文章编号:2095-6134(2018)03-0289-08

New project gradient methods for quadratic programming with linear equality constraints*

LI Mingqiang, GUO Tiande, HAN Congying†

(*School of Mathematical Sciences*, *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*)

(Received 28 Feburary 2017; Revised 12 April 2017)

Li M Q, Guo T D, Han C Y. New project gradient methods for quadratic programming with linear equality constraints [J]. Journal of University of Chinese Academy of Sciences, 2018,35(3):289-296.

Abstract Recently, researchers proposed many accelerated project gradient methods based on new step length choice rules for large scale optimization problems. In this paper, we propose two project gradient methods with variants of new selection rules for quadratic programming with linear equality constraints. One is a non-monotone project gradient method for which an adaptive line search method is adopted and the Barzilai-Borwein step size is applied, and the other is a monotoneproject gradient method with Yuan step size. We give the global convergence of these two methods under mild assumptions. Numerical experiments indicate that both the new methods are more efficient than traditional project gradient methods.

Keywords Barzilai-Borwein step size; linear equality; project gradient

CLC number:0224 **Document code**: A **doi**:10.7523/j.issn.2095-6134.2018.03.001

等式约束二次规划问题的新的梯度投影算法

李明强, 郭田德, 韩丛英

(中国科学院大学数学科学学院, 北京 100049)

摘 要 近些年来,众多学者提出基于新步长选择策略的加速梯度投影算法求解大规模优化问题。本文针对线性约束二次规划问题提出两种基于新步长的梯度投影算法。一种是基于采用自适应线搜索和 Barzilai-Borwein 步长的非单调投影算法。另一种是基于 Yuan 步长的单调投影算法。在较弱的假设条件下,给出这两种算法的全局收敛性。数值实验表明新算法比传统的梯度投影算法求解效率更高。

关键词 Barzilai-Borwein 步长;线性等式;投影梯度

In this paper, we consider the quadratic programming problem

$$\min_{x\in\Omega} f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}, \tag{1}$$

* Supported by National Natural Science Fund of China(11101420,71271204,11331012)

† Corresponding author, E-mail: hancy@ucas.ac.cn

where $\boldsymbol{Q} \in \mathbf{R}^{n\times n}$ is symmetric and positive definite, $\boldsymbol{x}, \boldsymbol{c} \in \mathbf{R}^n$, and the feasible region Ω is defined by

$$\Omega = \{\boldsymbol{x} \in \mathbf{R}^n, \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}, \tag{2}$$

where $\boldsymbol{A} \in \mathbf{R}^{m\times n}$, rank $(\boldsymbol{A}) = m$, and $m < n$.

First of all, we give a brief review for the development of the efficient step length selections in gradient methods for minimizing a large scale strictly convex quadratic function $f(\boldsymbol{x})$. Assume that $\boldsymbol{g}(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$ can be obtained at every $\boldsymbol{x}$ and $\boldsymbol{g}_k = \boldsymbol{g}(\boldsymbol{x}_k)$ is the gradient at $\boldsymbol{x}_k$. The classical examples of step size selections are the line searches used in the steepest descent (SD) and the minimal gradient (MG) methods, which minimize $f(\boldsymbol{x}_k - \alpha\boldsymbol{g}_k)$ and $\|\boldsymbol{g}(\boldsymbol{x}_k - \alpha\boldsymbol{g}_k)\|_2$, respectively,

$$\begin{aligned}\alpha_k^{SD} &= \operatorname*{argmin}_{\alpha} f(\boldsymbol{x}_k - \alpha\boldsymbol{g}_k)\\ &= \frac{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{g}_k}{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{g}_k},\end{aligned} \tag{3}$$

$$\begin{aligned}\alpha_k^{MG} &= \operatorname*{argmin}_{\alpha} \|\boldsymbol{g}(\boldsymbol{x}_k - \alpha\boldsymbol{g}_k)\|_2\\ &= \frac{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{g}_k}{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{Q}^2\boldsymbol{g}_k}.\end{aligned} \tag{4}$$

Though the SD gradient method converges linearly, the convergence rate may be very slow, especially when $\boldsymbol{Q}$ is ill-conditioned in the sense that the condition number cond$(\boldsymbol{Q})$ is very large. The cond $(\boldsymbol{Q})$ is defined as follows:

$$\text{cond}(\boldsymbol{Q}) = \frac{\lambda_n(\boldsymbol{Q})}{\lambda_1(\boldsymbol{Q})},$$

where $\lambda_n(\boldsymbol{Q})$ and $\lambda_1(\boldsymbol{Q})$ represent the largest and smallest eigenvalues of $\boldsymbol{Q}$, respectively.

In 1988, a significant step size selection is proposed by Barzilai and Borwein (see Ref. [1]),

$$\alpha_k^{BB1} = \frac{\boldsymbol{s}_{k-1}^{\mathrm{T}}\boldsymbol{s}_{k-1}}{\boldsymbol{s}_{k-1}^{\mathrm{T}}\boldsymbol{y}_{k-1}}, \tag{5}$$

$$\alpha_k^{BB2} = \frac{\boldsymbol{s}_{k-1}^{\mathrm{T}}\boldsymbol{y}_{k-1}}{\boldsymbol{y}_{k-1}^{\mathrm{T}}\boldsymbol{y}_{k-1}}, \tag{6}$$

where $\boldsymbol{s}_{k-1} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$ and $\boldsymbol{y}_{k-1} = \boldsymbol{g}_k - \boldsymbol{g}_{k-1}$. We refer to the step size (5) or (6) as BB (Barzilai and Borwein) step size. From (3) and (4), we can easily obtained the fact that in the unconstrained case, $\alpha_k^{BB1} = \alpha_{k-1}^{SD}$ and $\alpha_k^{BB2} = \alpha_{k-1}^{MG}$ (see Ref. [2]). In other words, the step lengths (5) and (6) use the SD and MG step lengths with one delay, respectively. A feature of BB step size is that the sequences $\{f(\boldsymbol{x}_k)\}$ and $\{\|\boldsymbol{g}_k\|_2\}$ are non-monotone, in contrast to the SD and MG methods. Global convergence of the BB method was established by Raydan[3] in the strictly convex quadratic case with either (5) or (6). Dai and Liao[4] have shown that the BB method is R-linearly convergent in the n-dimensional case. Moreover, numerical results indicate that the BB method outperforms the method with (3) or (4) for convex quadratic functions (see also Ref. [2]). Starting from (5) and (6), BB-like methods with longer delays have been studied[5-7]. In Ref. [5], Dai and Fletcher proposed a formula

$$\alpha_k^M = \frac{\sum_{i=1}^{M}\boldsymbol{s}_{k-i}^{\mathrm{T}}\boldsymbol{s}_{k-i}}{\sum_{i=1}^{M}\boldsymbol{s}_{k-i}^{\mathrm{T}}\boldsymbol{y}_{k-i}}, \tag{7}$$

where M is a positive integer, and they suggest that $M = 2$ is a better choice for box-constrained quadratic programming.

Recently, some monotone methods which are competitive to the BB-like method are proposed. One of these methods with step length

$$\alpha_k^{Y} = \frac{2}{\varphi_k^{Y} + 1\alpha_{k-1}^{SD} + 1\alpha_k^{SD}}, \tag{8}$$

where

$$\varphi_k^{Y} = \sqrt{(1\alpha_{k-1}^{SD} - 1\alpha_k^{SD})^2 + 4\|\boldsymbol{g}_k\|_2^2\|\boldsymbol{s}_{k-1}\|_2^2},$$

was proposed by Yuan in Ref. [8]. A property of (8) is that if α_k^{Y} has been taken after α_k^{SD}, only one more step α_k^{SD} is needed to get the solution to the minimizer of the two-dimensional strictly convex quadratics. If $\boldsymbol{x}_k$ is obtained by (3), then $\boldsymbol{s}_{k-1} = -\alpha_{k-1}^{SD}\boldsymbol{g}_{k-1}$. Thus a variant of (8) was proposed in Ref. [9] as follows:

$$\alpha_k^{VY} = \frac{2}{\varphi_k^{VY} + 1\alpha_{k-1}^{SD} + 1\alpha_k^{SD}}, \tag{9}$$

where

$$\varphi_k^{VY} = \sqrt{(1\alpha_{k-1}^{SD} - 1\alpha_k^{SD})^2 + \frac{4\|\boldsymbol{g}_k\|_2^2}{(\alpha_{k-1}^{SD}\|\boldsymbol{g}_{k-1}\|_2)^2}}.$$

Based on (9), Dai and Yuan[9] proposed a gradient method whose step length is given by

$$\alpha_k = \begin{cases}\alpha_k^{SD} & \text{if } \operatorname{mod}(k,4) = 1 \text{ or } 2,\\ \alpha_k^{VY} & \text{otherwise.}\end{cases} \tag{10}$$

It is not difficult to verify the monotonicity of the

method because $\alpha_k^{VY} \leqslant 2\alpha_k^{SD}$.

The efficiency of the project gradient methods with the above step length selections has been verified by a large amount of numerical experiments on box-constrained quadratic programming[10-13] and Support Vector Machines (SVMs)[5,14-16]. Inspired by the success of these new step selection rules, we improve project gradient methods for (1).

1 PBB method analysis

In this section, we focus our attention on designing efficient project gradient type methods for solving (1). Here we let $\boldsymbol{P}$ denote the projection operator on Ω:

$$\boldsymbol{P}(\boldsymbol{x}) = \underset{\boldsymbol{y}\in\Omega}{\operatorname{argmin}} \|\boldsymbol{x}-\boldsymbol{y}\|_2 \tag{11}$$

$$= \boldsymbol{x} - \boldsymbol{T}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}), \tag{12}$$

where $\boldsymbol{T} = \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}$. Let $\boldsymbol{d}(\boldsymbol{x})$ be defined in terms of the gradient $\boldsymbol{g}(\boldsymbol{x})$ as follows:

$$\boldsymbol{d}(\boldsymbol{x}) = \boldsymbol{P}(\boldsymbol{x}-\boldsymbol{g}(\boldsymbol{x})) - \boldsymbol{x}$$

$$= -\boldsymbol{H}\boldsymbol{g}(\boldsymbol{x}), \tag{13}$$

where $\boldsymbol{H} = \boldsymbol{I} - \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{A}$.

Given an iterate point $\boldsymbol{x}_k$, the project gradient method for (1) computes the next point in the following form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k, \tag{14}$$

where $\boldsymbol{d}_k = \boldsymbol{d}(\boldsymbol{x}_k)$ and $\alpha_k > 0$ is a step length. It is easy to know that $\boldsymbol{s}_{k-i} = \alpha_{k-i}\boldsymbol{d}_{k-i}$ and $\boldsymbol{y}_{k-i} = \alpha_{k-i}\boldsymbol{Q}\boldsymbol{d}_{k-i}$, so we obtain another form of (7)

$$\alpha_k^M = \frac{\sum_{i=1}^{M}\alpha_{k-i}^2\boldsymbol{d}_{k-i}^{\mathrm{T}}\boldsymbol{d}_{k-i}}{\sum_{i=1}^{M}\alpha_{k-i}^2\boldsymbol{d}_{k-i}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}_{k-i}}. \tag{15}$$

We refer to the project gradient method with BB-like step length (15) as the projected BB method or PBB method. Obviously, the formula (5) is as a special case of (15) with $M = 1$.

Compared with original project gradient method (PSD), the PBB method is much more effective. In PSD, there is $\alpha_k = \alpha_k^{SD'}$, where

$$\alpha_k^{SD'} = \min_{\alpha} f(\boldsymbol{x}_k + \alpha\boldsymbol{d}_k)$$

$$= -\frac{\boldsymbol{g}_k^{\mathrm{T}}\boldsymbol{d}_k}{\boldsymbol{d}_k^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}_k}. \tag{16}$$

In order to illustrate the performance of the PBB method for (1), we have generated 10 random test problems with $n = 1\,000$, $m = 200$ and $M = 2$. $\boldsymbol{A}$, $\boldsymbol{c}$ and the initial point $\boldsymbol{x}_0$ is generated randomly, $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_0$. Also $\boldsymbol{Q}$ is randomly generated with a range of condition numbers from 10^2 to 10^4. The terminal condition is $\|\boldsymbol{d}_k\|_\infty < 10^{-4}$. The numerical results of the PBB method are presented in Table 1.

Algorithm 1 MPBB method

Step 1: Let $\boldsymbol{x}_0$ be a feasible point of (1), $f_0^r = \infty$, $f_0^s = f(\boldsymbol{x}_0)$, $k = 0, l = 0, L, M \in N_+$, $\boldsymbol{H} = \boldsymbol{I} - \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{A}$, $\alpha_0^M = 1$, $k_0^s = k_0^r = 0, i = h = 0$.

Step 2: $\boldsymbol{d}_k = -\boldsymbol{H}\boldsymbol{g}_k$ and terminate if $\|\boldsymbol{d}_k\|_\infty = 0$.

Step 3: If $f(\boldsymbol{x}_k + \alpha_k^M\boldsymbol{d}_k) \geqslant f_h^r$

$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k^{SD'}\boldsymbol{d}_k$

else

$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k^M\boldsymbol{d}_k$

end.

Step 4: If $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_{k_i^s})$

$k_{i+1}^s = k+1, k_{h+1}^r = k+1, i = i+1, l = 0$

else

$l = l + 1$

if $l = L$

$h = h+1, f_h^r = \max_{0\leqslant j\leqslant L}\{f(\boldsymbol{x}_{k_h^r+j})\}, k_{h+1}^r = k_h^r + L, l = 0$

end

end.

Step 5: $k = k+1$, goto Step2.

Table 1　Numerical results of PSD and PBB methods

Problem	PBB		PSD	
cond($\boldsymbol{Q}$)	iter	sec	iter	sec
10e+3	126	0.86	1145	5.57
10e+2	47	0.39	213	1.30
10e+3	94	0.69	977	4.64
10e+4	334	1.97	5 333	23.37
10e+3	115	0.81	1 019	4.61
10e+3	154	0.94	1 151	5.23
10e+4	351	1.66	4 673	20.40
10e+2	49	0.52	207	1.02
10e+2	56	0.47	219	1.06
10e+4	302	1.55	4 411	19.05
average	162.8	0.98	1 934.8	8.67

In Table 1, cond($\boldsymbol{Q}$) is the condition number of $\boldsymbol{Q}$, iter and sec denote the iteration numbers and the time required respectively.

From Table 1, we can see that the PBB method is much more effective in solving the 10 test problems compared with the PSD method. PBB

method with an average of 162.8 iterations and 0.98 s have an obvious advantage than PSD with an average of 1 934.8 and 8.67 s.

2 Two efficient project gradient methods

Though the PBB method performs very well in solving (1), there is no theory to guarantee its global convergence in constrained case[10]. To ensure global convergence of this method, we modify this method by incorporating some sort of line search. It is important that the line search does not degrade the performance of the unmodified PBB method.

Firstly, the sequence $\{f(\boldsymbol{x}_k)\}$ generated by the PBB method is non-monotone, so a non-monotone line search is necessary. Secondly, we take α_k^M as the step length at each iteration as much as possible. Based on these two considerations, the GLL (Grippo-Lampariello-Lucidi)[17] non-monotone line search is a good choice. We refer to the modified PBB method with the GLL non-monotone line search as MPBB method (see Algorithm 1).

$f(\boldsymbol{x}_{k_i^s})$ is the smallest objective function value over the past k_i^s iterations, that is,

$$f(\boldsymbol{x}_{k_i^s}) = \min_{0\leqslant j\leqslant k_i^s} f(\boldsymbol{x}_j).$$

Obviously $\{f(\boldsymbol{x}_{k_i^s})\}$ is a strictly monotone decreasing sequence. If no smaller function value is found in $L+1$ iterations, the reference function value f_h^r will be updated by the maximum function value in most recent $L+1$ iterations.

Algorithm 2 Monotone algorithm

Step 1: Let $\boldsymbol{x}_0$ be a feasible point of (1), $k = 0, \boldsymbol{H} = \boldsymbol{I} - \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})^{-1}\boldsymbol{A}$.

Step 2: $\boldsymbol{d}_k = -\boldsymbol{H}\boldsymbol{g}_k$ and terminate if $\|\boldsymbol{d}_k\|_\infty = 0$.

Step 3: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, where $\alpha_k = \rho_k \alpha_k^{SD'}, \rho_k \in [0,2]$.

Step 4: $k = k+1$; goto Step2.

Step 5: $k = k+1$, goto Step2.

Next, we will introduce a kind of monotone project gradient method with a new step length similar to (10). Now we give a general form of the monotone project gradient method for (1) (see Algorithm 2).

The following step length similar to α_k^{VY}

$$\alpha_k^{VY'} = \frac{2}{\varphi_k^{VY'} + 1\alpha_{k-1}^{SD'} + 1\alpha_k^{SD'}} \tag{17}$$

where

$$\varphi_k^{VY'} = \sqrt{(1\alpha_{k-1}^{SD'} - 1\alpha_k^{SD'})^2 + \frac{4\|\boldsymbol{g}_k\|_2^2}{(\alpha_{k-1}^{SD'}\|\boldsymbol{g}_{k-1}\|_2)^2}}$$

is given by replacing α_k^{SD} with $\alpha_k^{SD'}$ in (9). Following the formula (10), the step length

$$\alpha_k^{SY} = \begin{cases} \alpha_k^{SD'} & \text{if } \mathrm{mod}(k,4) = 1 \text{or } 2 \\ \alpha_k^{VY'} & \text{otherwise} \end{cases} \tag{18}$$

is given. We refer to the project gradient method with (18) as the PSY method. Obviously, the methods PSD and PSY are special cases of the Algorithm 2.

Here we need to point out that $\alpha_k^{VY'}$ may not have the similar property of (8) and (9) in two-dimensions for (1), but the numerical results will show that the PSY method is comparable to the PBB method.

3 Global convergence analysis

In this section, we analyze the global convergence of Algorithm 1 and Algorithm 2. In what follows, we denote the set of solutions to (1) by

$\boldsymbol{X}^* = \{\boldsymbol{x}^* \in \Omega \mid f(x) \geqslant f(x^*), \text{for all } x \in \Omega\}$, and the optimal value of (1) by f^*.

Theorem 3.1 If Algorithm 1 does not terminate in a finite number of iterations, there must exist a strictly monotone decreasing infinite subsequence in $\{f(\boldsymbol{x}_k)\}$.

Proof Let p be the length of the $\{f(\boldsymbol{x}_{k_i^s})\}$. If $p = +\infty$, $\{f(\boldsymbol{x}_{k_i^s})\}$ is the subsequence satisfying the requirement.

If $p < +\infty$ is finite, there exists a const h_0 which satisfies $k_{h_0}^r = k_p^s$. Without loss of generality, let $h_0 = 0$. Since no smaller function value is found after iteration k_p^s, the reference function f_h^r will be updated every $L+1$ iterations and the length of $\{f_h^r\}$ is infinite. It is easy to see that

$$f_{h_0+q}^r = f_q^r = \max_{0\leqslant j\leqslant L}\{f(\boldsymbol{x}_{k_q^r+j})\}, q = 1,2,\cdots \tag{19}$$

On the other hand,

$$f_q^r > f(\boldsymbol{x}_{k_{q+1}^r+j}),j = 1,2,\cdots,L. \tag{20}$$

It follows from $k_{q+1}^r = k_q^r + L$, (19) and (20) that

$$f_q^r \geqslant \max_{0\leqslant j\leqslant L}\{f(\boldsymbol{x}_{k_{q+1}^r+j})\} \tag{21}$$

$$= f_{q+1}^r. \tag{22}$$

Let $q = q + 1$ in (20), then $f_{q+1}^r > f(\boldsymbol{x}_{k_{q+2}^r+j}),j = 1,2,\cdots,L$. Combining (22), we conclude that

$$f_q^r > f(\boldsymbol{x}_{k_{q+2}^r+j}),j = 1,2,\cdots,L. \tag{23}$$

Let $j = L$ in (20), then

$$f_q^r > f(\boldsymbol{x}_{k_{q+1}^r+L}) = f(\boldsymbol{x}_{k_{q+2}^r}) \tag{24}$$

It follows from (23) and (24) that

$$f_q^r > \max_{0\leqslant j\leqslant L}\{f(\boldsymbol{x}_{k_{q+2}^r+j})\} = f_{q+2}^r$$

which implies that either even or odd subsequence of $\{f_q^r\}$ is strictly monotone decreasing. In conclusion, the theorem is true. □

We give some properties of $\boldsymbol{P}(\boldsymbol{x})$ and $\boldsymbol{d}(\boldsymbol{x})$ which will be used in the following theorems, where $\boldsymbol{P}(\boldsymbol{x})$ and $\boldsymbol{d}(\boldsymbol{x})$ are defined as (11) and (13) respectively.

Proposition 3.1 (Properties of $\boldsymbol{P}(\boldsymbol{x})$ and $\boldsymbol{d}(\boldsymbol{x})$)

P1. $\forall y \in \mathbf{R}^n$ and $\boldsymbol{x} \in \Omega, (\boldsymbol{P}(\boldsymbol{y}) - \boldsymbol{y})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{P}(\boldsymbol{y})) \geqslant 0$.

P2. $\forall \boldsymbol{x} \in \Omega, \boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}) \leqslant -\boldsymbol{d}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x})$.

P3. $\forall \boldsymbol{x}^{**} \in \Omega, \boldsymbol{d}(\boldsymbol{x}^{**}) = \boldsymbol{0}$ if and only if $\boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{**}) \geqslant 0$ for all $\boldsymbol{x} \in \Omega$.

P4. $\forall \boldsymbol{x} \in \mathbf{R}^n$ and $\boldsymbol{x}^{**} \in \mathbf{X}^{**} = \{\boldsymbol{y} \in \Omega \mid \boldsymbol{d}(\boldsymbol{y}) = \boldsymbol{0}\}$, we have

$$\|\boldsymbol{x} - \boldsymbol{x}^{**}\|_2 \leqslant \frac{1 + \lambda_n}{\lambda_1}\|\boldsymbol{d}(\boldsymbol{x})\|_2.$$

Proof Firstly, P1 is the optimal condition of (11) (see Ref. [11]).

From P1, we can easily obtain P2 by replacing $\boldsymbol{y}$ with $\boldsymbol{x} - \boldsymbol{g}(\boldsymbol{x})$ in P1.

P3 is proved as follows.

From the definition of $\boldsymbol{d}(\boldsymbol{x})$, we know that if $\boldsymbol{d}(\boldsymbol{x}^{**}) = \boldsymbol{0}$, $\boldsymbol{x}^{**} = \boldsymbol{P}(\boldsymbol{x}^{**} - \boldsymbol{g}(\boldsymbol{x}^{**}))$. If $\boldsymbol{y}$ is replaced by $\boldsymbol{x}^{**} - \boldsymbol{g}(\boldsymbol{x}^{**})$ in P1, P1 yields

$$\boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{**}) \geqslant 0 \ \forall \boldsymbol{x} \in \Omega. \tag{25}$$

Conversely, due to $\boldsymbol{P}(\boldsymbol{x}^{**} - \boldsymbol{g}(\boldsymbol{x}^{**})) \in \Omega$, we have

$$\boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}^{**}) \geqslant 0.$$

From above inequality and P2, we obtain

$$0 \leqslant -\boldsymbol{d}(\boldsymbol{x}^{**})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}^{**}) \leqslant 0$$

which means $\boldsymbol{d}(\boldsymbol{x}^{**}) = 0$.

Finally, we consider P4. In P1, $\boldsymbol{y}$ is replaced with $\boldsymbol{x} - \boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{x}$ is replaced with $\boldsymbol{x}^{**}$, then we have inequality as follows,

$$(\boldsymbol{P}(\boldsymbol{x} - \boldsymbol{g}(\boldsymbol{x})) - \boldsymbol{x} + \boldsymbol{g}(\boldsymbol{x}))^{\mathrm{T}} (\boldsymbol{x}^{**} - \boldsymbol{P}(\boldsymbol{x} - \boldsymbol{g}(\boldsymbol{x}))) \geqslant 0. \tag{26}$$

In fact, by the definition of $\boldsymbol{d}(\boldsymbol{x})$ as (13), we know that (26) is equivalent to

$$(\boldsymbol{d}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x}))^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x} - \boldsymbol{d}(\boldsymbol{x})) + \boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x}) \geqslant \boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x}). \tag{27}$$

Rearranging (27), we have

$$\begin{aligned}
&\boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}) \\
&\geqslant \boldsymbol{d}(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{**}) + \|\boldsymbol{d}(\boldsymbol{x})\|_2 + (\boldsymbol{g}(\boldsymbol{x}^{**}) - \boldsymbol{g}(\boldsymbol{x}))^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x}) \\
&\geqslant \boldsymbol{d}(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{**}) + \|\boldsymbol{d}(\boldsymbol{x})\|_2 + (\boldsymbol{x}^{**} - \boldsymbol{x})^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}^{**} - \boldsymbol{x}) \\
&\geqslant \boldsymbol{d}(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{**}) + \lambda_1\|\boldsymbol{x} - \boldsymbol{x}^{**}\|_2^2.
\end{aligned} \tag{28}$$

On the other hand,

$$\begin{aligned}
&\boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}) \\
&= \boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x}) + (\boldsymbol{g}(\boldsymbol{x}^{**}) - \boldsymbol{g}(\boldsymbol{x}))^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}) \\
&= \boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x} - \boldsymbol{d}(\boldsymbol{x})) + (\boldsymbol{x}^{**} - \boldsymbol{x})^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}(\boldsymbol{x}) \\
&= \boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{P}(\boldsymbol{x} - \boldsymbol{g}(\boldsymbol{x}))) + (\boldsymbol{x}^{**} - \boldsymbol{x})^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}(\boldsymbol{x}).
\end{aligned} \tag{29}$$

Since $\boldsymbol{P}(\boldsymbol{x} - \boldsymbol{g}(\boldsymbol{x})) \in \Omega$ and from P3, then we have

$$\boldsymbol{g}(\boldsymbol{x}^{**})^{\mathrm{T}}(\boldsymbol{x}^{**} - \boldsymbol{x}) - \boldsymbol{g}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}) \leqslant (\boldsymbol{x}^{**} - \boldsymbol{x})^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}(\boldsymbol{x}). \tag{30}$$

Combining (28) and (30), we obtain that

$$\boldsymbol{d}(\boldsymbol{x})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^{**}) + \lambda_1\|\boldsymbol{x} - \boldsymbol{x}^{**}\|_2^2 \leqslant (\boldsymbol{x}^{**} - \boldsymbol{x})^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}(\boldsymbol{x}). \tag{31}$$

Rearranging (31), we have

$$\begin{aligned}
\|\boldsymbol{x} - \boldsymbol{x}^{**}\|_2 &\leqslant \frac{(\boldsymbol{x}^{**} - \boldsymbol{x})^{\mathrm{T}}(\boldsymbol{I} + \boldsymbol{Q})\boldsymbol{d}(\boldsymbol{x})}{\lambda_1\|\boldsymbol{x} - \boldsymbol{x}^{**}\|_2} \\
&\leqslant \frac{1 + \lambda_n}{\lambda_1}\|\boldsymbol{d}(\boldsymbol{x})\|_2.
\end{aligned} \tag{32}$$

This completes the proof. □

Now, we give the following theorem about the equivalent optimal conditions of (1).

Theorem 3.2 Suppose $\boldsymbol{x}^*$ is a feasible point of (1), the following results are equivalent:

E1. $\boldsymbol{x}^* \in \mathbf{X}^*$.

E2. $\boldsymbol{d}(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}^*) = 0$.

E3. $\boldsymbol{d}(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}(\boldsymbol{x}^*) = 0$.

E4. $\boldsymbol{d}(\boldsymbol{x}^*) = \boldsymbol{0}$.

Proof E1⇔ E4:

From P3, $\boldsymbol{d}(\boldsymbol{x}^*) = 0$ is equivalent to

$$\boldsymbol{g}(\boldsymbol{x}^*)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}^*) \geqslant 0, \forall \boldsymbol{x} \in \Omega.$$

The above inequality implies that $\boldsymbol{x}^*$ is the solution of the following linear programming

$$\begin{aligned} &\min \boldsymbol{g}(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{x} \\ &\text{s.t. } \boldsymbol{x} \in \Omega. \end{aligned} \tag{33}$$

Obviously, (1) and (37) have the same KKT system, so $\boldsymbol{x}^* \in \boldsymbol{X}^*$.

Since E1⇔ E4, $\boldsymbol{d}(\boldsymbol{x}^*) = \boldsymbol{0}$. Then proposition E1⇒ E2 and E1⇒ E3 can be easily obtained. From E2 and P2, we have that $\boldsymbol{d}(\boldsymbol{x}^*) = \boldsymbol{0}$, which means E1 is established.

E3⇒E1:

Assume E1 is not established, that is, $\boldsymbol{x}^* \notin \boldsymbol{X}^*$. From P2 and E2, we have $\boldsymbol{d}(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{g}(\boldsymbol{x}^*) < 0$. As $f(x)$ is quadratic and $\boldsymbol{d}(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}(\boldsymbol{x}^*) = 0$, we have $f(\boldsymbol{x}^* + \alpha\boldsymbol{d}(\boldsymbol{x}^*)) = f(\boldsymbol{x}^*) + \alpha\boldsymbol{g}(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{d}(\boldsymbol{x}^*)$. Let $\alpha \to +\infty$, then $f(\boldsymbol{x}^* + \alpha\boldsymbol{d}(\boldsymbol{x}^*)) \to -\infty$ which contradicts that (1) is bounded below. Thus, $\boldsymbol{x}^*$ belongs to $\boldsymbol{X}^*$. □

According to Theorem 3.2 and P4, we can see that $\boldsymbol{X}^* = \boldsymbol{X}^{**}$. Then, for all feasible points $\boldsymbol{x} \notin \boldsymbol{X}^*$,

$$\boldsymbol{d}(\boldsymbol{x})^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}(\boldsymbol{x}) > 0.$$

Thus, the formula (15) and (18) are well defined unless $\boldsymbol{x}_k$ is a optimal point of (1).

Finally, we present the global convergence of Algorithm 2.

Theorem 3.3 If Algorithm 2 does not terminate in a finite number of iterations and the sequence $\{\rho_k\}$ has an accumulation point $\tilde{\rho} \in (0, 2)$, then the sequence $\{f(\boldsymbol{x}_k)\}$ generated by Algorithm 2 converges to f^*.

Proof Observe that for all k,

$$\phi_k(\rho) = f(\boldsymbol{x}_k + \rho\alpha_k^{SD'}\boldsymbol{d}_k)$$

is a quadratic convex function of the variable ρ. From the definition of $\alpha_k^{SD'}$, we get that $\phi_k(\rho)$ attains global minimum when $\rho = 1$. Hence by symmetry of the quadratic convex function $\phi_k(\rho)$, $\phi_k(0) = \phi_k(2)$ and for any $\rho \in [0,2]$, $\phi_k(\rho) \leqslant \phi_k(0)$. Therefore

$$\phi_k(\rho_k) = f(\boldsymbol{x}_{k+1}) \leqslant f(\boldsymbol{x}_k),$$

for all k. The above inequality implies that $\{f(\boldsymbol{x}_k)\}$ is a monotonically decreasing sequence. Furthermore, $f(\boldsymbol{x}_k)$ is bounded below, so $\{f(\boldsymbol{x}_k)\}$ a convergent sequence,

$$\lim_{k\to+\infty}(f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})) = 0. \tag{34}$$

Because $\tilde{\rho}$ is an accumulation point of $\{\rho_k\}$ and $\tilde{\rho} \in (0, 2)$, there exists a const $\beta \in (0, 1)$ and subsequence $\rho_{k_j} \in [\beta, 2-\beta]$. Using the properties of ϕ_k we have

$$\begin{aligned} f(\boldsymbol{x}_{k_j}) - f(\boldsymbol{x}_{k_{j+1}}) &= \phi_{k_j}(0) - \phi_{k_j}(\rho_{k_j}) \\ &\geqslant \phi_{k_j}(0) - \phi_{k_j}(\beta) \\ &= -\beta\phi'_{k_j}(0) - \frac{\beta^2}{2}\phi''_{k_j}(0) \\ &= \beta\alpha_{k_j}^{SD'}\boldsymbol{g}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j} - \frac{\beta^2}{2}\alpha_{k_j}^{SD'2}\boldsymbol{d}_{k_j}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}_{k_j} \\ &= \frac{\beta(2-\beta)}{2}\frac{(\boldsymbol{g}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j})^2}{\boldsymbol{d}_{k_j}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}_{k_j}} \end{aligned}$$

Since $\boldsymbol{g}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j} \leqslant -\boldsymbol{d}_{k_j}^{\mathrm{T}} \leqslant 0$, then $|\boldsymbol{g}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j}| \geqslant |\boldsymbol{d}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j}|$. Thus, we have that

$$\begin{aligned} f(\boldsymbol{x}_{k_j}) - f(\boldsymbol{x}_{k_j+1}) &\geqslant \frac{\beta(2-\beta)}{2}\frac{(\boldsymbol{d}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j})^2}{\boldsymbol{d}_{k_j}^{\mathrm{T}}\boldsymbol{Q}\boldsymbol{d}_{k_j}} \\ &\geqslant \frac{\beta(2-\beta)\boldsymbol{d}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j}}{2\lambda_{\max}(\boldsymbol{Q})}. \end{aligned} \tag{35}$$

Let $j \to +\infty$ in the (35), then

$$\begin{aligned} 0 &= \lim_{j\to+\infty}(f(\boldsymbol{x}_{k_j}) - f(\boldsymbol{x}_{k_{j+1}})) \\ &\geqslant \lim_{j\to+\infty}\frac{\beta(2-\beta)\boldsymbol{d}_{k_j}^{\mathrm{T}}\boldsymbol{d}_{k_j}}{2\lambda_{\max}(\boldsymbol{Q})} \geqslant 0. \end{aligned}$$

From the above inequality, we conclude that $\boldsymbol{d}_{k_j} \to \boldsymbol{0}$, $j \to +\infty$.

It follows from P4 that $\boldsymbol{x}_{k_j}$ converges to some optimal point $\boldsymbol{x}^* \in \boldsymbol{X}^*$. Since $f(\boldsymbol{x})$ is continuous, we have

$$f(\boldsymbol{x}_{k_j}) \to f^*.$$

As $\{f(\boldsymbol{x}_k)\}$ is decreasing and bounded from below, the whole sequence $\{f(\boldsymbol{x}_k)\}$ converges to f^*. □

4 Numerical experiments

In this section, we report the numerical results of the project gradient methods proposed in this paper. The test problems are designed based on nine parameters n, m, ncond, l_x, u_x, l_A, u_A, l_c, u_c. In particular, we denote $\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{\mathrm{T}}$, where

$$\boldsymbol{P} = \prod_{i=1}^{3}(\boldsymbol{I} - 2\boldsymbol{\omega}_i\boldsymbol{\omega}_i^{\mathrm{T}})$$

and $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$, $\boldsymbol{\omega}_3$ are random unit vectors, and $\boldsymbol{\Lambda}$ is diagonal matrix whose i-th component is defined by

$$\log\lambda_i = \frac{i-1}{n-1}n\text{cond}, \ i = 1, \cdots, n.$$

The initial point $\boldsymbol{x}_0$, matrix $\boldsymbol{A}$ and the linear term $\boldsymbol{c}$ are generated by Matlab function rand with entries in $[l_x, u_x]$, $[l_A, u_A]$, $[l_c, u_c]$ respectively. $\boldsymbol{b}$ is built as $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_0$. In our tests, we have fixed $(l_x, u_x, l_A, u_A, l_c, u_c) = (-5, 5, -10, 10, -10, 10)$. m, n, and ncond are positive integers chosen randomly in the interval $[50, 800]$, $[1\,000, 2\,000]$ and $[2, 6]$ by the function *randint* respectively. As the stopping criterion we use

$$\| \boldsymbol{d}_k \|_\infty \leqslant 10^{-4} \| \boldsymbol{d}_0 \|_\infty.$$

All tests are conducted on a Windows XP professional computer (Pentium R, 2.79 GHZ) with Matlab 7.10.

Firstly, we generate 100 random test problems to test the PBB method and the MPBB method with $M = 1, 2, \cdots, 15$. The numerical results are reported in Table 2, where Aiter and Asec denote the average number of iterationsand time required with regard to these 100 test problems respectively.

From Table 2, we see that the PBB method performs worse as M ($M > 6$) increases with rare exception, so does the MPBB method ($M > 2$). On the whole, the iterations and time of these two methods have a tendency to ascend with the growth of M. At the same time, we suggest $M = 6$ for the PBB method and $M = 2$ for the MPBB method. Also we conclude that the MPBB method performs much better than PBB method.

Table 2　Testing the PBB and MPBB methods as M increases

M	PBB		MPBB	
	Aiter	Asec	Aiter	Asec
1	309.96	3.85	176.37	2.58
2	282.61	3.61	**138.73**	**2.14**
3	277.23	3.57	145.46	2.25
4	268.54	3.49	154.15	2.30
5	277.43	3.57	149.62	2.27
6	**268.04**	**3.47**	154.87	2.29
7	277.60	3.56	155.44	2.23
8	282.33	3.61	156.89	2.33
9	278.53	3.57	156.09	2.33
10	282.42	3.58	157.24	2.32
11	285.95	3.64	166.88	2.44
12	291.98	3.72	170.28	2.49
13	297.73	3.76	172.13	2.45
14	303.85	3.80	176.87	2.56
15	295.85	3.73	177.33	2.52

Table 3 lists the numbers of iterations and time required by the PSD, PBB ($M = 6$), MPBB ($M = 2$) and PSY methods for 15 random problems. From this table, we can clearly see that the PSD method is significantly slower than the other three methods. Although, the PSY is slightly worse than the MPBB method, the PSY method performs better than PBB method. What's more, the MPBB method proposed

Table 3　Comparison of numerical results for random test problems among the methods

Problem			PSD		PBB		MPBB		PSY	
ncond	m	n	Iter	Sec	Iter	Sec	Iter	Sec	Iter	Sec
2	661	1 906	145	3.73	47	2.04	33	2.15	43	2.58
5	793	1 833	5 553	84.23	402	7.84	201	5.41	232	7.73
2	258	1 805	251	4.11	59	1.31	52	1.28	54	1.88
3	242	1 288	1 195	8.88	118	1.09	121	1.22	119	1.58
2	383	1 015	129	0.94	41	0.48	40	0.59	42	0.70
4	676	1 334	1 035	9.25	137	2.08	93	1.78	126	2.70
2	716	1 686	117	3.03	41	2.01	32	2.05	46	2.30
2	226	1 802	241	3.63	61	1.48	52	1.30	62	1.92
4	418	1 388	3 107	25.16	207	2.29	153	1.84	179	2.75
6	322	1 107	22 329	123.30	810	4.77	262	1.64	418	3.75
3	470	1 914	1 005	15.17	111	2.52	118	2.92	147	4.42
5	78	1 910	25 911	469	548	9.81	192	2.95	208	4.72
6	489	1 906	26 883	486.67	1 090	21.93	253	5.97	227	7.25
2	638	1 128	69	1.39	40	1.18	26	1.14	25	1.16
5	689	1 777	6 569	102.78	357	7.60	194	5.28	255	8.06
5	399	1 511	14 087	161.25	519	5.93	237	3.37	300	5.92
4	766	1 060	225	2.38	74	1.58	49	1.43	50	1.54
	average		6 403	88.53	274.23	4.47	124	2.49	149	3.59

by us is not only to guarantee the convergence of PBB method but also to perform much better than the latter one.

5 Conclusions

In this paper, we proposed two methods, MPBB and PSY, for quadratic programming with linear equality constraints (QPLE) based on the efficient step length selections and have established their convergence under mild assumptions. Our numerical results demonstrate that the new project gradient methods with these step selection rules have superiority over the methods with classical step length.

References

[1] Barzilai J, Borwein J M. Two-point step size gradient methods [J]. IMA Journal of Numerical Analysis, 1988, 8(1): 141-148.

[2] Zhou B, Gao L, Dai Y H. Gradient methods with adaptive step-sizes[J]. Computational Optimization and Applications, 2006, 35(1): 69-86.

[3] Raydan M. On the Barzilai and Borwein choice of steplength for the gradient method [J]. IMA Journal of Numerical Analysis, 1993, 13(3): 321-326.

[4] Dai Y H, Liao L Z. R-linear convergence of the Barzilai and Borwein gradient method [J]. IMA Journal of Numerical Analysis, 2002, 22(1): 1-10.

[5] Dai Y H, Fletcher R. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds[J]. Mathematical Programming, 2006, 106(3): 403-421.

[6] Raydan M. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem[J]. SIAM Journal on Optimization, 1997, 7(1): 26-33.

[7] Yuan Y. Gradient methods for large scale convex quadratic functions [M] // Optimization and Regularization for Computational Inverse Problems and Applications. Berlin: Springer Heidelberg, 2010: 141-155.

[8] Yuan Y. A new stepsize for the steepest descent method[J]. Journal of Computational Mathematics, 2006: 149-156.

[9] Dai Y, Yuan Y X. Analysis of monotone gradient methods [J]. Journal of Industrial and Management Optimization, 2005, 1(2): 181.

[10] Dai Y H, Fletcher R. Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming [J]. Numerische Mathematik, 2005, 100(1): 21-47.

[11] Hager W W, Zhang H. A new active set algorithm for box constrained optimization[J]. SIAM Journal on Optimization, 2006, 17(2): 526-557.

[12] Zhou B, Gao L, Dai Y. Monotone projected gradient methods for largescale box-constrained quadratic programming [J]. Science in China Series A, 2006, 49(5): 688-702.

[13] De Asmundis R, di Serafino D, Riccio F, et al. On spectral properties of steepest descent methods[J]. IMA Journal of Numerical Analysis, 2013: drs056.

[14] Zanni L, Serafini T, Zanghirati G. Parallel software for training large scale support vector machines on multiprocessor systems[J]. Journal of Machine Learning Research, 2006, 7(7): 1 467-1 492.

[15] Serafini T, Zanghirati G, Zanni L. Parallel decomposition approaches for training support vector machines [J]. Advances in Parallel Computing, 2004, 13: 259-266.

[16] Zanghirati G, Zanni L. A parallel solver for large quadratic programs in training support vector machines [J]. Parallel computing, 2003, 29(4): 535-551.

[17] Grippo L, Lampariello F, Lucidi S. A nonmonotone line search technique for Newton's method[J]. SIAM Journal on Numerical Analysis, 1986, 23(4): 707-716.