Filomat 39:12 (2025), 4161–4172
https://doi.org/10.2298/FIL2512161W

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

Faster accelerated alternating direction method of multipliers

Shi-Liang Wu[a,b,*], Sha-Sha Fan[a], Cui-Xia Li[a]

[a] *School of Mathematics, Yunnan Normal University, Kunming, Yunnan, 650500, P.R. China*
[b] *Yunnan Key Laboratory of Modern Analytical Mathematics and Applications, Yunnan Normal University, Kunming, Yunnan, 650500, P.R. China*

Abstract. In this paper, by implanting the classical Gauss-Seidel method into the Fast ADMM method, we propose the faster accelerated alternating direction method of multipliers (F-AADMM) to solve the convex optimization problem, and discuss the convergence property and convergence rate of the F-AADMM method. Numerical experiments are provided to show the performance of the F-AADMM method.

1. Introduction

In this paper, we consider the following convex optimization problem with linear equality constraints:

$$\begin{aligned} \min \quad & f(x) + h(y) \\ \text{s.t.} \quad & x + A^{\mathrm{T}} y = b, \end{aligned} \tag{1}$$

where $f : R^n \to R \cup \{\infty\}$ and $h : R^l \to R \cup \{\infty\}$ are closed convex functions and differentiable with Lipschitz continuous gradient on their effective domains, $A \in R^{l \times n}$ is a given matrix, and $b \in R^n$ is a given vector.

For such a separable convex minimization model, ADMM in [8] turns out to be a benchmark solver, which has been widely used for many applications [3, 7]. The study of the ADMM algorithm and its variants has been well developed [1, 4, 10, 19], in particular, the convergence property and convergence rate of ADMM have been provided [6, 7, 11, 12, 15]. ADMM usually gets a low precision solution very quickly, but requires more iterations to converge to a high precision solution. In some engineering applications, the requirement for precision is very high, which prompts researchers to think about acceleration methods for ADMM. With the emergence of Nesterov's accelerated gradient descent [16], many scholars have successively improved the convergence speed of the ADMM method. For example, Zhang et al. [20] proposed its accelerated version on the base of the general minimum residual method (GMRES), Ouyang et al. [18] investigated an accelerated alternating direction method of multipliers (AADMM) for the acceleration of the linearized ADMM method, Goldstein et al. [9] studied the Fast ADMM method when the divisible

2020 *Mathematics Subject Classification*. Primary 90C26; Secondary 90C30, 65K05.
Keywords. Gauss-Seidel, Fast ADMM, F-AADMM.
Received: 31 January 2024; Revised: 18 August 2024; Accepted: 19 February 2025
Communicated by Marko Petković
Research supported by National Natural Science Foundation of China (No.11961082), Yunnan Key Laboratory of Modern Analytical Mathematics and Applications (No. 202302AN360007), Cross-integration Innovation Team of Modern Applied Mathematics and Life Sciences in Yunnan Province, China (202405AS350003), Basic Research Projects of Key Scientific Research Projects Plan in Henan Higher Education Institutions (25ZX013).
* Corresponding author: Shi-Liang Wu
Email addresses: slwuynnu@126.com (Shi-Liang Wu), fssynnu@126.com (Sha-Sha Fan), lixiatkynu@126.com (Cui-Xia Li)

objective function are strongly convex, and others. In addition, based on the block symmetric Gauss-Seidel (sGS) decomposition theorem, Li et al. [13] extended the classical block sGS method to solve the convex composite quadratic programming (CCQP).

In this paper, to develop the Fast ADMM method, by coupling the Fast ADMM method with the Gauss-Seidel method, we propose the F-AADMM method for solving the problem (1), and discuss the convergence property and convergence rate of the F-AADMM method. Numerical examples are provided to illustrate the performance of the F-AADMM method.

2. Preliminaries

In this section, we will recall some definitions, notations and lemmas. In the whole paper, $\|\cdot\|$ denotes the Euclidean norm.

Definition 2.1. *[17] (Conjugate Function) Given a function $f : C \subseteq R^n \to R$, its conjugate function is defined as*

$$f^*(y) = \sup_{x \in dom f} \{\langle x, y\rangle - f(x)\}.$$

In particular, $f^{**} = f$ when the function f is a closed convex function.

Lemma 2.2. *[17] (Fenchel-Young inequality) Let f^* be the conjugate function of the function f. Then*

$$\langle x, y\rangle \leq f(x) + f^*(y).$$

In particular, when the function f is a closed convex function, the following statements hold:
1) if f is differentiable, then

$$y = \nabla f(x) \Rightarrow x = \nabla f^*(y);$$

2) if f is not differentiable, then

$$y = \partial f(x) \Rightarrow x = \partial f^*(y).$$

Lemma 2.3. *[17] Let $f : C \subseteq R^n \to R$ be a continuous differentiable function. Then*

$$(\left\|\nabla f(x) - \nabla f(y)\right\| \leq L_f \left\|y - x\right\|)$$

and

$$f(y) \leq f(x) + \langle \nabla f(x), y - x\rangle + \frac{L_f}{2}\|y - x\|^2, \quad \forall x, y \in C.$$

Definition 2.4. *[17] (Proximity operator) Let the function $f : C \subseteq R^n \to R$ be a closed normal convex function. Then the proximity operator of the function f is defined as*

$$prox_{\mu f}(x) = \underset{y}{argmin}\left\{f(y) + \frac{1}{2\mu}\|y - x\|^2\right\}, \quad \forall x \in C.$$

To design the F-AADMM method, we briefly review the Gauss-Seidel method [5, 14].

Let $Ax = b$, where $A \in R^{n\times n}$ is non-singular with its diagonal elements $a_{ii} \neq 0$ $(i = 1, 2, \ldots, n)$, $b \in R^n$, x is unknown. Dividing A into the sum of three matrices $A = L + D + U$, where D is the diagonal matrix, L and U in order are the lower and upper triangular matrices of A, the Gauss-Seidel method can be designed as

$$x_{k+1} = -(D + L)^{-1}Ux_k + (D + L)^{-1}b.$$

3. The F-AADMM method

In this section, firstly, we propose the F-AADMM method by combing the Gauss-Seidel method and the Fast ADMM method. Secondly, we discuss the convergence property and convergence rate of the F-AADMM method.

3.1. Algorithm description

For the problem (1), introducing the lagrange multiplier λ for the constraint $x + A^{\mathrm{T}}y = b$ yields the Lagrangian function

$$\begin{aligned}\mathcal{L}(x, y, \lambda) &= f(x) + h(y) - \lambda^{\mathrm{T}}\left(x + A^{\mathrm{T}}y - b\right)\\ &= \left(f(x) - \lambda^{\mathrm{T}}x\right) + \left(h(y) - \lambda^{\mathrm{T}}A^{\mathrm{T}}y\right) + \lambda^{\mathrm{T}}b,\end{aligned}$$

and the augmented Lagrangian function (ALF) of (1) is given by

$$\hat{\mathcal{L}}(x, y, \lambda) = f(x) + h(y) - \lambda^{\mathrm{T}}\left(x + A^{\mathrm{T}}y - b\right) + \frac{\beta}{2}\left\|x + A^{\mathrm{T}}y - b\right\|^2,$$

where $\beta > 0$ is a penalty parameter. Using the definition of conjugate function, along with the duality theory, it is easy to know that the dual problem of (1) is

$$\max_{\lambda} \quad \phi(\lambda) = -f^*(\lambda) - h^*(A\lambda) + \lambda^{\mathrm{T}}b. \tag{2}$$

Here, we recall the Fast ADMM for problem (1), see Algorithm 3.1.

Algorithm 3.1. *Fast ADMM for solving the problem (1)*

Input: $y_{-1} = \hat{y}_0 \in R^l, \lambda_{-1} = \hat{\lambda}_0 \in R^n, \beta > 0, \alpha_1 = 1$

1. ***for*** $k = 0, 1, 2, 3, \cdots$ ***do***
2. $x_k = \underset{x\in R^n}{argmin}\left\{f(x) - \hat{\lambda}_k^T x + \frac{\beta}{2}\|x + A^T\hat{y}_k - b\|^2\right\};$
3. $y_k = \underset{y\in R^l}{argmin}\left\{h(y) - \hat{\lambda}_k^T(A^T y) + \frac{\beta}{2}\|x_k + A^T y - b\|^2\right\};$
4. $\lambda_k = \hat{\lambda}_k - \beta(x_k + A^T y_k - b);$
5. $\alpha_{k+1} = \frac{1+\sqrt{4\alpha_k^2+1}}{2};$
6. $\hat{y}_{k+1} = y_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(y_k - y_{k-1});$
7. $\hat{\lambda}_{k+1} = \lambda_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(\lambda_k - \lambda_{k-1});$
8. ***end***

To improve the Fast ADMM for solving the problem (1), we consider the following two aspects:
(i) the objective functions are not always strongly convex functions;
(ii) use the Gauss-Seidel method to further accelerate the existing Fast ADMM algorithm.
Based on this, the F-AADMM method can be designed, see Algorithm 3.2.

Algorithm 3.2. *F-AADMM for solving the problem (1)*

Input: $y_{-1} = \hat{y}_0 \in R^l, \lambda_{-1} = \hat{\lambda}_0 \in R^n, \beta > 0, \alpha_1 = 1$

1. ***for*** $k = 0, 1, 2, 3, \cdots$ ***do***
2. $x_k = \underset{x \in R^n}{argmin} \left\{ f(x) - \hat{\lambda}_k^T x + \frac{\beta}{2} \|x + A^T \hat{y}_k - b\|^2 \right\}$;
3. $y_k = \underset{y \in R^l}{argmin} \left\{ h(y) - \hat{\lambda}_k^T (A^T y) + \frac{\beta}{2} \|x_k + A^T y - b\|^2 \right\}$;
4. $\lambda_k = \hat{\lambda}_k - \beta(x_k + A^T y_k - b)$;
5. $\alpha_{k+1} = \frac{1 + \sqrt{4\alpha_k^2 + 1}}{2}$;
6. $y_k^+ = -(D + L)^{-1} U y_k + (D + L)^{-1} x_k$;
7. $\hat{y}_{k+1} = y_k^+ + \frac{\alpha_k - 1}{\alpha_{k+1}} (y_k^+ - y_{k-1})$;
8. $\hat{\lambda}_{k+1} = \lambda_k + \frac{\alpha_k - 1}{\alpha_{k+1}} (\lambda_k - \lambda_{k-1})$;
9. ***end***

Comparing the Fast ADMM method for solving the problem (1), the step 6 of F-AADMM algorithm (i.e., Algorithm 3.2) is critical, which uses the Gauss-Seidel method to further accelerate the above Fast ADMM algorithm. Unlike [13], we incorporate the Gauss-Seidel iteration into the Fast ADMM algorithm, and achieve a faster convergence speed, compared with the Fast ADMM method.

For the problem (1), its optimal solutions [2] can be described below

$$x^* + A^{\mathrm{T}} y^* - b = 0, \tag{3}$$
$$\nabla f(x^*) - \lambda^* = 0, \tag{4}$$
$$\nabla h(y^*) - A\lambda^* = 0, \tag{5}$$

where (3) represents the feasibility of the original variable (x^*, y^*), the dual variable λ^* satisfies the dual feasibility condition (4) and (5).

For solving the sub-problem about x in the F-AADMM algorithm (i.e., step 2 of Algorithm 3.2) and the subproblem of y (i.e., step 3 of Algorithm 3.2), together with the optimality conditions, we have

$$\nabla f(x_k) - \hat{\lambda}_k + \beta(x_k + A^{\mathrm{T}} \hat{y}_k - b) = 0, \tag{6}$$
$$\nabla h(y_k) - A\left[\hat{\lambda}_k - \beta(x_k + A^{\mathrm{T}} y_k - b)\right] = 0. \tag{7}$$

Substituting the step 4 of Algorithm 3.2 into Eqs. (6) and (7), we can get

$$\nabla f(x_k) - \lambda_k + \beta A^{\mathrm{T}}(\hat{y}_k - y_k) = 0, \tag{8}$$
$$\nabla h(y_k) - A\lambda_k = 0. \tag{9}$$

One common way to measure how well the iterates of Algorithm 3.2 satisfy the optimality conditions are to define the primal and dual residuals:

$$r_k = x_k + A^{\mathrm{T}} y_k - b,$$
$$d_k = \beta A^{\mathrm{T}}(\hat{y}_k - y_k),$$

where the primal residual r_k and dual residual d_k indicate how far the iterates are from a solution.

Looking carefully at Eqs. (8)-(9) and the optimality conditions (3)-(5) for the problem (1), it is not difficult to find that there is (x_k, y_k, λ_k) satisfying the optimality conditions, when $r_k = 0$ and $d_k = 0$.

In addition, for the classical ADMM, if (x, y, λ) is given in the domain of definition, then the corresponding new iteration point (x', y', λ') can be obtained, i.e.,

$$x' = \underset{x\in R^n}{\operatorname{argmin}} \left\{ f(x) - \lambda^{\mathrm{T}} x + \frac{\beta}{2} \left\| x + A^{\mathrm{T}} y - b \right\|^2 \right\}, \tag{10}$$

$$y' = \underset{y\in R^l}{\operatorname{argmin}} \left\{ h(y) - \lambda^{\mathrm{T}} (A^{\mathrm{T}} y) + \frac{\beta}{2} \left\| x' + A^{\mathrm{T}} y - b \right\|^2 \right\}, \tag{11}$$

$$\lambda' = \lambda - \beta(x' + A^{\mathrm{T}} y' - b). \tag{12}$$

3.2. Convergence analysis

In this subsection, we will discuss the convergence property of the F-AADMM method. In the convergence analysis, we always assume that the conjugate functions f and h are differentiable.

To obtain the convergence property of the F-AADMM method, we require some lemmas.

Lemma 3.3. *For any $y \in R^l, \lambda \in R^n$, we have*

$$x' = \nabla f^*(\bar{\lambda}),$$
$$A^T y' = A^T \nabla h^*(A\lambda'),$$

where $\bar{\lambda} = \lambda - \beta(x' + A^{\mathrm{T}} y - b)$.

Proof. According to the optimality condition of x' in (10), we have

$$\nabla f(x') - \lambda + \beta(x' + A^{\mathrm{T}} y - b) = \nabla f(x') - \bar{\lambda} = 0,$$

from which we can get

$$\bar{\lambda} = \nabla f(x'). \tag{13}$$

By using the differentiability of conjugate functions, Eq. (13) is equivalent to $x' = \nabla f^*(\bar{\lambda})$.

Similarly, according to the optimality condition of y' in (11), we can get

$$\nabla h(y') - A\lambda + \beta A(x' + A^{\mathrm{T}} y' - b) = \nabla h(y') - A\lambda' = 0. \tag{14}$$

By using the differentiability of conjugate functions again, Eq. (14) is equivalent to $A^{\mathrm{T}} y' = A^{\mathrm{T}} \nabla h^*(A\lambda')$. ■

Lemma 3.4. *Assume that $A^T y = \Phi(\lambda)$, $L(\Phi)$ is Lipschitz constant. The iterative sequences $\{\lambda_k\}$ and $\{\hat{\lambda}_k\}$ produced of Algorithm 3.2 satisfy*

$$A^T y_k = \Phi(\lambda_k);\ A^T \hat{y}_k = \Phi(\hat{\lambda}_k),$$

and

$$\phi(\lambda') - \phi(\kappa) \geq \frac{1}{\beta} \langle \kappa - \lambda, \lambda - \lambda' \rangle + \frac{1}{2\beta} \|\lambda - \lambda'\|^2, \quad \forall \kappa \in R^n. \tag{15}$$

Proof. From Lemma 3.3, we have $A^{\mathrm{T}} y_k = \Phi(\lambda_k)$ and $A^{\mathrm{T}} \hat{y}_k = \Phi(\hat{\lambda}_k)$.

Combining the first-order properties of the convex function and Lemma 3.3 yields

$$\begin{aligned}
\phi(\lambda') - \phi(\kappa) &= f^*(\kappa) - f^*(\bar{\lambda}) + f^*(\bar{\lambda}) - f^*(\lambda') + h^*(A\kappa) - h^*(A\lambda') - \langle b, \kappa - \lambda' \rangle \\
&\geq \langle \nabla f^*(\bar{\lambda}), \kappa - \bar{\lambda} \rangle - \langle \nabla f^*(\bar{\lambda}), \lambda' - \bar{\lambda} \rangle + \left\langle A^{\mathrm{T}} \nabla h^*(A\lambda'), \kappa - \lambda' \right\rangle - \langle b, \kappa - \lambda' \rangle \\
&\geq \langle \nabla f^*(\bar{\lambda}), \kappa - \lambda' \rangle - \frac{1}{2\beta} \|\lambda - \lambda'\|^2 + \left\langle A^{\mathrm{T}} \nabla h^*(A\lambda'), \kappa - \lambda' \right\rangle - \langle b, \kappa - \lambda' \rangle \\
&= \left\langle \nabla f^*(\bar{\lambda}) + A^{\mathrm{T}} \nabla h^*(A\lambda'), \kappa - \lambda' \right\rangle - \frac{1}{2\beta} \|\lambda - \lambda'\|^2 - \langle b, \kappa - \lambda' \rangle \\
&= \left\langle x' + A^{\mathrm{T}} y' - b, \kappa - \lambda' \right\rangle - \frac{1}{2\beta} \|\lambda - \lambda'\|^2 \\
&= \frac{1}{\beta} \langle \lambda - \lambda', \kappa - \lambda' \rangle + \frac{1}{2\beta} \|\lambda - \lambda'\|^2 , \qquad (16)
\end{aligned}$$

which completes the proof. ■

For the convenience of the following discussion, let

$$m_k = \alpha_k \lambda_k - (\alpha_k - 1)\lambda_{k-1} - \lambda^*. \qquad (17)$$

Lemma 3.5. *Based on (17) and Algorithm 3.2, we have*

$$m_{k+1} = m_k + \alpha_{k+1}(\lambda_{k+1} - \hat{\lambda}_{k+1}). \qquad (18)$$

Proof. Combining the definition of m_k and the step 8 of Algorithm 3.2, we have that

$$\begin{aligned}
m_{k+1} &= \alpha_{k+1}\lambda_{k+1} - (\alpha_{k+1} - 1)\lambda_k - \lambda^* \\
&= \lambda_k - \lambda^* + \alpha_{k+1}(\lambda_{k+1} - \lambda_k) \\
&= \lambda_k - (\alpha_k - 1)\lambda_{k-1} - \lambda^* + \alpha_{k+1}(\lambda_{k+1} - \lambda_k) + (\alpha_k - 1)\lambda_{k-1} \\
&= \alpha_k\lambda_k - (\alpha_k - 1)\lambda_{k-1} - \lambda^* + \alpha_{k+1}(\lambda_{k+1} - \lambda_k) - (\alpha_k - 1)(\lambda_k - \lambda_{k-1}) \\
&= m_k + \alpha_{k+1}(\lambda_{k+1} - \lambda_k) - (\alpha_k - 1)(\lambda_k - \lambda_{k-1}) \\
&= m_k + \alpha_{k+1}(\lambda_{k+1} - \lambda_k) - \alpha_{k+1}(\hat{\lambda}_{k+1} - \lambda_k) \\
&= m_k + \alpha_{k+1}(\lambda_{k+1} - \hat{\lambda}_{k+1}),
\end{aligned}$$

which completes the proof. ■

Lemma 3.6. *The iterative sequence $\{(\alpha_k, \lambda_k, \hat{\lambda}_k)\}$ generated by Algorithm 3.2 satisfies*

$$\|m_{k+1}\|^2 - \|m_k\|^2 \leq 2\beta\alpha_k^2 \left[\phi(\lambda^*) - \phi(\lambda_k)\right] - 2\beta\alpha_{k+1}^2 \left[\phi(\lambda^*) - \phi(\lambda_{k+1})\right]. \qquad (19)$$

Proof. According to the definition of m_k and Lemma 3.5, we can get

$$\begin{aligned}
&\|m_{k+1}\|^2 - \|m_k\|^2 \\
&= 2\alpha_{k+1} \left\langle m_k, \lambda_{k+1} - \hat{\lambda}_{k+1} \right\rangle + \alpha_{k+1}^2 \|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\
&= 2\alpha_{k+1} \left\langle \alpha_k\lambda_k - (\alpha_k - 1)\lambda_{k-1} - \lambda^*, \lambda_{k+1} - \hat{\lambda}_{k+1} \right\rangle + \alpha_{k+1}^2 \|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\
&= 2\alpha_{k+1} \left\langle (\alpha_k - 1)(\lambda_k - \lambda_{k-1}) + \lambda_k - \lambda^*, \lambda_{k+1} - \hat{\lambda}_{k+1} \right\rangle + \alpha_{k+1}^2 \|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2
\end{aligned} \qquad (20)$$

$$
\begin{aligned}
&= 2\alpha_{k+1}\left\langle \alpha_{k+1}\hat{\lambda}_{k+1} + (1-\alpha_{k+1})\lambda_k - \lambda^*, \lambda_{k+1} - \hat{\lambda}_{k+1}\right\rangle + \alpha_{k+1}^2\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\
&= 2\alpha_{k+1}\left\langle (\alpha_{k+1}-1)(\hat{\lambda}_{k+1} - \lambda_k) + \hat{\lambda}_{k+1} - \lambda^*, \lambda_{k+1} - \hat{\lambda}_{k+1}\right\rangle + \alpha_{k+1}^2\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\
&= 2\alpha_{k+1}\left\langle (\alpha_{k+1}-1)(\hat{\lambda}_{k+1} - \lambda_k) + \hat{\lambda}_{k+1} - \lambda^*, \lambda_{k+1} - \hat{\lambda}_{k+1}\right\rangle \\
&\quad + [\alpha_{k+1}(\alpha_{k+1}-1) + \alpha_{k+1}]\,\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\
&= 2\alpha_{k+1}(\alpha_{k+1}-1)\left(\left\langle \hat{\lambda}_{k+1} - \lambda_k, \lambda_{k+1} - \hat{\lambda}_{k+1}\right\rangle + \frac{1}{2}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2\right) \\
&\quad + 2\alpha_{k+1}\left(\left\langle \hat{\lambda}_{k+1} - \lambda^*, \lambda_{k+1} - \hat{\lambda}_{k+1}\right\rangle + \frac{1}{2}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2\right). \qquad (21)
\end{aligned}
$$

Applying Lemma 3.4 with $\kappa = \lambda_k, \lambda = \hat{\lambda}_{k+1}, \lambda' = \lambda_{k+1}$, we have

$$
\phi(\lambda_{k+1}) - \phi(\lambda_k) \geq \frac{1}{2\beta}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 + \frac{1}{\beta}\left\langle \hat{\lambda}_{k+1} - \lambda_k, \lambda_{k+1} - \hat{\lambda}_{k+1}\right\rangle. \qquad (22)
$$

Applying Lemma 3.4 with $\kappa = \lambda^*, \lambda = \hat{\lambda}_{k+1}, \lambda' = \lambda_{k+1}$ again, we get

$$
\phi(\lambda_{k+1}) - \phi(\lambda^*) \geq \frac{1}{2\beta}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 + \frac{1}{\beta}\left\langle \hat{\lambda}_{k+1} - \lambda^*, \lambda_{k+1} - \hat{\lambda}_{k+1}\right\rangle. \qquad (23)
$$

Substituting (22) and (23) into (21) yields

$$
\begin{aligned}
&\|m_{k+1}\|^2 - \|m_k\|^2 \\
&\leq 2\alpha_{k+1}(\alpha_{k+1}-1)\beta\left[\phi(\lambda_{k+1}) - \phi(\lambda_k)\right] + 2\alpha_{k+1}\beta\left[\phi(\lambda_{k+1}) - \phi(\lambda^*)\right] \\
&= 2\beta\alpha_{k+1}^2\phi(\lambda_{k+1}) - 2\beta\alpha_{k+1}^2\phi(\lambda_k) + 2\beta\alpha_{k+1}\phi(\lambda_k) - 2\beta\alpha_{k+1}\phi(\lambda^*) \\
&= 2\beta\alpha_{k+1}^2\phi(\lambda_{k+1}) - 2\beta(\alpha_{k+1}^2 - \alpha_{k+1})\phi(\lambda_k) - 2\beta(\alpha_{k+1}^2 - \alpha_k^2)\phi(\lambda^*) \qquad (24) \\
&= 2\beta\alpha_{k+1}^2\phi(\lambda_{k+1}) - 2\alpha_k^2\beta\phi(\lambda_k) - 2(\alpha_{k+1}^2 - \alpha_k^2)\beta\phi(\lambda^*) \qquad (25) \\
&= 2\beta\alpha_k^2\left[\phi(\lambda^*) - \phi(\lambda_k)\right] + 2\beta\alpha_{k+1}^2\left[\phi(\lambda_{k+1}) - \phi(\lambda^*)\right],
\end{aligned}
$$

where Eqs. (24) and (25) are obtained by step 5 in Algorithm 3.2, that is, $\alpha_{k+1} = \frac{1+\sqrt{4\alpha_k^2+1}}{2}$ or $\alpha_k^2 = \alpha_{k+1}^2 - \alpha_{k+1}$.■

Based on Lemmas 3.3-3.6, the convergence of the F-AADMM method can be obtained. Firstly, we can get the following result.

Theorem 3.5 The iterative sequence $\{\lambda_k\}$ produced by Algorithm 3.2 satisfies

$$
\phi(\lambda^*) - \phi(\lambda_k) \leq \frac{2\|\hat{\lambda}_1 - \lambda^*\|^2}{\beta(k+1)^2}. \qquad (26)
$$

Proof. From Lemma 3.6 we have

$$
\|m_{k+1}\|^2 + 2\beta\alpha_{k+1}^2\left[\phi(\lambda^*) - \phi(\lambda_{k+1})\right] \leq \|m_k\|^2 + 2\beta\alpha_k^2\left[\phi(\lambda^*) - \phi(\lambda_k)\right], \qquad (27)
$$

which implies $\left\{\|m_k\|^2 + 2\beta\alpha_k^2\left[\phi(\lambda^*) - \phi(\lambda_k)\right]\right\}$ is decreasing. Further, we have

$$
\|m_k\|^2 + 2\beta\alpha_k^2\left[\phi(\lambda^*) - \phi(\lambda_k)\right] \leq \|m_1\|^2 + 2\beta\alpha_1^2\left[\phi(\lambda^*) - \phi(\lambda_1)\right].
$$

Making $k = 0$ in Eq. (23), we have

$$
\begin{aligned}
\phi(\lambda_1) - \phi(\lambda^*) &\geq \frac{1}{2\beta}\|\lambda_1 - \hat{\lambda}_1\|^2 + \frac{1}{\beta}\left\langle \hat{\lambda}_1 - \lambda^*, \lambda_1 - \hat{\lambda}_1\right\rangle \\
&= \frac{1}{2\beta}\left(\|\lambda_1 - \lambda^*\|^2 - \|\hat{\lambda}_1 - \lambda^*\|^2\right).
\end{aligned}
$$

Further, inequality (27) can be transformed into

$$\begin{aligned}
2\beta\alpha_{k+1}^2\left[\phi(\lambda^*)-\phi(\lambda_{k+1})\right] &\le \|m_k\|^2-\|m_{k+1}\|^2+2\beta\alpha_k^2\left[\phi(\lambda^*)-\phi(\lambda_k)\right]\\
&\le \|m_k\|^2+2\beta\alpha_k^2\left[\phi(\lambda^*)-\phi(\lambda_k)\right]\\
&\le \cdots\cdots\\
&\le \|m_1\|^2+2\beta\alpha_1^2\left[\phi(\lambda^*)-\phi(\lambda_1)\right] \quad (\text{where } \alpha_1=1)\\
&\le \|\lambda_1-\lambda^*\|^2+2\beta\left[\phi(\lambda^*)-\phi(\lambda_1)\right]\\
&\le \|\hat{\lambda}_1-\lambda^*\|^2.
\end{aligned}$$

Since $\alpha_k > \alpha_{k-1}+\frac{1}{2} > 1+\frac{k}{2}$, we can easily obtain that

$$\phi(\lambda^*)-\phi(\lambda_k) \le \frac{2\|\hat{\lambda}_1-\lambda^*\|^2}{\beta(k+1)^2},$$

which completes the proof. ■

Theorem 3.6 Algorithm 3.2 produces an iterative sequence, whose primal-dual difference satisfies

$$\|r_k\|^2 \le O(1/k^2), \|d_k\|^2 \le O(1/k^2).$$

Proof. On the one hand, we apply Lemma 3.4 with $\lambda' = \lambda^*, \kappa = \lambda = \lambda_k$, and get

$$\phi(\lambda^*)-\phi(\lambda_k) \ge \frac{1}{2\beta}\|\lambda_k-\lambda^*\|^2.$$

According to Theorem 3.5, we get

$$\|\lambda_k-\lambda^*\|^2 \le 2\beta\left[\phi(\lambda^*)-\phi(\lambda_k)\right] \le \frac{4\|\hat{\lambda}_1-\lambda^*\|^2}{(k+1)^2},$$

this shows

$$\|\lambda_k-\lambda^*\|^2 \le O(1/k^2).$$

Further,

$$\|\lambda_{k+1}-\lambda_k\|^2 \le \|\lambda_{k+1}-\lambda^*\|^2+\|\lambda^*-\lambda_k\|^2 \le O(1/k^2).$$

On the other hand, by the definition of $\phi(\lambda) = -f^*(\lambda) - h^*(A\lambda) - \lambda^{\mathrm{T}}b$, whose gradient has the Lipschitz continuity property, we have

$$\begin{aligned}
\phi(\hat{\lambda}_k) &\ge \phi(\lambda^*)+\left\langle\nabla\phi(\lambda^*),\hat{\lambda}_k-\lambda^*\right\rangle-\frac{L}{2}\|\hat{\lambda}_k-\lambda^*\|^2\\
&= \phi(\lambda^*)-\frac{L}{2}\|\hat{\lambda}_k-\lambda^*\|^2,
\end{aligned} \tag{28}$$

where L is the Lipschitz constant of $\nabla\phi$.

From Eq. (28), we have

$$\begin{aligned}\phi(\lambda_{k+1}) - \phi(\hat{\lambda}_{k+1}) &\leq \phi(\lambda_{k+1}) - \phi(\lambda^*) + \frac{L}{2}\|\hat{\lambda}_{k+1} - \lambda^*\|^2 \\ &\leq \frac{L}{2}\|\hat{\lambda}_{k+1} - \lambda^*\|^2 \\ &= \frac{L}{2}\|\lambda_k - \lambda^* + \frac{\alpha_k - 1}{\alpha_{k+1}}(\lambda_k - \lambda_{k-1})\|^2 \\ &\leq \frac{L}{2}\left(\|\lambda_k - \lambda^*\|^2 + \frac{(\alpha_k - 1)^2}{\alpha_{k+1}^2}\|(\lambda_k - \lambda_{k-1})\|^2\right) \\ &\leq O(1/k^2).\end{aligned}$$

Applying Lemma 3.4 with $\kappa = \lambda = \hat{\lambda}_{k+1},\ \lambda' = \lambda_{k+1}$ again, we get

$$\begin{aligned}\phi(\lambda_{k+1}) - \phi(\hat{\lambda}_{k+1}) &\geq \frac{1}{2\beta}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\ &= \frac{1}{2\beta}\|\beta(x_{k+1} + A^{\mathrm{T}}y_{k+1} - b)\|^2 \\ &= \frac{\beta}{2}\|r_{k+1}\|^2,\end{aligned}$$

and

$$\|r_{k+1}\|^2 \leq O(1/k^2).$$

For the residual d_k, using Lemma 3.4, one obtains

$$\begin{aligned}\|d_k\|^2 &= \|\beta A^{\mathrm{T}}(y_k - \hat{y}_k)\|^2 \\ &= \|\beta A^{\mathrm{T}}\left[\nabla h^*(A\lambda_k) - \nabla h^*(A\hat{\lambda}_k)\right]\|^2 \\ &\leq \beta^2\rho(A^TA)L(\Phi)\|\lambda_k - \hat{\lambda}_k\|^2 \\ &\leq O(1/k^2),\end{aligned}$$

which completes the proof. ■

4. Numerical results

In this section, we aim at showing the linear convergence rate of our proposed F-AADMM algorithm for solving the quadratic program below

$$\begin{aligned}\min \quad & \frac{1}{2}x^{\mathrm{T}}Qx + q^{\mathrm{T}}x + \frac{1}{2}y^{\mathrm{T}}Ry + r^{\mathrm{T}}y \\ \text{s.t.} \quad & Ax + By = b,\end{aligned} \tag{29}$$

where Q and R are symmetric positive semidefinite matrices in $R^{n\times n}$ and $R^{m\times m}$, respectively, $A \in R^{l\times n}$ and $B \in R^{l\times m}$ are two given matrices, $q \in R^n$, $r \in R^m$, and $b \in R^l$ are given vectors, $x \in R^n$ and $y \in R^m$. All codes were written in MATLAB R2021b, and run on a Lenovo PC (Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz) with 16 GB RAM memory.

Now, we consider the following quadratic program (QP), see Example 4.1 and Example 4.2.

Example 4.1 Given matrix $A \in R^{n\times n}$ and vector $b \in R^n$. We consider the problem (1), where $f(x) = \|x\|^2$, $h(y) = \left\|y\right\|^2$ and $n = 1000$, i.e.,

$$\begin{aligned}\min \quad & \|x\|^2 + \left\|y\right\|^2 \\ \text{s.t.} \quad & x + Ay = b,\end{aligned} \tag{30}$$

with

$$A = \left[a_{ij}\right] = \begin{cases} 2, & i = j, \\ -3, & |i-j| = 1, \end{cases} \text{ and } b = (1, 1, \ldots, 1, 1)^{\mathrm{T}},$$

and the ALF of (30) is given by

$$\hat{\mathcal{L}}(x, y, \lambda) = \|x\|^2 + \|y\|^2 - \lambda^{\mathrm{T}}(x + Ay - b) + \frac{\beta}{2}\|x + Ay - b\|^2.$$

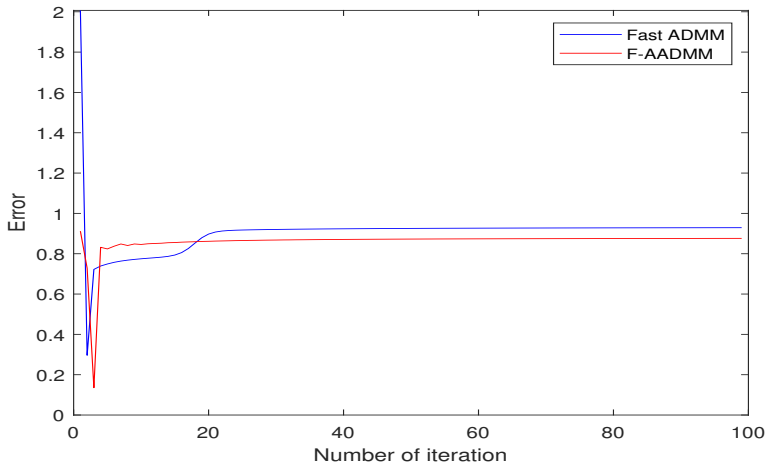

Figure 1: Numerical results of F-AADMM and Fast ADMM for Example 4.1.

In our computations, we set initial value $x_0 = (0, 0, \ldots, 0, 0)^{\mathrm{T}}$ and $y_0 = (1, 1, \ldots, 1, 1)^{\mathrm{T}}$ for $\hat{\mathcal{L}}(x, y, \lambda)$, $\lambda = (1, 0, \ldots, 0, 0)^{\mathrm{T}}$ and $\beta = 0.5$. The relationship between the number of iterations and the error (the error is defined as the relative error, i.e., $(f(x^k) + h(y^k) - (f(x^*) + h(y^*)))/(f(x^*) + h(y^*))$ is shown in Figure 1.

In Figure 1, we compare the performance of the F-AADMM method and the Fast ADMM method in terms of error and the number of iteration. It is found that the error of the F-AADMM method is significantly smaller than the Fast ADMM method in the first ten iterations. Further, we find that the error gradually tends to a stable value with the increase of the number of iteration. In other words, the error of F-AADMM is more sharper than Fast ADMM.

Example 4.2 Given matrix $A \in R^{n\times n}$, vector $e \in R^n$ and vector $b \in R^n$. We consider the problem (1), where $f(x) = \|x - e\|^2$, $h(y) = \|y - 2e\|^2$ and $n = 1000$, i.e.,

$$\begin{aligned} \min \quad & \|x - e\|^2 + \|y - 2e\|^2 \\ \text{s.t.} \quad & x + Ay = b, \end{aligned} \tag{31}$$

with

$$A = \left[a_{ij}\right] = \begin{cases} 2, & i = j, \\ 5, & i - j = 1, \\ -1, & j - i = 1, \end{cases} \quad e = (1, 1, \ldots, 1, 1)^{\mathrm{T}}, b = (1, 1, \ldots, 1, 1)^{\mathrm{T}},$$

and the ALF of (31) is given by

$$\hat{\mathcal{L}}(x, y, \lambda) = \|x - e\|^2 + \|y - 2e\|^2 - \lambda^{\mathrm{T}}(x + Ay - b) + \frac{\beta}{2}\|x + Ay - b\|^2.$$

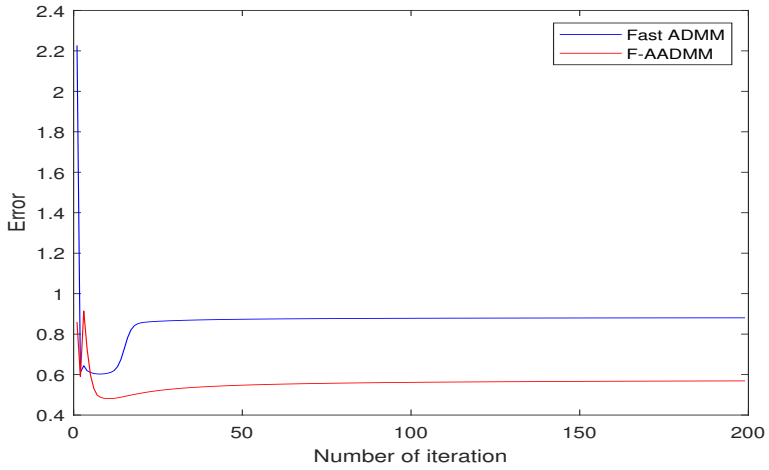

Figure 2: Numerical results of F-AADMM and Fast ADMM for Example 4.2.

In our computations, we set initial value $x_0 = (0, 0, \ldots, 0, 0)^{\mathrm{T}}$ and $y_0 = (1, 1, \ldots, 1, 1)^{\mathrm{T}}$ for $\hat{\mathcal{L}}(x, y, \lambda)$, $\lambda = (1, 0, \ldots, 0, 0)^{\mathrm{T}}$ and $\beta = 0.2$. The relationship between the number of iterations and the error (the error is defined as the relative error, i.e., $(f(x^k) + h(y^k) - (f(x^*) + h(y^*)))/(f(x^*) + h(y^*))$ is shown in Figure 2.

In Figure 2, similar to Example 4.1, we compare the performance of the F-AADMM method and the Fast ADMM method in terms of error and the number of iteration. It is found that the error gradually tends to a stable value with the increase of the number of iteration. Further, we find that the calculation error of our algorithm is significantly smaller than Fast ADMM. In other words, the error of F-AADMM is more sharper than Fast ADMM. This implies that for Example 4.2 the F-AADMM method achieves the higher accuracy than the Fast ADMM method. This further shows that our algorithm (F-AADMM) overmatches the Fast ADMM in a way.

5. Conclusions

In this paper, combining the Fast ADMM algorithm with the Gauss-Seidel method, we propose the F-AADMM method for solving the problem (1) and discuss the convergence property and convergence rate of the F-AADMM method. At the same time, the convergence conditions of the proposed method are given. Numerical examples are provided to illustrate the performance of the F-AADMM method.

References

[1] J. Bai, J. Li, F. Xu, H. Zhang, *Generalized symmetric ADMM for separable convex optimization*, Comput. Optim. Appl. **70** (2018), 129–170.
[2] S. P. Boyd, L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.
[3] T. Chan, R. Glowinski, *Finite element approximation and iterative solution of a class of mildly non-linear elliptic equations*, Computer Science Department, Stanford University, Stanford, 1978.
[4] Y. H. Dai, D. Han, X. Yuan, W. Zhang, *A sequential updating scheme of the Lagrange multiplier for separable convex programming*, Math. Comput. **86** (2017), 315–343.
[5] V. Edalatpour, D. Hezari, D. K. Salkuyeh, *A generalization of the Gauss-Seidel iteration method for solving absolute value equations*, Appl. Math. Comput. **293** (2017), 156–167.
[6] M. Fortin, R. Glowinski, *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*, Elsevier, 2000.
[7] D. Gabay, B. Mercier, *A dual algorithm for the solution of nonlinear variational problems via finite element approximation*, Comput. Math. Appl. **2** (1976), 17–40.
[8] R. Glowinski, A. Marroco, *Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de Dirichlet non linéaires*, ESAIM Math. Model. Num. **9** (1975), 41–76.

[9] T. Goldstein, B. O'Donoghue, S. Setzer, R. Baraniuk, *Fast alternating direction optimization methods*, SIAM J. Imaging Sci. **7** (2014), 1588–1623.
[10] D. Han, *A survey on some recent developments of alternating direction method of multipliers*, J. Oper. Res. Soc. China. **10** (2022), 1–52.
[11] B. He, X. Yuan, *On the* $O(1/n)$ *convergence rate of the Douglas-Rachford alternating direction method*, SIAM J. Numer. Anal. **50** (2012) 700–709.
[12] B. He, X. Yuan, *On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers*, Numer. Math. **130** (2015) 567–577.
[13] X. Li, D. Sun, K. C. Toh, *A block symmetric Gauss-Seidel decomposition theorem for convex composite quadratic programming and its applications*, Math. Program. **175** (2019), 395–418.
[14] J. P. Milaszewicz, *Improving jacobi and gauss-seidel iterations*, Linear Algebra Appl. **93** (1987), 161–170.
[15] R. D. C. Monteiro, B. F. Svaiter, *Iteration-complexity of block-decomposition algorithms and the alternating direction method of multipliers*, SIAM J. Optim. **23** (2013), 475–507.
[16] Y. E. Nesterov, *A method of solving a convex programming problem with convergence rate* $O(1/k^2)$, Dokl. Akad. Nauk. **269** (1983), 543–547.
[17] Y. E. Nesterov, *Introductory lectures on convex optimization: A basic course*, Springer Verlag, 2013.
[18] Y. Ouyang, Y. Chen, G. Lan, J. E. Pasiliao, *An accelerated linearized alternating direction method of multipliers*, SIAM J. Imaging Sci. **8** (2015), 644–681.
[19] Y. Shen, Y. Zuo, A. Yu, *A partially proximal S-ADMM for separable convex optimization with linear constraints*, Appl. Numer. Math. **160** (2021), 65–83.
[20] R. Y. Zhang, J. K. White, *GMRES-accelerated ADMM for quadratic objectives*, SIAM J. Optim. **28** (2018), 3025–3056.