *On the steplength selection in gradient methods for unconstrained optimization*, Appl. Math. Comput., 318 (2018), pp. 176–195.
[22] E. D. DOLAN AND J. J. MORÉ, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[23] R. FLETCHER, *On the Barzilai-Borwein method*, in: Optimization and Control with Applications, Volume 96, L. Qi, K. Teo and X. Yang, eds., Springer, Boston, 2005, pp. 235–256.
[24] G. E. FORSYTHE, *On the asymptotic directions of the s-dimensional optimum gradient method*, Numer. Math., 11 (1968), pp. 57–76.
[25] G. FRASSOLDATI, L. ZANNI, AND G. ZANGHIRATI, *New adaptive stepsize selections in gradient methods*, J. Ind. Manag. Optim., 4 (2008), pp. 299–312.
[26] A. FRIEDLANDER, J. M. MARTÍNEZ, B. MOLINA, AND M. RAYDAN, *Gradient method with retards and generalizations*, SIAM J. Numer. Anal., 36 (1998), pp. 275–289.
[27] N. I. GOULD, D. ORBAN, AND P. L. TOINT, *CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization*, Comp. Optim. Appl., 60 (2015), pp. 545–557.

[28] L. Grippo, F. Lampariello, and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM J. Numer. Anal., 23 (1986), pp. 707–716.
[29] J. Hu, B. Jiang, L. Lin, Z. Wen, and Y. Yuan, *Structured quasi-Newton methods for optimization with orthogonality constraints*, SIAM J. Sci. Comput., 41 (2019), pp. 2239–2269.
[30] Y. Huang, Y.-H. Dai, X.-W. Liu, and H. Zhang, *On the asymptotic convergence and acceleration of gradient methods*, preprint, arXiv:1908.07111v1 [math.OC], (2019).
[31] Y. Huang, Y.-H. Dai, X.-W. Liu, and H. Zhang, *Gradient methods exploiting spectral properties*, Optim. Method Softw., 35 (2020), pp. 681–705.
[32] Y. Huang, Y.-H. Dai, X.-W. Liu, and H. Zhang, *On the acceleration of the Barzilai-Borwein method*, preprint, arXiv:2001.02335v1 [math.OC], (2020).
[33] Y. Huang and H. Liu, *On the rate of convergence of projected Barzilai-Borwein methods*, Optim. Method Softw., 30 (2015), pp. 880–892.
[34] B. Jiang, C. Cui, and Y.-H. Dai, *Unconstrained optimization models for computing several extreme eigenpairs of real symmetric matrices*, Pac. J. Optim., 10 (2014), pp. 53–71.
[35] D.-W. Li and R.-Y Sun, *On a faster R-linear convergence rate of the Barzilai-Borwein method*, preprint, arXiv:2101.00205v1 [math.OC], (2021).
[36] J. Moré and G. Toraldo, *Algorithms for bound constrained quadratic programming problems*, Numer. Math., 55 (1989), pp. 377–400.
[37] M. Raydan, *On the Barzilai and Borwein choice of steplength for the gradient method*, IMA J. Numer. Anal., 13 (1993), pp. 321–326.
[38] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim., 7 (1997), pp. 26–33.
[39] M. Raydan and B. F. Svaiter, *Relaxed steepest descent and Cauchy-Barzilai-Borwein method*, Comp. Optim. Appl., 21 (2002), pp. 155–167.
[40] A. Sameh and Z. Tong, *The trace minimization method for the symmetric generalized eigenvalue problem*, J. Comput. Appl. Math., 123 (2000), pp. 155–175.
[41] A. Sameh and J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal., 19 (1982), pp. 1243–1259.
[42] Y.-X. Yuan, *A new stepsize for the steepest descent method*, J. Comput. Math., 24 (2006), pp. 149–156.
[43] Y.-X. Yuan, *Step-sizes for the gradient method*, AMS IP Studies in Advanced Mathematics, 42 (2008), pp. 785–796.
[44] H. Zhang and W. W. Hager, *A nonmonotone line search technique and its application to unconstrained optimization*, SIAM J. Optim., 14 (2004), pp. 1043–1056.
[45] B. Zhou, L. Gao, and Y.-H. Dai, *Gradient methods with adaptive step-sizes*, Comp. Optim. Appl., 35 (2006), pp. 69–86.