J. Appl. Numer. Optim. 7 (2025), No. 1, pp. 85-95
Available online at http://jano.biemdas.com
https://doi.org/10.23952/jano.7.2025.1.06

SOLVING A LARGE-SCALE LEAST SQUARES PROBLEM WITH A SIMPLEX CONSTRAINT BY A TWO-STAGE ALGORITHM

CAN XIANG[1], CHENGJING WANG[1,*], PIQIN SHI[1], PEIPEI TANG[2], AIMIN XU[3]

[1]*School of Mathematics, Southwest Jiaotong University, Chengdu 611731, China*
[2]*School of Computer and Computing Science, Hangzhou City University, Hangzhou 310015, China*
[3]*Institute of Mathematics, Zhejiang Wanli University, Ningbo 315100, China*

Abstract. In this paper, we introduce a two-stage algorithm for a large-scale least squares problem with a simplex constraint. The basic idea is to use the accelerated proximal gradient (APG) method to generate a relatively good precision solution for the primal problem, and take this point as the initial point of the second stage. In the second stage, we use the semismooth Newton (SsN) based on the augmented Lagrangian method (ALM) to solve the dual problem, that is, the ALM is used to solve the dual problem, and the SsN method is used to solve the subproblem. It is worth mentioning that, due to the piecewise linearity of the subproblem, the accuracy of the SsN method decreases relatively slowly in the early stage, and the APG method can effectively reduce the time required in this stage. In addition, we also prove the convergence and convergence rate of the proposed algorithm under certain conditions. Numerical experiments demonstrate that our algorithm outperforms the current state-of-the-art algorithms for the least squares problem with a simplex constraint.

Keywords. Augmented Lagrangian method; Accelerated proximal gradient method; Least squares problem; Simplex constraint; Semismooth Newton method.

2020 Mathematics Subject Classification. 90C20, 90C30.

1. Introduction

In this paper, we focus on the following problem

$$\min_{x\in H} \frac{1}{2}\|Ax-\tilde{b}\|^2, \tag{P0}$$

where $H := \{x \in \mathbb{R}^n \mid \sum_{i=1}^{n} x_i = 1, x_i \geq 0\}$ is a simplex, $A \in \mathbb{R}^{m\times n}$, and $\tilde{b} \in \mathbb{R}^n$. Let $Q = A^T A$ and $b = -A^T\tilde{b}$. We can reformulate (P0) as

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\langle x, Qx\rangle + \langle b, x\rangle + \delta_H(x), \tag{P}$$

[*]Corresponding author.
E-mail address: renascencewang@hotmail.com (C. Wang).

Received 31 August 2024; Accepted 10 October 2024; Published online 20 March 2025.

where $\delta_H(x)$ is an indicator function, i.e., if $x \in H$, then $\delta_H(x) = 0$, otherwise $\delta_H(x) = \infty$. The dual problem of (P) is

$$\min_{u,v \in \mathbb{R}^n} \frac{1}{2}\langle u, Qu\rangle + \delta_H^*(v) \tag{D}$$

$$\text{s.t.}\, Qu + v + b = 0, u \in \operatorname{Ran}(Q),$$

where $\delta_H^*(v)$ is the conjugate function of $\delta_H(x)$. Here we artificially restrict u in the range space of Q to guarantee that the subproblem in the subsequent algorithm has a unique solution.

The least squares problem with a simplex constraint has significant real applications. For instance, the unmixing problem, a crucial task in the hyperspectral image decomposition [1], involves the abundance as the basic unit. This abundance must be non-negative and sum to 1, which is highly consistent with the model that we propose. In the financial domain, the problem of constructing a constrained global minimum variance portfolio [2] can be framed as a least squares problem with a simplex constraint. This model can be utilized to optimize a portfolio, ensuring that the weights of each asset are non-negative and sum to 1, thereby maximizing the portfolio's return or minimizing its risk. Furthermore, in signal processing, particularly for blind source separation techniques [3], this model is primarily employed to separate independent source signals from mixed signals, ensuring that the separated signals are non-negative and satisfy certain constraint conditions.

For problem (P), Boyd et al. [4] and Condat [5] proposed the alternating direction method of multipliers and the Douglas-Rachford algorithm, respectively. However, when the problem becomes large and ill-conditioned, the costs of these algorithms are very high, even unable to converge within a certain number of iterations. In handling large-scale optimization problems, the semismooth Newton (SsN) based on the augmented Lagrangian method (ALM) stands as an excellent approach. However, owing to the piecewise linear nature of the objective functions within the inner subproblems, we observe that, when the iteration point is not proximate to the solution point, the accuracy hovers at low levels until the last iteration. That is, during the initial stage, the computationally expensive Newton method is not the most suitable choice. Consequently, we propose to employ a simple algorithm to provide a warm start for the ALM.

In this paper, we introduce a two-stage algorithm. In the first stage, we utilize the accelerated proximal gradient (APG) method to generate an initial point. Subsequently, in the second stage, we adopt the ALM, where the inner subproblems are solved using the SsN method. Notably, we fully exploit the sparsity structure of the Hessian matrices associated with the inner subproblems, significantly reducing the computational cost. The results from our numerical experiments demonstrate the efficiency and effectiveness of our proposed algorithm. The rest of this paper is organized as follows. In Section 2, we present some preliminaries. In Section 3, we describe the details of the algorithm. In Section 4, we analyze the convergence of the algorithm. In Section 5, we present some numerical results. In Section 6, we give the conclusion.

1.1. **Additional notations.** Let $\mathbb{R}^n$ be the n-dimensional Euclidean space, and let I_n be the n-dimensional identity matrix. For a convex function $f : \mathbb{R}^n \to (-\infty, +\infty]$, its conjugate function is $f^*(x) := \sup_y\{\langle x, y\rangle - f(y)\}$. In addition, for a given closed set S and a given vector x, we denote the distance from x to S by $\operatorname{dist}(x, S)$, and the projection of x onto S by $\Pi_S(x)$. For a given matrix $P \in \mathbb{R}^{n\times n}$, we use $P^\dagger$ to represent the Moore-Penrose pseudo-inverse of P.

2. Preliminaries

In this section, we present some basic preliminaries. More details about these contents, one can refer to [6]. For a function $f:\mathbb{R}^n \to [-\infty,+\infty]$, we define its effective domain as $\mathrm{dom}(f) := \{x\in\mathbb{R}^n | f(x) < +\infty\}$. We call f a proper function if $\mathrm{dom}(f)$ is not empty and for any $x \in \mathrm{dom}(f)$, $f(x) > -\infty$. We say that $f:\mathbb{R}^n \to [-\infty,+\infty]$ is lower semicontinuous at $\hat{x}$ if $\lim_{x\to\hat{x}}\inf f(x) = f(\hat{x})$, and if f is lower semicontinuous at every point, then we call it a lower semicontinuous function. For a convex function $f:\mathbb{R}^n \to (-\infty,+\infty]$, for any $\hat{x}\in\mathbb{R}^n$, its subdifferential is defined as

$$\partial f(\hat{x}) := \{z\in\mathbb{R}^n \,|\, \forall x\in\mathrm{dom}(f), f(x) \geq f(\hat{x}) + \langle z, x-\hat{x}\rangle\}.$$

Next, let $\mathscr{X}$ and $\mathscr{Y}$ be finite-dimensional Euclidean spaces, and let $T:\mathscr{X} \rightrightarrows \mathscr{Y}$ be a multi-valued function. We call T a monotone operator if, for any $x, x' \in \mathscr{X}$ and $y\in T(x)$, $y'\in T(x')$, $\langle x-x', y-y'\rangle \geq 0$. Especially, we call T a maximal monotone operator if the graph of T

$$\mathrm{graph}(T) := \{(x,y)\in\mathscr{X}\times\mathscr{Y} \,|\, y\in T(x)\}$$

cannot be covered by the graph of any other monotone operator $T':\mathscr{X} \rightrightarrows \mathscr{Y}$.

3. Algorithms

In this section, we address the details of the algorithms. For problem (D), its Lagrangian function is defined by

$$l(u,v;x) = \frac{1}{2}\langle u, Qu\rangle + \delta_H^*(v) - \langle x, Qu+v+b\rangle. \tag{3.1}$$

The Karush-Kuhn-Tucker (KKT) condition is

$$\begin{cases} Qu - Qx = 0 \\ \partial\delta_H^*(v) - x = 0 \\ Qu + v + b = 0 \end{cases}. \tag{3.2}$$

In addition, we can also derive the optimality condition without the dual variables

$$0 \in Qx + b + \partial\delta_H(x),$$

which is equivalent to

$$G_f(x) := x - \Pi_H(x - Qx - b) = 0.$$

For problem (D) and a given parameter $\sigma > 0$, the augmented Lagrangian function for is as follows

$$L_\sigma(u,v;x) := \frac{1}{2}\langle u, Qu\rangle + \delta_H^*(v) + \frac{\sigma}{2}\|Qu+v+b-\frac{1}{\sigma}x\|^2 - \frac{1}{2\sigma}\|x\|^2. \tag{3.3}$$

For problem (D), we adopt a two-stage algorithm. In the first stage, we use the APG method to generate an initial point, and in the second stage, we use the ALM. Firstly, we provide the details of the APG method.

Algorithm 1 : The APG method for the problem (P)

Let $\varepsilon_1 > 0$ and $L = \lambda_{\max}(Q)$. Input an initial point x^0. Take $y^1 = x^0$ and $t_1 = 1$. For $k = 1, 2, \cdots$, iterate as follows:

Step 1. Compute

$$\hat{b} = y^k - \frac{1}{L}A^T(Ay^k - b),$$

$$x^k = \Pi_H(\hat{b}),$$

where the projection operator can be computed according to [7].
Step 2. Compute

$$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}.$$

Step 3. Update

$$y^{k+1} = x^k + \frac{t_k - 1}{t_{k+1}}(x^k - x^{k-1}).$$

Step 4. If $\frac{\|G_f(y^{k+1})\|}{1+\|y^{k+1}\|} \leq \varepsilon_1$, stop; else, go to the Step 1.

After describing the details of the APG method, we move to the second stage.

Algorithm 2 : The ALM for the problem (D)

Let $\varepsilon_2 > 0$ and $\sigma_0 > 0$. Take the result obtained by Algorithm 1 as x^0. For $k = 1, 2, \cdots$, iterate as follows:
Step 1. Compute

$$(u^{k+1}, v^{k+1}) \approx \underset{u\in \text{Ran}(Q), v\in\mathbb{R}^n}{\text{argmin}} \ \Phi^k(u,v) := L_\sigma(u,v;x^k). \tag{3.4}$$

Step 2. Compute

$$x^{k+1} = x^k - \sigma_k(Qu^{k+1} + v^{k+1} + b).$$

Step 3. If $\frac{\|G_f(x^{k+1})\|}{1+\|x^{k+1}\|} \leq \varepsilon_2$, stop; else $\sigma_{k+1} = 2\sigma_k$, end.

In Algorithm 2, we apply the SsN method to solve subproblem (3.4). We define

$$\begin{aligned} \varphi^k(u) &:= \inf_{v\in\mathbb{R}^n} \Phi^k(u,v) \\ &= \frac{1}{2\sigma_k}\|f(u)\|^2 + \frac{1}{2}\langle u, Qu\rangle - \frac{1}{2\sigma_k}\|x^k\|^2 \\ &- \frac{1}{2\sigma_k}\|f(u) - \Pi_H(f(u))\|^2, \end{aligned}$$

where $f(u) = x^k - \sigma_k(Qu + b)$. Then we can obtain (u^{k+1}, v^{k+1}) by

$$\begin{aligned} u^{k+1} &\approx \underset{u\in\text{Ran(Q)}}{\text{argmin}} \ \varphi^k(u), \\ v^{k+1} &= (f(u^{k+1}) - \Pi_H(f(u^{k+1})))/\sigma_k. \end{aligned} \tag{3.5}$$

Since $\varphi^k(u)$ is convex, we can obtain the optimal solution of (3.5) by solving the following equation

$$\nabla\varphi^k(u) = Qu - Q\Pi_H(f(u)) = 0.$$

To ensure that subproblem (3.4) has a unique solution, we restrict u to the range space $\operatorname{Ran}(Q)$. That is, we solve the following problem

$$\nabla\varphi^k(u) = Qu - Q\Pi_H(f(u)) = 0, u \in \operatorname{Ran}(Q).$$

Letting u be a given point, we define a multivalued mapping $\hat{\partial}^2\varphi^k(u) : \mathbb{R}^n \to \mathbb{S}^n$ as

$$\hat{\partial}^2\varphi^k(u) := Q + \sigma_k QJ(f(u))Q,$$

where $J(f(u))$ is the generalized Jacobian of Π_H at $f(u)$.

Next, we provide the details about the computation of $J(f(u))$. More details about $J(f(u))$, one may refer to [8]. For the sake of simplicity, we rewrite the projection problem as

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \frac{1}{2}\|x - f(u)\|^2 \\ &\text{s.t.} a^T x = 1, Bx \le z, \end{aligned}$$

where $a = [1,\cdots,1]^T \in \mathbb{R}^n$, $B = -I_n$, and $z = [0,\cdots,0]^T \in \mathbb{R}^n$. Then we denote

$$\mathscr{I}(f(u)) := \{i \,|\, B_i\Pi_H(f(u)) = z_i, i = 1,\cdots,n\},$$

where B_i is the i-th row of the matrix B, and

$$N = I_n - \begin{bmatrix} B_{\mathscr{I}(f(u))} \\ a^T \end{bmatrix}^T \begin{bmatrix} B_{\mathscr{I}(f(u))}B^T_{\mathscr{I}(f(u))} & B_{\mathscr{I}(f(u))}a \\ a^T B^T_{\mathscr{I}(f(u))} & a^T a \end{bmatrix}^\dagger \begin{bmatrix} B_{\mathscr{I}(f(u))} \\ a^T \end{bmatrix},$$

where $B_{\mathscr{I}(f(u))}$ consists of the rows of B indexed by $\mathscr{I}(f(u))$. According to [9], we have $N \in J(f(u))$. From the definition of $B_{\mathscr{I}(f(u))}$, we have $B_{\mathscr{I}(f(u))}B^T_{\mathscr{I}(f(u))} = I_k$, where k represents the number of elements in $\mathscr{I}(f(u))$. This is because the element $(B_{\mathscr{I}(f(u))}B^T_{\mathscr{I}(f(u))})_{ij}$ is the result of the inner product between i-th row and the j-th column of $B_{\mathscr{I}(f(u))}$. If $i = j$, $(B_{\mathscr{I}(f(u))}B^T_{\mathscr{I}(f(u))})_{ij} = 1$, else $(B_{\mathscr{I}(f(u))}B^T_{\mathscr{I}(f(u))})_{ij} = 0$. For the sake of simplicity, let $S = I_n - B^T_{\mathscr{I}(f(u))}B_{\mathscr{I}(f(u))}$ and

$$M = \begin{bmatrix} B_{\mathscr{I}(f(u))}B^T_{\mathscr{I}(f(u))} & B_{\mathscr{I}(f(u))}a \\ a^T B^T_{\mathscr{I}(f(u))} & a^T a \end{bmatrix}.$$

We can calculate the determinant of M as $\det(M) = a^T Sa$. If $a^T Sa \neq 0$, then M is invertible and the Moore-Penrose pseudo-inverse of M is just the inverse of M. Hence we have

$$\begin{aligned} N &= I_n - \begin{bmatrix} B_{\mathscr{I}(f(u))} \\ a^T \end{bmatrix}^T M^{-1} \begin{bmatrix} B_{\mathscr{I}(f(u))} \\ a^T \end{bmatrix} \\ &= S(I_n - \frac{1}{a^T Sa} aa^T)S. \end{aligned}$$

Similarly, if $a^T Sa = 0$, we have $N = S$. In summary, we have

$$N = \begin{cases} S(I_n - \frac{1}{a^T Sa} aa^T)S, & \text{if } a^T Sa \neq 0, \\ S, & \text{otherwise,} \end{cases} \tag{3.6}$$

is an element of $J(f(u))$.

Now we summerize the SsN method as follows.

Algorithm 3 : The SsN method for the subproblem (3.4)

Give $\mu \in (0,\frac{1}{2})$, $\delta \in (0,1)$, $\tau \in (0,1]$, $\eta \in (0,1)$, and an initial point $u^0 \in \mathbb{R}^n$. For $i = 0,1,2,\cdots$, iterate the following steps:

Step 1. Let $P_i = Q + \sigma_k QN_iQ$, where N_i defined in (3.6) is an element of $J(f(u^i))$. Then we apply the conjugate gradient method to find an approximate solution to the following system:

$$P_id^i = -\nabla\varphi^k(u^i) \quad d^i \in \mathrm{Ran}(Q), \tag{3.7}$$

such that

$$\|P_id^i + \nabla\varphi^k(u^i)\| \le \min(\eta, \|\nabla\varphi^k(u^i)\|^{1+\tau}).$$

Step 2. (Line search) Set $\alpha_i = \delta^t$, where t is the first non-negative integer such that

$$\varphi^k(u^i + \delta^t d^i) \le \varphi^k(u^i) + \mu\delta^t\langle\nabla\varphi^k(u^i), d^i\rangle.$$

Step 3. Set $u^{i+1} = u^i + \alpha_i d^i$.

In fact, when the problem size is large, directly solving linear system (3.7) becomes costly. But we note that at each iteration of Algorithm 3 we only require updating Qd and dQd. Hence, instead of computing the solution d of (3.7), we solve a relatively smaller linear system to obtain Qd and d^TQd by making full use of the sparsity structure of the generalized Hessian. Let $\rho := \{1,\cdots,n\}\setminus\mathscr{I}(f(u))$ be an index set, and $t = |\rho|$ represent the number of the elements in the set ρ. Assume that $\hat{d}$ is the solution to linear system (3.7). According to the theory in [10], we have

(a) If $a^TSa \neq 0$,

$$Q\hat{d} = Q_\rho^T h_\rho(u) - \nabla\varphi^k(u) + \sigma_k \frac{a_\rho^T Q_{\rho\rho}h_\rho(u) - a_\rho^T\nabla\varphi_\rho^k(u)}{a_\rho^Ta_\rho - \sigma_k a_\rho^T Q_{\rho\rho}a_\rho} Q_\rho^T a_\rho,$$

where $Q_{\rho\rho}$ is a matrix consisting of the rows and columns of Q indexed by ρ, $\nabla\varphi_\rho^k(u) \in \mathbb{R}^t$, $a_\rho \in \mathbb{R}^t$, $Q_\rho \in \mathbb{R}^{t\times n}$ are matrices consisting of the rows of $\nabla\varphi^k(u)$, a, Q indexed by ρ. Letting $s(u) := \Pi_H(f(u)) - u$, we have

$$\begin{aligned} \hat{d}^TQ\hat{d} &= (h_\rho(u))^TQ_{\rho\rho}h_\rho(u) - 2(h_\rho(u))^T\nabla\varphi_\rho^k(u) + (s(u))^TQs(u) \\ &+ 2\sigma_k\frac{(a_\rho^TQ_{\rho\rho}h_\rho(u) - a_\rho^T\nabla\varphi_\rho^k(u))^2}{(a_\rho^Ta_\rho - \sigma_k a_\rho^TQ_{\rho\rho}a_\rho)^2}a_\rho^Ta_\rho \\ &- \sigma_k^2\frac{(a_\rho^TQ_{\rho\rho}h_\rho(u) - a_\rho^T\nabla\varphi_\rho^k(u))^2}{(a_\rho^Ta_\rho - \sigma_k a_\rho^TQ_{\rho\rho}a_\rho)^2}a_\rho^TQ_{\rho\rho}a_\rho, \end{aligned}$$

where $h_\rho(u) \in \mathbb{R}^t$ is the solution of the following linear system

$$\left(\frac{1}{\sigma_k}I_t + Q_{\rho\rho} + \frac{\sigma_k Q_{\rho\rho}a_\rho a_\rho^T Q_{\rho\rho}}{a_\rho^Ta_\rho - \sigma_k a_\rho^TQ_{\rho\rho}a_\rho}\right)h_\rho(u) = \left(I_t + \frac{\sigma_k Q_{\rho\rho}a_\rho a_\rho^T}{a_\rho^Ta_\rho - \sigma_k a_\rho^TQ_{\rho\rho}a_\rho}\right)\nabla\varphi_\rho^k(u) \tag{3.8}$$

(b) If $a^TSa = 0$, then $Q\hat{d} = Q_\rho^T h_\rho(u) - \nabla\varphi^k(u)$ and

$$\hat{d}^TQ\hat{d} = (h_\rho(u))^TQ_{\rho\rho}h_\rho(u) - 2(h_\rho(u))^T\nabla\varphi_\rho^k(u) + (s(u))^TQs(u),$$

where $h_\rho(u) \in \mathbb{R}^t$ is the solution of the following linear system

$$(\frac{1}{\sigma_k}I_t + Q_{\rho\rho})h_\rho(u) = \nabla\varphi_\rho^k(u). \tag{3.9}$$

For case (a), compared to linear system (3.7), the coefficient matrix in (3.8) is only a $t \times t$ matrix. Due to the sparsity structure of the generalized Hessian, t is typically much smaller than n, which means that we only need to solve a t-dimensional linear system. This greatly reduces both the computational and storage costs. For case (b), we solve linear system (3.9) instead of (3.7) in a similar approach.

4. Convergence of the Algorithms

In this section, we present the convergence analysis of the algorithms.

For Algorithm 1, the APG method is a traditional algorithm with a well-developed theoretical foundation. Let $g := \frac{1}{2}\langle x, Qx\rangle + \langle b, x\rangle$, and ∇g be the gradient of g. Especially, ∇g is Lipschitz continuous with constant L. Now we directly present the theory below. As for the details, one may refer to [11].

Theorem 4.1. *Let $\{x^k\}$ be a sequence generated by Algorithm 1 and x^* be the optimal solution of the problem (P). Then we have*

$$g(x^k) - g(x^*) \leq \frac{L\|x^* - x^0\|^2}{k^2}, k \geq 1.$$

For Algorithm 2, we first analyze the stopping criteria for subproblem (3.4). Based on [12, 13], we can adopt the following stopping criteria

$$\Phi^k(u^{k+1}, v^{k+1}) - \inf_{u,v\in\mathbb{R}^n} \Phi(u,v) \leq \varepsilon_k^2/(2\sigma_k), \Sigma_{k=1}^{+\infty}\sigma_k < +\infty, \tag{4.1}$$

$$\Phi^k(u^{k+1}, v^{k+1}) - \inf_{u,v\in\mathbb{R}^n} \Phi(u,v) \leq \delta_k^2/(2\sigma_k)\|x^{k+1} - x^k\|, \Sigma_{k=1}^{+\infty}\delta_k < +\infty, \tag{4.2}$$

where $\{\varepsilon_k\}$ and $\{\delta_k\}$ are two non-negative sequences. However, since we cannot directly obtain the value of $\inf_{u,v\in\mathbb{R}^n} \Phi(u,v)$, we cannot use such a stopping criteria directly. Thus we need to estimate an upper bound of $\Phi^k(u^{k+1}, v^{k+1}) - \inf\limits_{u,v\in\mathbb{R}^n} \Phi(u,v)$. By [14, Theorem 2.1.10], we have

$$\begin{aligned}&\Phi^k(u^{k+1}, v^{k+1}) - \inf_{u\in\text{Ran}(Q), v\in\mathbb{R}^n} \Phi(u,v)\\ &= \varphi^k(u^{k+1}) - \inf_{u\in\text{Ran}(Q)} \varphi^k(u) \leq \frac{\|\nabla\varphi^k(u^{k+1})\|^2}{2\lambda_{\min}(Q)},\end{aligned}$$

where $\lambda_{\min}(Q)$ is the minimal eigenvalue of Q. When $u^* = \underset{u\in\text{Ran}(Q)}{\text{argmin}}\ \varphi^k(u)$, we have $\nabla\varphi^k(u^*) = 0$. Hence, we can replace (4.1) and (4.2) with the following easy-to-check stopping criteria

$$\|\nabla\varphi^k(u^{k+1})\| \leq \frac{\hat{\varepsilon}_k}{\sqrt{\sigma_k}}, \tag{4.3}$$

$$\|\nabla\varphi^k(u^{k+1})\| \leq \frac{\hat{\delta}_k}{\sqrt{\sigma_k}}\|x^{k+1} - x^k\|, \tag{4.4}$$

where $\{\hat{\varepsilon}_k\}$ and $\{\hat{\delta}_k\}$ are two non-negative summable sequences.

Now we present the following convergence theorem.

Theorem 4.2. *Assume that the solution set of KKT system (3.2) is not empty, and $\{x^k\}$ and $\{(u^k,v^k)\}$ generated by Algorithm 2 under the stopping criteria defined in (4.3) are bounded. Let K be the solution set of (P), and (u^*,v^*) be the unique solution to dual problem (D). Then $\{x^k\}$ converges to $x^* \in K$, and $\{(u^k,v^k)\}$ converges to the unique solution (u^*,v^*).*

Proof. Since the solution set of the KKT system (3.2) is nonempty, then dual problem (D) has a solution, and the supremum of its dual problem is not negative infinity. According to [12, Theorem 1], the sequence $\{(u^k,v^k)\}$ and $\{x^k\}$ generated by the ALM is bounded. Based on [13, Theorem 4], the above conclusion holds immediately. □

For the convenience of the discussion, we define the maximal monotone operator T_l as follows

$$T_l(u,v,x) := \{(u',v',x') | (u',v',-x') \in \partial l(u,v,x)\},$$

where $\partial l(u,v,x)$ is subdifferential of Lagrangian function (3.1).

Since what we solve is a quadratic programming problem, by [15], $T_l(u,v,x)$ is a polyhedral multifunction. Let c be a number satisfying $c > \Sigma_{k=1}^{+\infty}\varepsilon_k$. According to [16], one sees that there exists $\kappa > 0$ associated with c such that, for any $(u,v;x)$ satisfying $\text{dist}((u,v;x),T_l^{-1}(0)) < c$,

$$\text{dist}((u,v;x),T_l^{-1}(0)) \leq \kappa \text{dist}(0,T_l(u,v;x)).$$

Now we are in the position to present the theorem on the convergence rate of the algorithm without the proof since we can follow [9, Theorem 2.5] to obtain the proof.

Theorem 4.3. *Assume that the solution set of problems (P) and (D) is nonempty, and let $(u^0,v^0;x^0)$ be an initial point satisfying $\text{dist}((u^0,v^0;x^0),T_l^{-1}(0)) < c - \Sigma_{k=1}^{+\infty}\varepsilon_k$. Then, for any $k > 0$, the sequence $\{(u^k,v^k;x^k)\}$ generated by Algorithm 2 under the stopping criteria (4.3) and (4.4) satisfy*

$$\text{dist}((u^{k+1},v^{k+1};x^{k+1}),T_l^{-1}(0)) \leq \mu_k \text{dist}((u^k,v^k;x^k),T_l^{-1}(0)),$$

where $\mu_k = (1-\delta_k)^{-1}(\delta_k + (1+\delta_k)\kappa(\sigma_k^2+\kappa^2)^{-\frac{1}{2}})$, and $\mu_\infty = \kappa(\kappa^2+\sigma_\infty^2)^{-\frac{1}{2}} < 1$.

Based on the theory in [9], we directly present the conclusion of the convergence of the subproblem.

Theorem 4.4. *Let $\{u^i\}$ be a sequence generated by Algorithm 3. Then $\{u^i\}$ converges to the unique u^* which satisfies $\nabla\psi^k(u^*) = 0$ and $\|u^{i+1}-u^*\| = O(\|u^i-u^*\|^{1+\tau})$.*

5. Numerical Experiments

In this section, we compare the performances of the ALM and APG-ALM in solving problem (P). All the numerical experiments in this paper are implemented in MATLAB R2022a on a laptop with AMD R7-7735H CPU 3.2GHz, 8 GB RAM.

We use the primal infeasibility R_P and the dual infeasibility R_D to measure the accuracy of the solution, where R_P, R_D are defined as below

$$\begin{aligned}\text{R}_\text{P} &:= \frac{\|Qu+v+b\|}{1+\|Qu+v+b\|},\\ \text{R}_\text{D} &:= \frac{\|Qu-Qx\|+\|\Pi_H(v+x)-x\|}{1+\|Qu-Qx\|+\|\Pi_H(v+x)-x\|}.\end{aligned}$$

If

$$\mathrm{R_{KKT}} := \max\{\mathrm{R_P}, \mathrm{R_D}\} \le 10^{-1},$$

then we transfer from the first stage of the algorithm to the second stage. If $\mathrm{R_{KKT}} < 10^{-5}$, then we stop the iteration.

We use pobj, dobj, and $\mathrm{R_G}$ to represent the primal objective value, the dual problem objective value, and the relative gap between them, where

$$\mathrm{R_G} := \frac{|\mathrm{pobj} - \mathrm{dobj}|}{1 + |\mathrm{pobj}| + |\mathrm{dobj}|}.$$

We focus on problem (P) and compare the numerical performances of the APG-ALM and ALM on the different datasets. In our comparison, we report the date name, the number of samples (m), features (n), the iteration number (iter) (especially, since the iteration number of the APG method is usually one, at most two, we omit it in the tables), the relative KKT residual ($\mathrm{R_{KKT}}$), the relative gap ($\mathrm{R_G}$), the primal objective value (pobj), the dual objective value (dobj) and the runing time (time). We list the numerical results with the random data in Tables 1 and 2, and the numerical results with the real data in Table 3. From the results, we can observe that using the APG-ALM is more efficient than directly using the ALM. The reason for this phenomenon is that the APG method in the first stage can quickly generate a rough solution, and in this process the algorithm in the second stage is less efficient.

TABLE 1. Comparisons of the ALM and APG-ALM for the problem (P) with the random data ($m > n$)

data name (m,n)	Algorithm	iter	$\mathrm{R_{KKT}}$	$\mathrm{R_G}$	pobj	dobj	time(s)
rand1	ALM	26	9.12e-6	1.25e-9	-8.93152e+3	-8.93152e+3	213.88
(71254,8135)	APG-ALM	5	8.70e-6	3.74e-11	-8.93152e+3	-8.93152e+3	81.08
rand2	ALM	61	2.29e-6	4.12e-11	-3.37249e+3	-3.37249e+3	254.5
(26521,10051)	APG-ALM	7	8.77e-7	9.97e-12	-3.37382e+3	-3.37382e+3	38.66
rand3	ALM	30	9.06e-6	1.01e-6	-6.83225e+3	-6.83224e+3	105.80
(54725,5675)	APG-ALM	12	8.67e-6	6.23e-7	-6.83224e+3	-6.83223e+3	47.47
rand4	ALM	9	9.06e-6	6.58e-9	-2.17901e+3	-2.17901e+3	23.68
(17452,1024)	APG-ALM	3	8.67e-6	4.03e-9	-2.18002e+3	-2.18002e+3	6.74
rand5	ALM	41	9.25e-6	1.57e-8	-1.38204e+4	-1.38204e+4	188.03
(110053,2390)	APG-ALM	11	8.47e-6	6.51e-10	-1.38216e+4	-1.38216e+4	40.05
rand6	ALM	30	8.60e-6	3.97e-11	-1.04426e+4	-1.04426e+4	69.24
(82564,5975)	APG-ALM	7	8.51e-6	3.56e-11	-1.04430e+4	-1.04430e+4	17.24
rand7	ALM	18	9.35e-6	5.43e-8	-7.81152e+3	-7.81152e+3	339.48
(62500,7849)	APG-ALM	13	8.74e-6	9.64e-9	-7.81160e+3	-7.81160e+3	69.11
rand8	ALM	10	8.88e-6	6.80e-7	-3.87032e+3	-3.87031e+3	110.52
(30396,14749)	APG-ALM	4	9.05e-7	3.19e-7	-3.87108e+3	-3.87108e+3	66.68

TABLE 2. Comparisons of the ALM and APG-ALM for the problem (P) with the random data ($m < n$)

data name (m,n)	Algorithm	iter	R_{KKT}	R_G	pobj	dobj	time(s)
rand1	ALM	26	1.96e-6	2.78e-11	-1.02115e+3	-1.02115e+3	235.81
(7924,72724)	APG-ALM	5	1.70e-7	2.38e-12	-1.02115e+3	-1.02115e+3	79.24
rand2	ALM	26	1.07e-6	3.15e-12	-1.27032e+3	-1.27032e+3	134.98
(9997,27824)	APG-ALM	7	9.18e-7	2.80e-12	-1.27032e+3	-1.27032e+3	64.04
rand3	ALM	30	5.42e-7	8.23e-9	-7.22162e+2	-7.22162e+2	117.56
(5678,57321)	APG-ALM	12	8.90e-8	9.45e-12	-7.21706e+2	-7.21706e+2	65.57
rand4	ALM	9	9.06e-6	4.29e-10	-1.73230e+2	-1.73230e+2	24.63
(1278,15732)	APG-ALM	3	4.12e-6	6.09e-11	-1.73142e+2	-1.73142e+2	13.90
rand5	ALM	71	9.14e-6	5.63e-7	-1.09041e+3	-1.09041e+3	310.14
(8499,62101)	APG-ALM	11	1.69e-3	1.51e-10	-1.09035e+3	-1.09035e+3	133.55
rand6	ALM	30	9.11e-6	7.03e-9	-1.89174e+3	-1.89174e+3	239.49
(15039,29467)	APG-ALM	7	6.66e-6	4.92e-12	-1.89092e+3	-1.89092e+3	108.41
rand7	ALM	18	6.42e-6	1.43e-10	-1.66305e+3	-1.66305e+3	228.64
(13094,27764)	APG-ALM	13	1.69e-7	6.29e-11	-1.66304e+3	-1.66304e+3	182.07
rand8	ALM	7	9.08e-6	1.54e-5	-1.88122e+3	-1.88116e+3	131.55
(14874,35031)	APG-ALM	5	8.91e-6	7.51e-6	-1.88306e+3	-1.88303e+3	95.21

TABLE 3. Comparisons of the ALM and APG-ALM for the problem (P) with the real data

date name (m,n)	Algorithm	iter	R_{KKT}	R_G	pobj	dobj	time(s)
pyrim.expanded	ALM	30	8.06e-6	2.12e-8	-1.66307e+1	-1.66307e+1	18.99
(74,201376)	APG-ALM	13	7.02e-6	1.71e-8	-1.66307e+1	-1.66307e+1	12.10
housing.expanded	ALM	7	9.57e-6	1.12e-5	-1.11262e+4	-1.11258e+4	19.10
(506,77520)	APG-ALM	5	9.13e-6	1.04e-5	-1.11257e+4	-1.11254e+4	15.31
mpg.expanded	ALM	4	8.51e-6	5.56e-5	-8.99219e+3	-8.99119e+3	2.78
(392,3432)	APG-ALM	3	8.04e-6	3.56e-5	-8.99162e+3	-8.99098e+3	1.93
bodyfat.expanded	ALM	9	9.89e-6	3.57e-6	-1.38962e+2	-1.38961e+2	13.47
(252,116280)	APG-ALM	6	8.74e-6	3.55e-6	-1.39402e+2	-1.39401e+2	9.18
E2006	ALM	13	8.67e-6	3.12e-5	-1.01467e+5	-1.01461e+5	17.50
(16087,150360)	APG-ALM	9	8.34e-6	2.94e-5	-1.01465e+5	-1.01459e+5	16.19

6. CONCLUSION

In this paper, we investigated a two-stage algorithm to solve a large-scale least squares problem with a simplex constraint. Numerical experiments demonstrate that the APG-ALM is a more efficient algorithm than the ALM.

Acknowledgements

The authors are grateful to the reviewers for useful suggestions which improved the contents of this paper. Chengjing Wang's work was supported by the Zhejiang Provincial Natural Science Foundation of China (Grant No. LTGY23H240002) and the National Natural Science Foundation of China (Grant No. U21A20169). Aimin Xu's work was supported by the Zhejiang Provincial Natural Science Foundation of China (Grant No. LTGY23H240002).

References

[1] J. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, J. Chanussot, Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches, IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens. 5(2012), 354-379.

[2] R. Jagannathan, T.S. Ma, Risk reduction in large portfolios: why imposing the wrong constraints helps, J. Finance. 58 (2003), 1651-1683.

[3] A. Cichocki, S. Amari, Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications, John Wiley & Sons, Hoboken, 2002.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers, Found. Trends Mach. Learn. 3 (2011), 1-122.

[5] L. Condat, Least-squares on the simplex for multispectral unmixing, Research Report, GIPSA-Lab, Univ. Grenoble Alpes, Grenoble, France. 2017.

[6] R.T. Rockafellar, Convex Analysis, Princeton University Press, Princeton, 1970.

[7] L. Condat, Fast projection onto the simplex and the L-1 ball, Math. Program. 158 (2016), 575-585.

[8] J. Han, D.F. Sun, Newton and quasi-Newton methods for normal maps with polyhedral sets, J. Optim. Theory Appl. 94 (1997), 659-676.

[9] X.D. Li, D.F. Sun, K.-C. Toh, On the efficient computation of a generalized Jacobian of the projector over the Birkhoff polytope, Math. Program. 179 (2020), 419-446.

[10] D.B. Niu, C.J. Wang, P.P. Tang, Q.S. Wang, E.B. Song, An efficient algorithm for a class of large-scale support vector machines exploiting hidden sparsity, IEEE Trans. Signal Process. 70(2022), 5608-5623.

[11] Yu. Nesterov, Gradient methods for minimizing composite functions, Math. Program. 140 (2013), 125-161.

[12] R.T. Rockafellar, Monotone operators and the proximal point algorithm, SIAM J. Control Optim. 14 (1976), 877-898.

[13] R.T. Rockafellar, Augmented Lagrangians and applications of the proximal point algorithm in convex programming, Math. Oper. Res. 1 (1976), 97-116.

[14] Yu. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, Springer, Berlin, 2013.

[15] J. Sun, On monotropic piecewise quadratic programming, Ph.D. thesis, Dept. of Math. Univ. of Washington, Washington, 1986.

[16] S.M. Robinson, Some continuity properties of polyhedral multifunctions, Math. Program. Stud. 14 (1981), 206-214.