ARTICLE

Gradient methods exploiting spectral properties

Yakui Huang[a], Yu-Hong Dai[b*], Xin-Wei Liu,[a] and Hongchao Zhang[c]

[a]Institute of Mathematics, Hebei University of Technology, Tianjin, China; [b]LSEC, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, China; [c]Department of Mathematics, Louisiana State University, Baton Rouge, LA 70803-4918, USA

ARTICLE HISTORY
Compiled May 9, 2019

ABSTRACT
We propose a new stepsize for the gradient method. It is shown that this new stepsize will converge to the reciprocal of the largest eigenvalue of the Hessian, when Dai-Yang's asymptotic optimal gradient method (Computational Optimization and Applications, 2006, 33(1): 73-88) is applied for minimizing quadratic objective functions. Based on this spectral property, we develop a monotone gradient method that takes a certain number of steps using the asymptotically optimal stepsize by Dai and Yang, and then follows by some short steps associated with this new stepsize. By employing one step retard of the asymptotic optimal stepsize, a nonmonotone variant of this method is also proposed. Under mild conditions, R-linear convergence of the proposed methods is established for minimizing quadratic functions. In addition, by combining gradient projection techniques and adaptive nonmonotone line search, we further extend those methods for general bound constrained optimization. Two variants of gradient projection methods combining with the Barzilai-Borwein stepsizes are also proposed. Our numerical experiments on both quadratic and bound constrained optimization indicate that the new proposed strategies and methods are very effective.

KEYWORDS
gradient method; spectral property; Barizilai-Borwein methods; linear convergence; quadratic optimization; bound constrained optimization

AMS CLASSIFICATION
90C20; 90C25; 90C30

1. Introduction

We consider the problem of minimizing a convex quadratic function

$$\min \quad f(x) = \frac{1}{2}x^T Ax - b^T x \tag{1}$$

and its extensions on bound constrained optimization, where $b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite with eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ and condition number $\kappa = \frac{\lambda_n}{\lambda_1}$. This problem (1) is one of the simplest non-trivial non-linear programming problems and efficiently solving (1) is usually a pre-requisite for a method to be

*Corresponding author. Email: dyh@lsec.cc.ac.cn

generalized for solving more general optimization. In addition, various optimization problems arising in many applications including machine learning [6], sparse reconstruction [20], nonnegative matrix factorization [29, 31] can be formulated as the form of (1), possibly with the addition of regularization or bound constraints.

The simplest and easily implemented method for solving (1) is the gradient method, which updates the iterates by

$$x_{k+1} = x_k - \alpha_k g_k, \tag{2}$$

where $g_k = \nabla f(x_k)$ and $\alpha_k > 0$ is the stepsize determined by different strategies.

The classic steepest descent (SD) method for solving (1) can be dated back to Cauchy [5], who suggested to compute the stepsize by exact line search:

$$\alpha_k^{SD} = \arg\min_{\alpha \in \mathbb{R}} \ f(x_k - \alpha g_k) = \frac{g_k^T g_k}{g_k^T A g_k}. \tag{3}$$

It has been shown that the method converges linearly [1] with Q-linear rate $\frac{\kappa-1}{\kappa+1}$. Thus, the SD method can be very slow especially when the condition number is large. Further analysis shows that the gradients will asymptotically perform zigzag between two orthogonal directions in the subspace spanned by the two eigenvectors corresponding to λ_1 and λ_n, see [22, 33] for more details.

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

$$\alpha_k^{BB1} = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}, \quad \text{and} \quad \alpha_k^{BB2} = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}, \tag{4}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. The BB method was shown to be globally convergent for minimizing general n-dimensional strictly convex quadratics [34] with R-linear convergence rate [11]. Recently, the BB method and its variants have been successfully extended to general unconstrained problems [35], to constrained optimization problems [3, 28] and to various applications [29, 30, 32]. One may see [4, 10, 17, 21, 37] and the references therein.

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

$$g_k = \sum_{i=1}^{n} \mu_i^k \xi_i.$$

The above decomposition of gradient g_k together with the update rule (2) give

$$g_{k+1} = g_k - \alpha_k A g_k = \prod_{j=1}^{k} (I - \alpha_j A) g_1 = \sum_{i=1}^{n} \mu_i^{k+1} \xi_i,$$

where

$$\mu_i^{k+1} = \mu_i^k(1-\alpha_k\lambda_i) = \mu_i^1\prod_{j=1}^{k}(1-\alpha_j\lambda_i).$$

This relation implies that the closer α_k to $\frac{1}{\lambda_i}$, the smaller $|\mu_i^{k+1}|$ would be. In addition, if $\mu_i^k = 0$, the corresponding component will vanish at all subsequent iterations.

Since the SD method will asymptotically zigzag between ξ_1 and ξ_n, a natural way to break the zigzagging pattern is to eliminate the component μ_1^k or μ_n^k, which can be achieved by employing a stepsize approximating $\frac{1}{\lambda_1}$ or $\frac{1}{\lambda_n}$. One seminal work in this line of research is due to Yuan [14, 36], who derived the following stepsize by imposing finite termination for minimizing two-dimensional convex quadratics:

$$\alpha_k^Y = \frac{2}{\sqrt{(1/\alpha_{k-1}^{SD}-1/\alpha_k^{SD})^2+4\|g_k\|^2/(\alpha_{k-1}^{SD}\|g_{k-1}\|)^2}+(1/\alpha_{k-1}^{SD}+1/\alpha_k^{SD})}. \tag{5}$$

Based on (5), Dai and Yuan [14] further suggested a new gradient method whose stepsize is given by

$$\alpha_k^{DY} = \begin{cases} \alpha_k^{SD}, & \text{if mod}(k,4)< 2; \\ \alpha_k^Y, & \text{otherwise.} \end{cases} \tag{6}$$

The DY method (6) keeps monotonicity and appears better than the nonmonotone BB method [14]. It is shown by De Asmundis et al. [16] that the stepsize α_k^Y converges to $\frac{1}{\lambda_n}$ if the SD method is applied to solve problem (1). That is, occasionally employing the stepsize α_k^Y along the SD method will enhance the elimination of the component μ_n. Recently, Gonzaga and Schneider [25] suggest a monotone method with all stepsizes of the form (3). Their method approximates $\frac{1}{\lambda_n}$ by a short stepsize calculated by replacing g_k in (3) with $\tilde{g} = (I-\eta A)g_k$ for some large scalar η.

Nonmonotone gradient methods exploiting spectral properties have been developed as well. Frassoldati et al. [23] developed a new short stepsize by maximizing the next SD stepsize α_{k+1}^{SD}. They further suggested a method, called $\text{ABB}_{\min 2}$, which tries to enforce BB1 stepsizes close to $\frac{1}{\lambda_1}$ by using short stepsizes to eliminate gradient components associated with large eigenvalues. More recently, based on the favorable property of α_k^Y, De Asmundis et al. [16] suggested to reuse it in a cyclic fashion after a certain number of SD steps. Precisely, their approach, referred to as the SDC method, employs the stepsize

$$\alpha_k^{SDC} = \begin{cases} \alpha_k^{SD}, & \text{if mod}(k, h+s) < h; \\ \alpha_t^Y, & \text{otherwise, with } t = \max\{i \le k : \text{mod}(i, h+s) = h\}, \end{cases} \tag{7}$$

where $h \ge 2$ and $s \ge 1$. They also proposed a monotone version of (7) by imposing safeguards on the stepsizes.

One common character of the aforementioned methods is making use of spectral properties of the stepsizes. The recent study [17] points out that gradient methods using long and short stepsizes that attempt to exploit the spectral properties have generally better numerical performance for minimizing both quadratic and general nonlinear objective functions. For more works on gradient methods, see [7, 9, 16, 17,

23, 25, 37, 39]. In [12], Dai and Yang introduced a gradient method with a new stepsize that possesses similar spectral property as α_k^Y. More specifically, their stepsize is given by

$$\alpha_k^{AOPT} = \frac{\|g_k\|}{\|Ag_k\|}, \tag{8}$$

which asymptotically converges to $\frac{2}{\lambda_1+\lambda_n}$, that is in some sense an optimal stepsize since it minimizes $\|I-\alpha A\|$ over the stepsize α [12, 19]. Since $\alpha_k^{AOPT} \leq \alpha_k^{SD}$, the Dai-Yang method (8) is monotone but without using exact line searches. In addition, the method converges Q-linearly with the same rate as the SD method. More importantly, by applying this method it is possible to recover the eigenvectors ξ_1 and ξ_n.

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

$$\bar{\alpha}_k = \frac{d_k^T d_k}{d_k^T A d_k}, \tag{9}$$

where

$$d_k = \frac{g_{k-1}}{\|g_{k-1}\|} - \frac{g_k}{\|g_k\|}. \tag{10}$$

We show that the stepsize $\bar{\alpha}_k$ asymptotically converges to $\frac{1}{\lambda_n}$ if the Dai-Yang method (8) is applied to problem (1). Therefore, the stepsize $\bar{\alpha}_k$ is helpful in eliminating the gradient component μ_n. Thanks to this desired property, we are able to develop a new efficient gradient method by taking a certain number of steps using the asymptotically optimal stepsize α_k^{AOPT} followed by some short steps, which are determined by the smaller one of α_k^{AOPT} and $\bar{\alpha}_{k-1}$. Thus, this method is a monotone method without using exact line searches. We also construct a nonmonotone variant of the method which simply use the stepsize α_k^{AOPT} with one step retard. R-linear convergence of the proposed methods is established for minimizing strongly convex quadratic functions. In addition, by combining gradient projection techniques and the adaptive nonmonotone line search in [15], we further to extend those proposed methods for general bound constrained optimization. Two variants of gradient projection methods combining with the BB stepsizes are also proposed. Our numerical comparisons with DY (6), $\text{ABB}_{\min 2}$ [23] and SDC (7) methods on minimizing quadratic functions indicate the proposed strategies and methods are very effective. Moreover, our numerical comparisons with the spectral projected gradient (SPG) method [3, 4] on solving bound constrained optimization problems from the CUTEst collection [26] also highly suggest the potential benefits of extending the strategies and methods in the paper for more general large-scale bound constrained optimization.

The paper is organized as follows. In Section 2, we analyze the asymptotic spectral property of the new stepsize $\bar{\alpha}_k$ and propose our new methods based on this spectral property. In Section 3, we show that the new proposed methods have R-linear convergence for minimizing strongly convex quadratic functions. We generalize the proposed ideas and methods for bound constrained optimization in Section 4. Some numerical comparisons of our new methods on solving both quadratic and bound constrained optimization problems are shown in Section 5. Finally, in Section 6 we give some concluding remarks.

2. Method for quadratics

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

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

We first recall some important properties of the Dai-Yang method (8).

Lemma 2.1. *[12] For any starting point x_1 satisfying*

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

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

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

Furthermore,

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

and

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

which indicates that

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

and

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

where

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

Lemma 2.1 indicates that the method (8) asymptotically conducts its searches in the two-dimensional subspace spanned by ξ_1 and ξ_n. So, in order to accelerate the convergence, we could employ some stepsizes approximating $\frac{1}{\lambda_1}$ or $\frac{1}{\lambda_n}$ to eliminate the component μ_1^k or μ_n^k. Note that the vector d_k given in (10) tends to align in the direction of the eigenvector ξ_n. Hence, if we take some consecutive gradient steps with stepsize α_k^{AOPT} so that $d_k \approx \pm 2\sqrt{c_2}\xi_n$, the stepsize $\bar{\alpha}_k$ will be an approximation of $\frac{1}{\lambda_n}$. The next theorem provides theoretical justification for this strategy.

Theorem 2.2. *Under the conditions in Lemma 2.1, let* $\{g_k\}$ *be the sequence generated by applying the method* (8) *to problem* (1)*. Then we have*

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

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

$$d_k^T d_k = 2 - 2\frac{g_{k-1}^T g_k}{\|g_{k-1}\|\|g_k\|} \tag{11}$$

and

$$d_k^T A d_k = \frac{g_{k-1}^T A g_{k-1}}{\|g_{k-1}\|^2} + \frac{g_k^T A g_k}{\|g_k\|^2} - 2\frac{g_{k-1}^T A g_k}{\|g_{k-1}\|\|g_k\|}, \tag{12}$$

which indicate that

$$\begin{aligned}\lim_{k\to\infty} d_k^T d_k &= 2 - 2\lim_{k\to\infty}\sum_{j=1}^{n} \frac{\mu_i^{2k-1}}{\sqrt{\sum_{j=1}^n (\mu_j^{2k-1})^2}} \frac{\mu_i^{2k}}{\sqrt{\sum_{j=1}^n (\mu_j^{2k})^2}} \\ &= 2 - 2(c_1 - c_2)\end{aligned} \tag{13}$$

and

$$\begin{aligned}\lim_{k\to\infty} d_k^T A d_k &= \lim_{k\to\infty} \frac{\sum_{j=1}^n \lambda_i (\mu_i^{2k-1})^2}{\sum_{j=1}^n (\mu_j^{2k-1})^2} + \frac{\sum_{j=1}^n \lambda_i (\mu_i^{2k})^2}{\sum_{j=1}^n (\mu_j^{2k})^2} \\ &\quad - 2\sum_{j=1}^{n} \lambda_i \frac{\mu_i^{2k-1}}{\sqrt{\sum_{j=1}^n (\mu_j^{2k-1})^2}} \frac{\mu_i^{2k}}{\sqrt{\sum_{j=1}^n (\mu_j^{2k})^2}} \\ &= 2(\lambda_1 c_1 + \lambda_n c_2) - 2(\lambda_1 c_1 - \lambda_n c_2) \\ &= 4\lambda_n c_2.\end{aligned} \tag{14}$$

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

$$\begin{aligned}\lim_{k\to\infty} \bar{\alpha}_k &= \lim_{k\to\infty} \frac{d_k^T d_k}{d_k^T A d_k} = \frac{2 - 2(c_1 - c_2)}{4\lambda_n c_2} \\ &= \frac{4(\lambda_1 + \lambda_n) - (2\lambda_n - 2\lambda_1)}{2\lambda_n(3\lambda_1 + \lambda_n)} = \frac{1}{\lambda_n}.\end{aligned}$$

This completes the proof. □

Using the same argument as those in Theorem 2.2, we can also get the following result.

Theorem 2.3. *Under the conditions in Lemma 2.1, let* $\{g_k\}$ *be the sequence generated by applying the method* (8) *to problem* (1)*, we have*

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

where

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

with

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

2.2. The algorithm

Theorems 2.2 and 2.3 in the former subsection provide us the possibility of employing the two stepsizes $\hat{\alpha}_k$ and $\bar{\alpha}_k$ to significantly reduce the gradient components μ_1^k and μ_n^k. However, the following example shows some negative aspects of using $\hat{\alpha}_k$. Particularly, we applied the method (8) to a problem of (1) with

$$A = diag\{a_1, a_2, \ldots, a_n\} \quad \text{and} \quad b = 0, \tag{15}$$

where $a_1 = 1$, $a_n = n$ and a_i is randomly generated in $(1, n)$, $i = 2, \ldots, n-1$. Figure 1 presents the result of an instance with $n = 1,000$. We can see that $\bar{\alpha}_k$ approximates $\frac{1}{\lambda_n}$ with satisfactory accuracy in a few iterations. However, $\hat{\alpha}_k$ converges to $\frac{1}{\lambda_1}$ very slowly in the first few hundreds of iterations. although we did observe that after 1,000 iterations the value of $|\hat{\alpha}_k - \frac{1}{\lambda_1}|$ is reduced by a factor of 0.01.

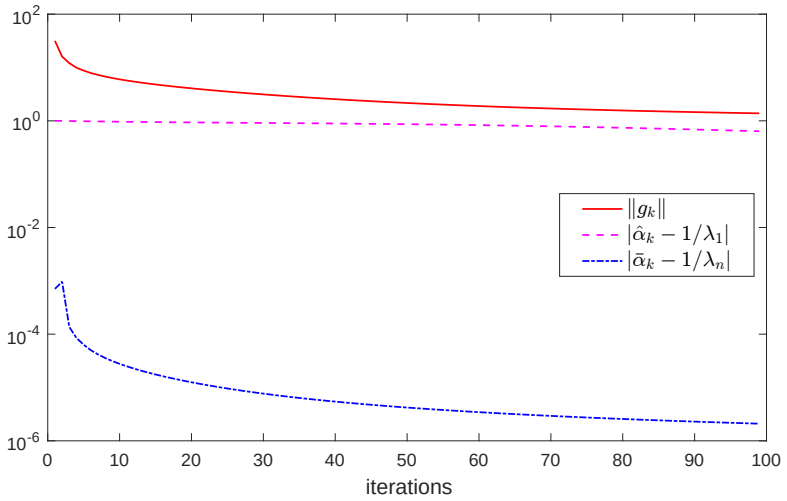

Figure 1. Problem (15): convergence history of the sequence $\{\hat{\alpha}_k\}$ and $\{\bar{\alpha}_k\}$ for the first 100 iterations of the method (8).

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

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

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

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

We consider the following cases:

Case 1. $|1 - \alpha_k\lambda_i| \leq 0.6$.

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

Case 2. $|1 - \alpha_k\lambda_i| > 0.6$.

(i) If $1 - \alpha_k\lambda_i \geq 0.9$, the value of $\frac{g_k^{(i)}}{\|g_k\|}$ will not change much and thus, $d_k^{(i)}$ may not affect the value of $\bar{\alpha}_k$ too much, which indicates that the component can be also neglected.

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

(iii) If $1 - \alpha_k\lambda_i < -0.6$, the value of $\frac{g_k^{(i)}}{\|g_k\|}$ changes signs and hence, $|d_k^{(i)}|$ may get significant increase.

The above analysis shows that $\bar{\alpha}_k$ will be mostly determined by the components corresponding to those eigenvalues in (iii) of Case 2. Notice that the required inequality in (iii) implies that $\lambda_i > \frac{4(\lambda_1+\lambda_n)}{5}$. If A has many large eigenvalues satisfying this condition, $\bar{\alpha}_k$ will be an estimation of the reciprocal of certain average of large eigenvalues. When A has few such large eigenvalues, $\bar{\alpha}_k$ will be mostly determined by the gradient components corresponding to the these few largest eigenvalues, which would yield a good estimation of $\frac{1}{\lambda_n}$. So, $\bar{\alpha}_k$ will approximate $\frac{1}{\lambda_n}$ with satisfactory accuracy in small number of iterations. This coincides with our observation in Figure 1.

For the stepsize $\hat{\alpha}_k$, when $1-\alpha_k\lambda_i > 0$, the value of $\frac{g_k^{(i)}}{\|g_k\|} + \frac{g_{k-1}^{(i)}}{\|g_{k-1}\|}$ may increase even when $1 - \alpha_k\lambda_i \leq 0.1$. So, most of the components of the gradient corresponding to those eigenvalues less than $\frac{1}{\alpha_k}$ will affect the value of $\hat{\alpha}_k$. Thus, $\hat{\alpha}_k$ would not be a good approximation of $\frac{1}{\lambda_1}$ until those components become very small. Moreover, a rough estimation of $\frac{1}{\lambda_1}$ may yield a large step which will increase most of the components of the gradient. As a result, it is impractical to use $\hat{\alpha}_k$ for eliminating the gradient component μ_1^k.

Based on the above observations, our method would combine the stepsizes $\bar{\alpha}_k$ and α_k^{AOPT}. In particular, our method takes h steps with α_k^{AOPT} to drive $\bar{\alpha}_k$ towards a good approximation of $\frac{1}{\lambda_n}$ and then takes s short steps in the hope of eliminating the corresponding component μ_n. As $\bar{\alpha}_k$ is expected to be short, we resort to α_k^{AOPT} if $\bar{\alpha}_k$ is relatively large. Hence, more precisely, we take

$$\alpha_k = \begin{cases} \alpha_k^{AOPT}, & \text{if } \mathrm{mod}(k, h+s) < h; \\ \min\{\alpha_k^{AOPT}, \bar{\alpha}_k\}, & \text{otherwise.} \end{cases} \tag{16}$$

Notice that, for quadratics, the BB1 stepsize α_k^{BB1} is just the former SD stepsize (3). As we know, the BB method performs much better than the SD method [21, 37]. And

gradient methods with retard stepsizes often have better performances [24]. Moreover, it can be seen from Figure 1 that $\bar{\alpha}_{k-1}$ is also a good approximation of $\frac{1}{\lambda_n}$ after about 10 to 20 iterations. Thus, we also consider to use the retard stepsize $\bar{\alpha}_{k-1}$, i.e.,

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

Numerical comparisons between the methods (16) and (17) in Table 1 show the benefits of using the one stepsize delay. Table 1 lists the averaged iterations of these two methods on solving 10 instances of problem (15), where the condition number of A is $\kappa = 10^4$ with $a_1 = 1$, $a_n = \kappa$, and other diagonal elements are randomly generated in $(1, \kappa)$. The iteration was stopped once the gradient norm is less than an ϵ factor of its initial value. We can see that the performance of the method (17) dominates that of (16) for most of the instances. Another advantage of using the retard stepsize $\bar{\alpha}_{k-1}$ is that it can be easily extended to more general problems. This will be more clear in Section 4.

Table 1. Number of averaged iterations of the methods (16) and (17).

method	ϵ	(h,s) for the method									
		$(10,20)$	$(10,30)$	$(10,50)$	$(10,80)$	$(10,100)$	$(20,20)$	$(20,30)$	$(20,50)$	$(20,80)$	$(20,100)$
(16)	10^{-6}	298.9	287.3	308.7	321.4	336.9	306.5	333.5	331.2	342.5	371.7
	10^{-9}	711.5	636.3	650.2	606.0	602.9	785.7	680.3	582.5	545.7	655.0
	10^{-12}	1048.0	893.1	1013.7	910.3	816.3	1225.8	989.8	904.1	773.8	892.4
(17)	10^{-6}	318.8	312.7	311.5	350.9	333.7	341.0	337.1	342.0	362.2	349.2
	10^{-9}	642.6	581.3	566.0	561.0	525.3	650.6	655.0	636.5	539.6	571.3
	10^{-12}	957.6	789.9	772.0	771.0	752.5	952.6	881.1	839.3	755.3	766.4

Remark 1. Although our method (17) looks like the SDC method (7), they differ in the following ways: (i) the method (17) does not use exact line searches which are necessary for the SDC method; (ii) the method (17) does not reuse any stepsize while the SDC method uses the same Yuan's stepsize α_k^Y for s steps; (iii) the method (17) is monotone while the SDC method is nonmonotone and its monotone version is obtained by using a safeguard with $2\alpha_k^{SD}$.

The analysis at the beginning of this subsection indicates that a small $\bar{\alpha}_k$ will be generated once its value is mostly determined by the first few largest eigenvalues. This can be achieved by using short stepsizes such that $|1 - \alpha_k \lambda_i| \leq 1$ hold for all $i = 1, 2, \ldots, n$ and several subsequent iterations. In addition, if there exist subsequences $\{\alpha_{k_i}\}$ approximate $\frac{1}{\lambda_i}$ for all but the first few largest eigenvalues, $\bar{\alpha}_k$ would be also small. Notice that the retard stepsize α_{k-1}^{AOPT} is an approximation of some $\frac{1}{\lambda_i}$ and also short in the sense that $\alpha_{k-1}^{AOPT} \leq \alpha_{k-1}^{SD}$. In fact, we can see from Figure 2 that, when the gradient method with α_{k-1}^{AOPT} is applied to problem (15), the stepsize $\bar{\alpha}_k$ approximates $\frac{1}{\lambda_n}$ with high accuracy if its value is small. Moreover, some promising numerical results of applying α_{k-1}^{AOPT} are given in [8]. So, motivated by the above observation and analysis, we also suggest the following nonmonotone variant of (17):

$$\alpha_k = \begin{cases} \alpha_{k-1}^{AOPT}, & \text{if } \mathrm{mod}(k, h+s) < h; \\ \min\{\alpha_{k-1}^{AOPT}, \bar{\alpha}_{k-1}\}, & \text{otherwise.} \end{cases} \tag{18}$$

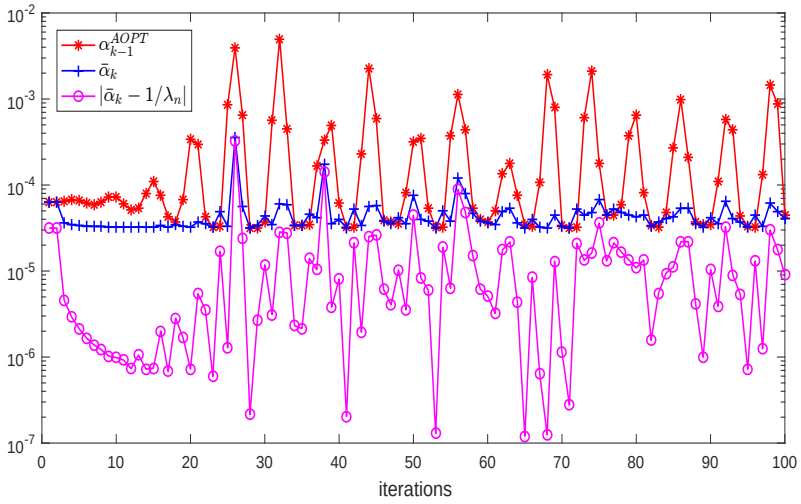

Figure 2. Problem (15): history of the sequence $\{\bar{\alpha}_k\}$ for the first 100 iterations of the gradient method with α_{k-1}^{AOPT}.

3. Convergence

In this section, we establish the R-linear convergence of the method (17) and its nonmonotone variant (18) for minimizing strongly convex quadratic function. Since the gradient method (2) is invariant under translations and rotations when applying to problem (1), we make the following assumption throughout the analysis.

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

$$A = \mathrm{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_n\}, \tag{19}$$

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

In order to give a unified analysis of the methods (17) and (18), we recall the following property given by Dai [7].

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

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

(ii) for any integer $l \in [1, n-1]$ and $\epsilon > 0$, if $G(k-j, l) \leq \epsilon$ and $(g_{k-j}^{(l+1)})^2 \geq M_2\epsilon$ hold for $j \in [0, \min\{k, m\} - 1]$, then $\alpha_k^{-1} \geq \frac{2}{3}\lambda_{l+1}$.

Here,

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

Dai [7] has proved that if A has the form (19) with $1 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and the stepsizes of gradient method (2) have the Property (A), then either $g_k = 0$ for some finite k or the sequence $\{\|g_k\|\}$ converges to zero R-linearly. Therefore, in order to establish R-linear convergence of the methods (17) and (18), we only need to show these methods satisfy Property (A).

Theorem 3.1. *Suppose that the sequence $\{\|g_k\|\}$ is generated by any of the method* (17) *or* (18) *applied to solve problem (1) with the matrix A having the form* (19) *and $1 = \lambda_1 < \lambda_2 < \cdots < \lambda_n$. Then either $g_k = 0$ for some finite k or the sequence $\{\|g_k\|\}$ converges to zero R-linearly.*

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

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

Notice that $\alpha_k^{AOPT} \le \alpha_k^{SD}$. For the method (18), we have that

$$\alpha_k^{-1} \ge (\alpha_{k-1}^{AOPT})^{-1} \ge (\alpha_{k-1}^{SD})^{-1}. \tag{20}$$

Suppose that $G(k-j,l) \le \epsilon$ and $(g_{k-j}^{(l+1)})^2 \ge 2\epsilon$ hold for $j \in [0, \min\{k,m\} - 1]$. It follows from (20) and the definition of α_k^{SD} that

$$\begin{aligned}\alpha_k^{-1} \ge (\alpha_{k-1}^{SD})^{-1} &= \frac{\sum_{i=1}^n \lambda_i (g_{k-1}^{(i)})^2}{\sum_{i=1}^n (g_{k-1}^{(i)})^2} \ge \frac{\lambda_{l+1} \sum_{i=l+1}^n (g_{k-1}^{(i)})^2}{G(k-1,l) + \sum_{i=l+1}^n (g_{k-1}^{(i)})^2} \\ &\ge \frac{\lambda_{l+1}}{\epsilon/2\epsilon + 1} \ge \frac{2}{3}\lambda_{l+1},\end{aligned}$$

where the first inequality in the second line is due to the assumption and $k-1 \in \{\max\{k-1,0\}, \ldots, k\}$. Thus, (ii) holds for method the (18). For the method (17), we obtain the desired inequality by replacing $k-1$ with k in the above analysis. This completes the proof. □

4. Extension to bound constrained optimization

In this section, we would like to extend the strategies and the stepsize (9) discussed in previous sections for the bound constrained optimization.

$$\min_{x \in \Omega} \quad f(x), \tag{21}$$

where f is a Lipschitz continuously differentiable function defined on the set $\Omega = \{x \in \mathbb{R}^n |\ l \le x \le u\}$. Here, $l \le x \le u$ means componentwise $l_i \le x_i \le u_i$ for all $i = 1, \ldots, n$. Clearly, when $l_i = -\infty$ and $u_i = +\infty$ for all i, problem (21) reduces to an unconstrained problem.

Our algorithm belongs to the class of projected gradient methods, which update the iterates as

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

where λ_k is a step length determined by some line searches and d_k is the search direction given by

$$d_k = P_\Omega(x_k - \alpha_k g_k) - x_k. \tag{22}$$

Here, $P_\Omega(\cdot)$ is the Euclidean projection onto Ω and α_k is our proposed stepsize.

For a general objective function, both α_{k-1}^{AOPT} in (8) and $\bar{\alpha}_{k-1}$ in (9) can not be computed as in Section 2 because the Hessian is usually difficult to obtain. Hence, we replace α_{k-1}^{AOPT} by the following positive stepsize suggested by Dai et al. [8]:

$$\alpha_k^P = \frac{\|s_{k-1}\|}{\|y_{k-1}\|}. \tag{23}$$

It is easy to see this stepsize is the geometrical mean of the two BB stepsizes in (4) and it will reduce to α_{k-1}^{AOPT} when the objective function is quadratic. One may see [8] for details about α_k^P. Note that by applying gradient projection methods for bound constrained optimization, the variables which are at the boundary usually changes during early iterations and often become unchanged at the end. So, the algorithm usually eventually solves an unconstrained problem in the subspace corresponding to free variables. Hence, for bound constrained optimization, we modify the stepsize (23) as the following:

$$\bar{\alpha}_k^P = \frac{\|s_{k-1}\|}{\|\bar{y}_{k-1}\|}, \tag{24}$$

where

$$\bar{y}_{k-1}^{(i)} = \begin{cases} 0, & \text{if } s_{k-1}^{(i)} = 0; \\ g_k^{(i)} - g_{k-1}^{(i)}, & \text{otherwise.} \end{cases} \tag{25}$$

We now reformulate the stepsize $\bar{\alpha}_{k-1}$ in (9) for general functions. In fact, for quadratics, by the update rule (2) we have

$$g_{k-1}^T g_k = \|g_{k-1}\|^2 - \alpha_{k-1} g_{k-1}^T A g_{k-1} = g_{k-1}^T A g_{k-1}(\alpha_{k-1}^{SD} - \alpha_{k-1})$$

and

$$g_{k-1}^T A g_k = g_{k-1}^T A g_{k-1} - \alpha_{k-1} g_{k-1}^T A^2 g_{k-1} = g_{k-1}^T A^2 g_{k-1}(\alpha_{k-1}^{MG} - \alpha_{k-1}),$$

which together with (9), (11) and (12) give

$$\bar{\alpha}_k = \frac{2 - 2\frac{g_{k-1}^T g_k}{\|g_{k-1}\|\|g_k\|}}{\frac{1}{\alpha_{k-1}^{SD}} + \frac{1}{\alpha_k^{SD}} - 2\frac{g_{k-1}^T A g_k}{\|g_{k-1}\|\|g_k\|}} = \frac{2 - 2\frac{\|g_{k-1}\|}{\|g_k\|}\frac{1}{\alpha_k^{BB1}}(\alpha_k^{BB1} - \alpha_{k-1})}{\frac{1}{\alpha_k^{BB1}} + \frac{1}{\alpha_k^{SD}} - 2\frac{\|g_{k-1}\|}{\|g_k\|}\frac{1}{\alpha_k^{BB1}\alpha_k^{BB2}}(\alpha_k^{BB2} - \alpha_{k-1})}.$$

Similarly as before, we would modify the BB stepsizes in the above formula, and replace α_k^{BB1} and α_k^{BB2} by

$$\bar{\alpha}_k^{BB1} = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T \bar{y}_{k-1}} \quad \text{and} \quad \bar{\alpha}_k^{BB2} = \frac{s_{k-1}^T \bar{y}_{k-1}}{\bar{y}_{k-1}^T \bar{y}_{k-1}}, \tag{26}$$

respectively, where $\bar{y}_{k-1}$ is given in (25). Note that in fact α_k^{BB1} automatically takes care of the changes of free variables since $\alpha_k^{BB1} = \bar{\alpha}_k^{BB1}$. Then, by replacing the

iteration number k with $k-1$, we have

$$\bar{\alpha}_{k-1} = \frac{2 - 2\frac{\|g_{k-2}\|}{\|g_{k-1}\|}\frac{1}{\bar{\alpha}_{k-1}^{BB1}}(\bar{\alpha}_{k-1}^{BB1} - \alpha_{k-2})}{\frac{1}{\bar{\alpha}_{k-1}^{BB1}} + \frac{1}{\bar{\alpha}_{k}^{BB1}} - 2\frac{\|g_{k-2}\|}{\|g_{k-1}\|}\frac{1}{\bar{\alpha}_{k-1}^{BB1}\bar{\alpha}_{k-1}^{BB2}}(\bar{\alpha}_{k-1}^{BB2} - \alpha_{k-2})}. \tag{27}$$

To ensure global convergence and achieve good performance, nonmonotone line searches [15, 27, 38] are usually employed for BB-like methods. Here, we prefer to use the adaptive nonmonotone line search proposed by Dai and Zhang [15], which is designed to accept BB stepsizes as frequently as possible. Particularly, the step length $\lambda_k = 1$ is accepted if

$$f(x_k + d_k) \leq f_r + \sigma g_k^T d_k, \tag{28}$$

where f_r is the so-called reference function value adaptively updated by the rules given in [15] and $\sigma \in (0, 1)$ is a line search parameter. However, when (28) is not accepted, an Armijo-type back tracking line search is performed to find the step length λ_k satisfying a relatively more strict condition

$$f(x_k + \lambda_k d_k) \leq \min\{f_{\max}, f_r\} + \sigma\lambda_k g_k^T d_k, \tag{29}$$

where $f_{\max}$ is the maximal function value in recent M iterations, i.e.,

$$f_{\max} = \max_{0 \leq i \leq \min\{k, M-1\}} f(x_{k-i}).$$

It has been observed that such an nonmonotone line search is specially suitable for BB-like methods [15].

Our specific gradient projection algorithm combining with the above nonmonotone line search is stated as Algorithm 1. It is proved in [15] that when the objective function is Lipschitz continuously differentiable, Algorithm 1 ensures convergence in the sense that

$$\lim_{k \to \infty} \inf \|g_k\| = 0.$$

For Algorithm 1, we have the following additional comments.

Remark 2. When $s_k^T y_k \leq 0$, both $\bar{\alpha}_{k+1}^{BB1}$ and $\bar{\alpha}_{k+1}^{BB2}$ are not well-defined. In this case, we simply take the stepsize $\alpha_{k+1} = 1/\|g_{k+1}\|$. Moreover, when the stepsize $\bar{\alpha}_k$ is negative, we would like to take the shorter stepsize $\bar{\alpha}_{k+1}^{BB2}$, since $\bar{\alpha}_{k+1}^{BB2} = \min\{\bar{\alpha}_{k+1}^{P}, \bar{\alpha}_{k+1}^{BB1}, \bar{\alpha}_{k+1}^{BB2}\}$. Here, $0 < \alpha_{\min} << \alpha_{\max}$ serves as the stepsize safeguards.

We would also propose two variants of Algorithm 1. As mentioned in Section 2, a small $\bar{\alpha}_k$ will be generated if there are subsequences $\{\alpha_{k_i}\}$ approximating $\frac{1}{\lambda_i}$ for all but the first a few largest eigenvalues. It has been pointed out in [14, 37] that the BB method reduces the gradient components more or less at the same asymptotic rate. In other words, the BB stepsize will approximate all the reciprocals of eigenvalues during the iteration process. Similar observations have been presented in [23]. Thus, for quadratic problem (1) we may consider the following two variants of (18), which

Algorithm 1 Gradient method for bound constrained minimization

1: Initialization: $x_1 \in \mathbb{R}^n$, $\epsilon, \sigma \in (0,1)$, $M, h, s \in \mathbb{N}$, $\alpha_1 \in [\alpha_{\min}, \alpha_{\max}]$.
2: **while** $\|g_k\| > \epsilon$ **do**
3: Compute the search direction d_k by (22);
4: Determine λ_k by nonmonotone line search (28) and (29);
5: $x_{k+1} = x_k + \lambda_k d_k$;
6: **if** $s_k^T y_k > 0$ **then**
7: **if** $\mathrm{mod}(k, h+s) \geq h$ **then**
8: Compute $\bar{\alpha}_k$ by (27);
9: **if** $\bar{\alpha}_k > 0$ **then**
10: $\tilde{\alpha}_{k+1} = \min\{\bar{\alpha}_k, \bar{\alpha}_{k+1}^P\}$;
11: **else**
12: $\tilde{\alpha}_{k+1} = \bar{\alpha}_{k+1}^{BB2}$;
13: **end if**
14: **else**
15: $\tilde{\alpha}_{k+1} = \bar{\alpha}_{k+1}^P$;
16: **end if**
17: $\alpha_{k+1} = \max\{\alpha_{\min}, \min\{\tilde{\alpha}_{k+1}, \alpha_{\max}\}\}$;
18: **else**
19: $\alpha_{k+1} = 1/\|g_{k+1}\|$;
20: **end if**
21: **end while**

combine BB stepsizes with the new stepsize (9):

$$\alpha_k = \begin{cases} \alpha_k^{BB1}, & \text{if } \mathrm{mod}(k, h+s) < h; \\ \min\{\alpha_k^{BB1}, \bar{\alpha}_{k-1}\}, & \text{otherwise,} \end{cases} \tag{30}$$

and

$$\alpha_k = \begin{cases} \alpha_k^{BB2}, & \text{if } \mathrm{mod}(k, h+s) < h; \\ \min\{\alpha_k^{BB2}, \bar{\alpha}_{k-1}\}, & \text{otherwise.} \end{cases} \tag{31}$$

In fact, we have also found reasonably good numerical performances of the methods (30) and (31) for minimizing quadratic functions. To generalize the methods (30) and (31) for bound constrained optimization, we can replace $\bar{\alpha}_{k+1}^P$ in lines 10 and 15 by $\bar{\alpha}_{k+1}^{BB1}$ and $\bar{\alpha}_{k+1}^{BB2}$, respectively. In what follows, we refer to Algorithm 1 using $\bar{\alpha}_{k+1}^P$, α_{k+1}^{BB1} and $\bar{\alpha}_{k+1}^{BB2}$ in lines 10 and 15 as A1, A1-BB1 and A1-BB2, respectively.

5. Numerical results

In this section, we do numerical experiments of the proposed methods for solving both quadratic and bound constrained optimization problems. All our codes were written in Matlab.

5.1. Quadratic problems

Firstly, we compare our methods (17) and (18) with the DY method (6) in [14], the $\text{ABB}_{\min 2}$ method in [23], and the SDC method (7) in [16] for minimizing quadratic problems. Note that the SDC method has been shown performing better than its monotone variants [16]. For all the comparison methods, the iteration stops when

$$\|g_k\| \leq \epsilon \|g_1\|, \tag{32}$$

where $\epsilon > 0$ is a given tolerance, or the iteration number exceeds 20,000. Based on the observation from Figures 1 and 2, we tested h with values 10 and 20 for our methods. As in [23], the parameter τ of the $\text{ABB}_{\min 2}$ method was set to 0.9 for all the problems.

Our first set of test problems are quadratic problems (1) from [10, 13, 24, 39], whose Hessian have different spectral distributions. In particular, the objective function has Hessian $A = QVQ^T$ with

$$Q = (I - 2w_3 w_3^T)(I - 2w_2 w_2^T)(I - 2w_1 w_1^T),$$

where w_1, w_2, and w_3 are unitary random vectors, $V = \text{diag}(v_1, \ldots, v_n)$ is a diagonal matrix with $v_1 = 1$ and $v_n = \kappa$, and v_j, $j = 2, \ldots, n-1$, being randomly generated between 1 and κ. The vector b were randomly generated with components between -10 and 10. Five sets of different spectral distributions of the test problems are given in Table 2 and the problem dimension is set as $n = 1000$. For each problem set, three different values of condition number κ and tolerances ϵ are tested. For each value of κ or ϵ, 10 problem instances were randomly generated. Tables 3 and 4 show the average number of iterations over those instances with the starting point $x_1 = (1, \ldots, 1)^T$, where the parameter pair (h, s) used for the SDC method was set to $(8, 6)$ which is more efficient than other choices for this test set.

Table 2. Distributions of v_j.

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

We can see from Table 3 that, our method (17) is competitive with the DY, $\text{ABB}_{\min 2}$ and SDC methods. For a fixed h, larger values of s seem to be preferable for the method (17). In addition, different settings of s lead to comparable results, with differences of less than 10% in the number of iterations for most of the test problems. For the first problem set, our method (17) outperforms the DY and SDC methods, although the $\text{ABB}_{\min 2}$ method seems surprisingly efficient for this first problem set among the compared methods. Particularly, when a high accuracy is required, the method (17) with $(h, s) = (20, 100)$ often takes less than $\frac{1}{6}$ and $\frac{1}{4}$ number of iterations needed by the DY and SDC methods, respectively. As for the second to fourth problem sets, the method (17) with different settings performs better than the DY and $\text{ABB}_{\min 2}$ methods and also peforms better than the SDC method if proper h and s are selected.

Table 3. Number of averaged iterations of the method (17), DY, $\text{ABB}_{\min 2}$ and SDC on problems in Table 2.

problem	ϵ	(h,s) for the method (17)										DY	$\text{ABB}_{\min 2}$	SDC
		$(10,20)$	$(10,30)$	$(10,50)$	$(10,80)$	$(10,100)$	$(20,20)$	$(20,30)$	$(20,50)$	$(20,80)$	$(20,100)$			
1	10^{-6}	332.9	317.7	317.1	327.6	340.1	360.3	326.6	321.8	324.0	327.7	439.0	**249.3**	382.7
	10^{-9}	2325.4	1298.7	1176.3	875.2	931.3	1509.2	1326.9	1030.9	789.6	754.9	3979.5	**489.1**	2970.0
	10^{-12}	4332.8	2379.9	2028.7	1349.0	1345.6	2795.3	2114.3	1651.0	1291.8	1111.0	7419.6	**629.6**	5113.0
2	10^{-6}	243.2	227.8	236.2	260.9	281.5	278.4	243.9	**226.9**	237.0	244.2	342.5	394.0	228.5
	10^{-9}	895.7	823.7	845.7	906.7	966.3	873.8	**803.0**	825.7	869.3	854.6	1584.3	1504.7	891.9
	10^{-12}	1537.0	1387.5	1372.0	1390.5	1446.0	1419.8	1284.3	**1257.4**	1359.0	1359.3	2748.1	2362.4	1410.0
3	10^{-6}	315.0	291.4	315.1	333.6	357.2	345.4	303.3	**272.5**	306.5	311.4	473.4	471.4	293.5
	10^{-9}	977.9	938.0	938.3	970.4	1025.0	1005.8	869.2	870.0	910.2	998.7	1766.6	1674.4	**860.8**
	10^{-12}	1577.8	1510.6	1515.0	1556.8	1587.9	1636.8	1363.8	**1351.3**	1434.6	1529.0	2846.7	2533.5	1397.7
4	10^{-6}	366.1	346.9	393.1	393.6	413.0	405.8	360.0	**334.7**	382.0	375.9	585.0	670.4	369.2
	10^{-9}	1069.5	988.7	996.2	1039.7	1063.1	1077.5	**904.1**	954.3	982.5	1029.1	1991.4	1644.8	977.2
	10^{-12}	1658.8	1508.7	1517.8	1620.1	1656.5	1658.3	1421.0	**1392.1**	1492.0	1565.5	3072.5	2559.2	1484.2
5	10^{-6}	826.3	**797.6**	845.9	856.3	909.4	895.5	876.1	885.7	877.7	899.1	878.8	1041.9	934.2
	10^{-9}	3500.9	3427.6	3388.3	3374.9	3432.8	3383.2	3466.1	3225.5	3234.4	**3156.2**	4275.8	3293.9	4117.5
	10^{-12}	5667.2	5428.9	5424.8	5303.5	5540.9	5868.5	5956.5	5484.2	5228.4	**5153.8**	7661.2	5310.5	6465.7
total	10^{-6}	2083.5	**1981.4**	2107.4	2172.0	2301.2	2285.4	2109.9	2041.6	2127.2	2158.3	2718.7	2827.0	2208.1
	10^{-9}	8769.4	7476.7	7344.8	7166.9	7418.5	7849.5	7369.3	6906.4	**6786.0**	6793.5	13597.6	8606.9	9817.4
	10^{-12}	14773.6	12215.6	11858.3	11219.9	11576.9	13378.7	12139.9	11136.0	10805.8	**10718.6**	23748.1	13395.2	15870.6

Table 4. Number of averaged iterations of the method (18), DY, $\text{ABB}_{\min 2}$ and SDC on problems in Table 2.

problem	ϵ	(h,s) for the method (18)										DY	$\text{ABB}_{\min 2}$	SDC
		$(10,20)$	$(10,30)$	$(10,50)$	$(10,80)$	$(10,100)$	$(20,20)$	$(20,30)$	$(20,50)$	$(20,80)$	$(20,100)$			
1	10^{-6}	345.2	338.9	351.7	330.9	344.4	354.6	335.1	329.0	336.2	336.0	439.0	**249.3**	382.7
	10^{-9}	1833.5	1537.8	1408.0	989.7	826.0	1602.5	1147.4	966.2	932.7	812.8	3979.5	**489.1**	2970.0
	10^{-12}	3569.5	2340.2	2249.1	1405.7	1124.8	2293.7	2169.0	1630.1	1304.4	1068.6	7419.6	**629.6**	5113.0
2	10^{-6}	220.4	**218.7**	222.7	224.9	234.4	278.0	263.2	244.6	242.6	255.8	342.5	394.0	228.5
	10^{-9}	821.9	815.7	760.9	**740.7**	807.4	956.5	910.4	828.9	818.4	862.3	1584.3	1504.7	891.9
	10^{-12}	1277.2	1247.4	**1214.4**	1216.1	1216.3	1544.0	1466.5	1344.9	1307.5	1364.2	2748.1	2362.4	1410.0
3	10^{-6}	**272.8**	275.9	284.6	291.7	309.0	353.6	333.0	305.4	303.8	320.3	473.4	471.4	293.5
	10^{-9}	**842.9**	848.6	844.4	889.3	863.5	1028.9	964.7	946.7	929.0	938.3	1766.6	1674.4	860.8
	10^{-12}	1367.1	1356.7	1312.8	1350.1	**1305.4**	1644.4	1511.1	1472.4	1446.4	1368.2	2846.7	2533.5	1397.7
4	10^{-6}	337.3	348.8	363.8	344.2	**333.9**	413.9	395.4	400.2	388.6	392.4	585.0	670.4	369.2
	10^{-9}	894.3	895.2	**869.1**	872.3	874.8	1101.8	1029.6	1014.8	989.3	977.1	1991.4	1644.8	977.2
	10^{-12}	1389.2	1385.7	1364.3	**1344.2**	1393.1	1676.0	1591.9	1554.4	1440.7	1438.8	3072.5	2559.2	1484.2
5	10^{-6}	**794.0**	802.7	811.9	831.9	876.0	843.1	832.3	822.3	805.9	853.5	878.8	1041.9	934.2
	10^{-9}	3415.2	3252.1	3081.2	3262.6	2995.6	3150.4	3259.6	2990.9	**2980.3**	3045.4	4275.8	3293.9	4117.5
	10^{-12}	5492.1	5272.7	5102.2	5102.0	4982.8	4861.3	5150.9	**4700.0**	5035.7	4808.7	7661.2	5310.5	6465.7
total	10^{-6}	**1969.7**	1985.0	2034.7	2023.6	2097.7	2243.2	2159.0	2101.5	2077.1	2158.0	2718.7	2827.0	2208.1
	10^{-9}	7807.8	7349.4	6963.6	6754.6	**6367.3**	7840.1	7311.7	6747.5	6649.7	6635.9	13597.6	8606.9	9817.4
	10^{-12}	13095.1	11602.7	11242.8	10418.1	**10022.4**	12019.4	11889.4	10701.8	10534.7	10048.5	23748.1	13395.2	15870.6

The method (17) also performs better than the DY and SDC methods on the last problem set. From the total number of iterations, we can see the overall performance of the method (17) is quite good. Here, we want to point out that our method (17) and the DY method are monotone, while the $\text{ABB}_{\min 2}$ and SDC methods are not.

Table 4 shows the averaged number of iterations of our method (18) for the first set of test problems. For comparison purposes, the results of the DY, $\text{ABB}_{\min 2}$ and SDC methods are also listed here. Similar performance as the method (17) can be observed. In particular, for each accuracy level, our method (18) takes around 30% less total iterations than the $\text{ABB}_{\min 2}$ method and also much less total iterations than the DY and SDC methods. Notice that problems of the last set are difficult for the compared methods since more iterations are needed than other four sets. However, the method (18) always dominates the compared three methods except with the pair $(h,s) = (10,20)$. As compared with the method (17), the retard strategy used in the method (18) tends to improve the performance when $h = 10$. For the case $h = 20$, the method (18) is also comparable to and better than (17) in terms of total number of iterations.

Our second set of quadratic test problems are the two large-scale real problems Laplace1(a) and Laplace1(b) described in [21]. Both of the problems require the solution of a system of linear equations derived from a 3D Laplacian on a box, discretized using a standard 7-point finite difference stencil. The solution is fixed by a Gaussian function whose center is (α, β, γ), multiplied by $x(x-1)$. A parameter σ is used to control the rate of decay of the Gaussian. Both Laplace1(a) and Laplace1(b) have $n = N^3$ variables, where N is the interior nodes taken in each coordinate direction, and have a highly sparse Hessian matrix with condition number $10^{3.61}$. We refer the readers to [21] for more details on these problems. In our tests, the associated parameters are set as follows:

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

We use the null vector as the starting point. The number of iterations required by the compared methods for solving the two problems Laplace1(a) and Laplace1(b) are listed in Tables 5 and 6, respectively.

Table 5. Number of iterations of the method (17), DY, $\text{ABB}_{\min 2}$ and SDC on the 3D Laplacian problem.

n	ϵ	(h,s) for the method (17)										DY	$\text{ABB}_{\min 2}$	SDC
		$(10,20)$	$(10,30)$	$(10,50)$	$(10,80)$	$(10,100)$	$(20,20)$	$(20,30)$	$(20,50)$	$(20,80)$	$(20,100)$			
Problem Laplace1(a)														
60^3	10^{-6}	271	241	243	197	331	321	247	211	211	241	249	**192**	213
	10^{-9}	348	**257**	421	357	334	521	351	281	303	331	373	329	393
	10^{-12}	451	394	437	452	441	523	401	**351**	459	482	546	401	529
80^3	10^{-6}	315	441	362	303	331	395	426	351	301	361	383	**289**	297
	10^{-9}	436	480	481	510	**349**	521	525	420	402	407	570	430	553
	10^{-12}	602	548	601	631	**451**	762	677	561	526	601	789	608	705
100^3	10^{-6}	500	482	**301**	371	441	441	351	421	401	361	427	351	513
	10^{-9}	691	639	541	527	565	562	**453**	556	457	601	651	485	609
	10^{-12}	826	900	649	808	771	881	734	701	**602**	721	918	687	825
total	10^{-6}	1086	1164	906	871	1103	1157	1024	983	913	963	1059	**832**	1023
	10^{-9}	1475	1376	1443	1394	1248	1604	1329	1257	**1162**	1339	1594	1244	1555
	10^{-12}	1879	1842	1687	1891	1663	2166	1812	1613	**1587**	1804	2253	1696	2059
Problem Laplace1(b)														
60^3	10^{-6}	327	241	241	271	331	281	351	**211**	245	241	236	217	213
	10^{-9}	361	361	361	**320**	367	521	351	352	401	361	399	365	437
	10^{-12}	509	482	481	451	**442**	641	451	585	502	507	532	502	555
80^3	10^{-6}	319	361	421	361	387	321	351	**281**	401	295	454	294	309
	10^{-9}	511	561	468	448	551	549	477	561	502	506	567	**433**	485
	10^{-12}	751	702	653	658	652	801	638	739	**601**	669	794	634	766
100^3	10^{-6}	393	401	396	361	532	402	393	421	425	**361**	371	369	379
	10^{-9}	632	635	602	631	662	801	752	701	701	707	700	**585**	653
	10^{-12}	931	961	902	901	991	961	1001	937	**802**	1024	1038	880	965
total	10^{-6}	1039	1003	1058	993	1250	1004	1095	913	1071	897	1061	**880**	901
	10^{-9}	1504	1557	1431	1399	1580	1871	1580	1614	1604	1574	1666	**1383**	1575
	10^{-12}	2191	2145	2036	2010	2085	2403	2090	2261	**1905**	2200	2364	2016	2286

From Table 5 we can see that, for the problem Laplace1(a), our method (17) with a large s outperforms the DY and SDC methods when the accuracy level is high. Moreover, with proper (h, s), it performs better than the $\text{ABB}_{\min 2}$ method, which dominates the DY and SDC methods. For the problem Laplace1(b), our method (17) is also competitive with the compared methods. Table 6 shows similar results as the former one. However, for the problem Laplace1(b), our method (18) clearly outperforms the DY and SDC methods especially when high accurate solutions are needed. Moreover, the method (18) is very competitive with the $\text{ABB}_{\min 2}$ method in terms of total number of iterations.

Table 6. Number of iterations of the method (18), DY, $\text{ABB}_{\min 2}$ and SDC on the 3D Laplacian problem.

n	ϵ	(h,s) for the method (18)										DY	$\text{ABB}_{\min 2}$	SDC
		$(10,20)$	$(10,30)$	$(10,50)$	$(10,80)$	$(10,100)$	$(20,20)$	$(20,30)$	$(20,50)$	$(20,80)$	$(20,100)$			
Problem Laplace1(a)														
60^3	10^{-6}	279	208	241	209	283	241	201	212	247	241	249	**192**	213
	10^{-9}	421	361	308	326	331	320	301	357	301	**241**	373	329	393
	10^{-12}	486	424	342	380	361	363	396	427	**322**	367	546	401	529
80^3	10^{-6}	301	335	**271**	290	303	303	278	367	308	361	383	289	297
	10^{-9}	489	490	421	376	450	**359**	384	561	499	481	570	430	553
	10^{-12}	676	681	552	596	551	524	585	636	515	**495**	789	608	705
100^3	10^{-6}	474	361	361	541	397	441	451	421	416	361	427	**351**	513
	10^{-9}	721	441	**439**	576	561	681	536	529	601	481	651	485	609
	10^{-12}	811	761	612	740	771	786	701	644	756	**601**	918	687	825
total	10^{-6}	1054	904	873	1040	983	985	930	1000	971	963	1059	**832**	1023
	10^{-9}	1631	1292	**1168**	1278	1342	1360	1221	1447	1401	1203	1594	1244	1555
	10^{-12}	1973	1866	1506	1716	1683	1673	1682	1707	1593	**1463**	2253	1696	2059
Problem Laplace1(b)														
60^3	10^{-6}	229	241	241	271	258	249	277	281	305	241	236	217	**213**
	10^{-9}	364	361	385	335	344	396	411	**334**	399	368	399	365	437
	10^{-12}	546	451	541	467	551	561	443	**413**	505	496	532	502	555
80^3	10^{-6}	309	278	312	**271**	361	292	289	363	363	460	454	294	309
	10^{-9}	521	521	547	528	551	443	507	491	501	601	567	**433**	485
	10^{-12}	684	692	721	619	**614**	681	642	631	679	721	794	634	766
100^3	10^{-6}	391	450	355	361	402	361	385	**351**	362	362	371	369	379
	10^{-9}	565	657	630	707	771	561	644	**491**	701	650	700	585	653
	10^{-12}	**731**	904	972	921	771	880	911	841	814	881	1038	880	965
total	10^{-6}	929	969	908	903	1021	902	951	995	1030	1063	1061	**880**	901
	10^{-9}	1450	1539	1562	1570	1666	1400	1562	**1316**	1601	1619	1666	1383	1575
	10^{-12}	1961	2047	2234	2007	1936	2122	1996	**1885**	1998	2098	2364	2016	2286

5.2. Bound constrained problems

This subsection compares our methods A1, A1-BB1 and A1-BB2 with the spectral projected gradient (SPG) method [3, 4], which is a nonmonotone projected gradient method using the Barzilai-Borwein stepsize.

For our methods, the parameter values are set as the following:

$$\alpha_{\min} = 10^{-30},\ \alpha_{\max} = 10^{30},\ h = 10,\ s = 4,\ M = 8,\ \sigma = 10^{-4}.$$

Default parameters were used for SPG [4]. The stopping condition for all methods is

$$\|P_\Omega(x_k - g_k) - x_k\|_\infty \le 10^{-6}.$$

Our test problem set consists of all bound constrained problems from the CUTEst collection [26] with dimension more than 50. There are 3 problems for which none of these comparison algorithms can solve. Hence, we simply delete them and only 47 problems are left for our test.

We compare all these algorithms by using the performance profiles of Dolan and Moré [18] on different metric. In these performance profiles, the vertical axis shows the percentage of the problems the method solves within the factor τ of the metric used by the most effective method in this comparison. Figure 3 shows the performance profiles on the number of iterations. It can be observed that our methods A1, A1-BB1 and A1-BB2 clearly outperform SPG in terms of iteration numbers. We can also see from Figure 4 that the performance gap is even larger in terms of the number of function evaluations. Moreover, Figure 5 shows that our methods are also better than SPG in terms of the overall CPU time.

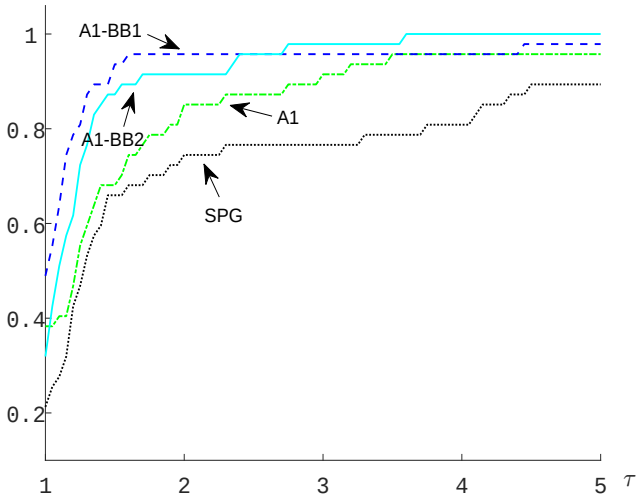

Figure 3. Performance profiles, iteration metric, 47 CUTEst bound constrained problems.

6. Conclusions

Based on the asymptotic optimal stepsize, we have proposed a new monotone gradient method, which employs a new stepsize that converges to the reciprocal of the largest eigenvalue of the Hessian of the objective function. A nonmonotone variant of this method has been proposed as well. R-linear convergence of the proposed methods has been established for minimizing strongly convex quadratic functions. Our numerical experiments on minimizing quadratic functions show that the proposed methods are very effective with other recent successful gradient methods.

By making use of projected gradient strategy and the Dai-Zhang nonmonotone line search [15], the proposed methods are extended for solving general bound constrained optimization. In addition, we have also proposed two variants of those methods based on the Barzilai-Borwein stepsizes. Numerical comparisons with the spectral projected gradient (SPG) method [3, 4] on bound constrained problems from the CUTEst collection show that these new methods are very promising for solving bound constrained optimization.

Funding

This work was supported by the National Natural Science Foundation of China (11701137, 11631013, 11671116), by the National 973 Program of China (2015CB856002), by the China Scholarship Council (No. 201806705007), and by the USA National Science Foundation (1522654, 1819161).

References

[1] 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.

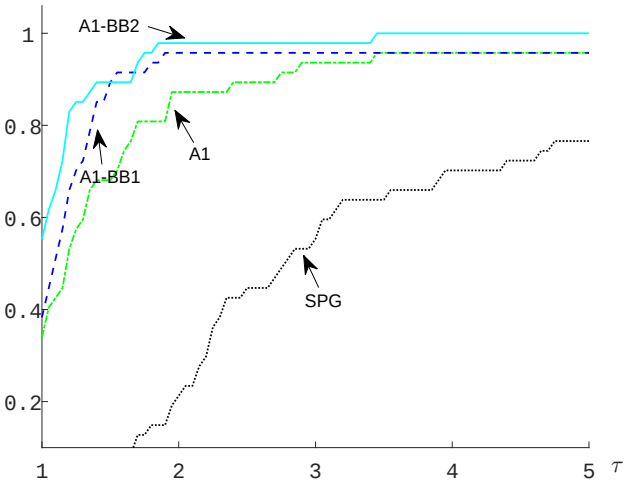

Figure 4. Performance profiles, function evaluation metric, 47 CUTEst bound constrained problems.

[2] J. Barzilai and J.M. Borwein, *Two-point step size gradient methods*, IMA J. Numer. Anal. 8 (1988), pp. 141–148.

[3] 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.

[4] E.G. Birgin, J.M. Martínez, M. Raydan, *et al.*, *Spectral projected gradient methods: review and perspectives*, J. Stat. Softw. 60 (2014), pp. 539–559.

[5] A. Cauchy, *Méthode générale pour la résolution des systemes d'équations simultanées*, Comp. Rend. Sci. Paris 25 (1847), pp. 536–538.

[6] C. Cortes and V. Vapnik, *Support-vector networks*, Mach. Learn. 20 (1995), pp. 273–297.

[7] Y.H. Dai, *Alternate step gradient method*, Optimization 52 (2003), pp. 395–415.

[8] Y.H. Dai, M. Al-Baali, and X. Yang, *A positive Barzilai–Borwein-like stepsize and an extension for symmetric linear systems*, in *Numerical Analysis and Optimization*, Springer, 2015, pp. 59–75.

[9] Y.H. Dai and R. Fletcher, *Projected Barzilai–Borwein methods for large-scale box-constrained quadratic programming*, Numer. Math. 100 (2005), pp. 21–47.

[10] Y.H. Dai, Y. Huang, and X.W. Liu, *A family of spectral gradient methods for optimization*, preprint (2018). Available at: http://www.optimization-online.org/DB_HTML/2018/05/6628.html.

[11] 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.

[12] Y.H. Dai and X. Yang, *A new gradient method with an optimal stepsize property*, Comp. Optim. Appl. 33 (2006), pp. 73–88.

[13] Y.H. Dai and Y.X. Yuan, *Alternate minimization gradient method*, IMA J. Numer. Anal. 23 (2003), pp. 377–393.

[14] Y.H. Dai and Y.X. Yuan, *Analysis of monotone gradient methods*, J. Ind. Mang. Optim. 1 (2005), pp. 181–192.

[15] Y.H. Dai and H. Zhang, *Adaptive two-point stepsize gradient algorithm*, Numer. Algor. 27 (2001), pp. 377–385.

[16] 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.

[17] 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.

[18] E.D. Dolan and J.J. Moré, *Benchmarking optimization software with performance profiles*,

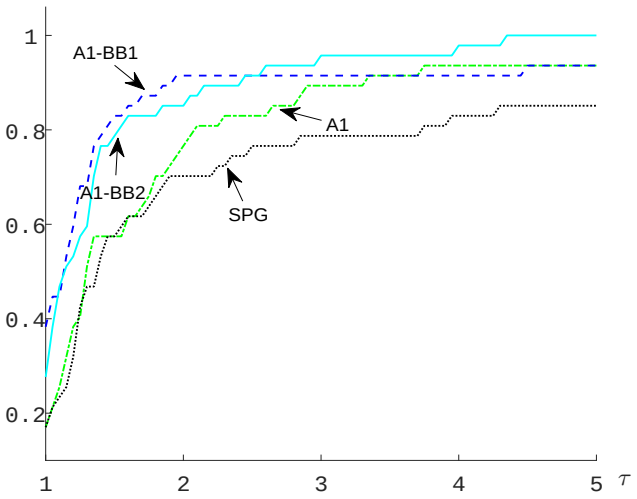

Figure 5. Performance profiles, CPU time metric, 47 CUTEst bound constrained problems.

Math. Program. 91 (2002), pp. 201–213.

[19] H.C. Elman and G.H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal. 31 (1994), pp. 1645–1661.

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

[21] R. Fletcher, *On the Barzilai–Borwein method*, in *Optimization and Control with Applications*, L. Qi, K. Teo, X. Yang, P. M. Pardalos and D. Hearn, eds., Springer, US, 2005, pp. 235–256.

[22] G.E. Forsythe, *On the asymptotic directions of the s-dimensional optimum gradient method*, Numer. Math. 11 (1968), pp. 57–76.

[23] G. Frassoldati, L. Zanni, and G. Zanghirati, *New adaptive stepsize selections in gradient methods*, J. Ind. Mang. Optim. 4 (2008), pp. 299–312.

[24] 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.

[25] C.C. Gonzaga and R.M. Schneider, *On the steepest descent algorithm for quadratic functions*, Comp. Optim. Appl. 63 (2016), pp. 523–542.

[26] 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.

[27] 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.

[28] Y. Huang and H. Liu, *Smoothing projected Barzilai–Borwein method for constrained non-Lipschitz optimization*, Comp. Optim. Appl. 65 (2016), pp. 671–698.

[29] Y. Huang, H. Liu, and S. Zhou, *Quadratic regularization projected Barzilai–Borwein method for nonnegative matrix factorization*, Data Min. Knowl. Disc. 29 (2015), pp. 1665–1684.

[30] B. Jiang and Y.H. Dai, *Feasible Barzilai–Borwein-like methods for extreme symmetric eigenvalue problems*, Optim. Method Softw. 28 (2013), pp. 756–784.

[31] D.D. Lee and H.S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature 401 (1999), pp. 788–791.

[32] Y.F. Liu, Y.H. Dai, and Z.Q. Luo, *Coordinated beamforming for MISO interference channel: complexity analysis and efficient algorithms*, IEEE Trans. Signal Process. 59 (2011), pp. 1142–1157.

[33] J. Nocedal, A. Sartenaer, and C. Zhu, *On the behavior of the gradient norm in the steepest descent method*, Comp. Optim. Appl. 22 (2002), pp. 5–35.
[34] M. Raydan, *On the Barzilai and Borwein choice of steplength for the gradient method*, IMA J. Numer. Anal. 13 (1993), pp. 321–326.
[35] M. Raydan, *The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem*, SIAM J. Optim. 7 (1997), pp. 26–33.
[36] Y.X. Yuan, *A new stepsize for the steepest descent method*, J. Comput. Math. (2006), pp. 149–156.
[37] Y.X. Yuan, *Step-sizes for the gradient method*, AMS IP Studies in Advanced Mathematics 42 (2008), pp. 785–796.
[38] 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.
[39] B. Zhou, L. Gao, and Y.H. Dai, *Gradient methods with adaptive step-sizes*, Comp. Optim. Appl. 35 (2006), pp. 69–86.