Accelerated sampling Kaczmarz Motzkin algorithm for the linear feasibility problem

Md Sarowar Morshed1 · Md Saiful Islam1 · Md. Noor-E-Alam1

Received: 6 February 2019 / Accepted: 17 October 2019 / Published online: 24 October 2019 © Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract
The Sampling Kaczmarz Motzkin (SKM) algorithm is a generalized method for solving large-scale linear systems of inequalities. Having its root in the relaxation method of Agmon, Schoenberg, and Motzkin and the randomized Kaczmarz method, SKM outperforms the state-of-the-art methods in solving large-scale Linear Feasibility (LF) problems. Motivated by SKM’s success, in this work, we propose an Accelerated Sampling Kaczmarz Motzkin (ASKM) algorithm which achieves better convergence compared to the standard SKM algorithm on ill-conditioned problems. We provide a thorough convergence analysis for the proposed accelerated algorithm and validate the results with various numerical experiments. We compare the performance and effectiveness of ASKM algorithm with SKM, Interior Point Method (IPM) and Active Set Method (ASM) on randomly generated instances as well as Netlib LPs. In most of the test instances, the proposed ASKM algorithm outperforms the other state-of-the-art methods.

Keywords  Kaczmarz method · Nesterov’s acceleration · Motzkin method · Sampling Kaczmarz Motzkin algorithm

Mathematics Subject Classification  90C05 · 65F10 · 90C25 · 15A39 · 68W20

1 Introduction
We consider the problem of solving large-scale systems of linear inequalities:

\[ Ax \leq b, \quad b \in \mathbb{R}^m, \quad A \in \mathbb{R}^{m \times n} \]

(1)

Since, iterative methods are usually better suited to problems with large number of constraints compared to the number of variables, we confine the scope of this work to the \( m \gg n \) regime. We denote the rows of matrix \( A \) by \( a_i^T \) for \( i = 1, 2, \ldots, m \). In addition, we make the following assumptions: (1) the system is consistent, (2) matrix \( A \) has no zero rows and (3) the rows of \( A \)
are normalized (i.e. $\|a_i\| = 1$). It is worth noting that the last assumption is not a significantly important requirement for algorithmic efficiency, but it helps in the convergence analysis.

While most classical iterative methods are deterministic, recent works [1–8] suggest that randomization can play a huge role in the design of efficient algorithms for solving LF problems and randomized algorithms often perform better than existing deterministic methods. As shown in [9], randomized iterative methods can outperform state-of-the-art methods (i.e., IPM, ASM) for large-scale LF. In the field of large-scale optimization, mainly IPMs, there is a growing interest in approximate Newton-type methods [10–16] which use fast sub-schemes for calculating approximate solutions of large-scale Linear System (LS).

The Kaczmarz method for solving LS, discovered in 1937 [17], remained unnoticed to the western research community until the early 1980s, when it found an important application in the area of Algebraic Reconstruction Techniques (ART) for image reconstruction [18]. Since then it has been used for several other areas like digital signal processing, computer tomography, and belongs to a general category of methods including row-action, component solution, cyclic projection, and successive projection methods (see [19]). It gained immense popularity in the research community after the convergence analysis done in 2009 for the randomized version [1]. The convergence analysis of Strohmer [1] encouraged numerous extensions and generalizations of the randomized Kaczmarz method (see [2,3,5–7,20], for instance when we replace the equality constraints with inequality constraints we get a variant of the original problem.

Motzkin’s relaxation method is a variation of the Kaczmarz method which was introduced in the early 1950s [21,22] for solving systems of linear inequalities. Since then, it has been rediscovered several times. For instance, the famous perceptron algorithm in machine learning [23–25] can be thought of as a member of this family of methods. Additionally, the relaxation method has been referred to as the Kaczmarz method with the “most violated constraint control” or the “maximal-residual control” [19,26,27]. The rate of convergence of Motzkin’s method depends on step lengths and the so called Hoffman constants [21,28].

Combining both the Kaczmarz and Motzkin method together, the SKM algorithm proposed in [9] for solving LF problem given in (1) requires only $O(n)$ memory storage and it has a linear convergence rate. As shown by the authors, SKM is much more efficient than the state-of-the-art techniques such as IPMs, ASMs, and Kaczmarz Methods. Roughly, the SKM algorithm selects a row out of $\beta$ rows (sampled from $A$) by the maximum violation criterion (i.e. choose the row $i^*$ with $i^* = \arg \max_{i \in \tau_k} \{a_i^T x_k - b_i, 0\}$, $\beta = |\tau_k|$) and then updates the next point as follows:

$$x_{k+1} = x_k - \delta \frac{(a_{i^*}^T x_k - b_{i^*})^+}{\|a_{i^*}\|^2} a_{i^*}$$  (2)

In Eq. (2), $\delta$ can be $0 \leq \delta \leq 2$. Without the loss of generality, we consider $\delta = 1$ in this work.

The SKM method described in [9] overcomes the drawbacks of the individual methods (Kaczmarz, Motzkin) and combines their strengths. By selecting the maximum violated hyperplane from a sample, SKM achieves faster convergence compared to the randomized Kaczmarz method. In addition, per iteration computational cost is cheaper compared to Motzkin’s method.

Recently, Wright et al. [20] applied the acceleration scheme of Nesterov to the randomized Kaczmarz method. In a different work, Xu et al. [29] investigated the acceleration scheme in the context of the extended randomized Kaczmarz method for least square problems. Moreover, there is a recent work in applying Nesterov scheme in IPMs for solving large-scale
linear programming problems [30]. The above-mentioned works showed that the introduction of Nesterov’s acceleration scheme fasten the convergence of the original method.

In this work, we apply Nesterov’s acceleration scheme [31–35] to the generalized SKM algorithm. This can be seen as a generalized accelerated scheme for both randomized Kaczmarz method for solving linear systems as well as linear system of inequalities. It can be noted that with some modification, like the one stated in the work of Lewis et al. [2], we can apply this method to linear systems with both equality and inequality constraints. The overarching goal of this paper is to incorporate the ideas of the Kaczmarz method [1,17,36] for LS and Motzkin’s relaxation [9] for LF problem and develop an accelerated randomized scheme for solving large-scale LF problem.

The paper is organized as follows. The proposed algorithm is discussed in Sect. 2, and the convergence analysis of the proposed algorithm is given in Sect. 3. Extensive Numerical experiments performed on random and Netlib LP instances are provided in Sect. 4. And finally the paper is concluded with the conclusion in Sect. 5.

## 2 ASKM algorithm

### 2.1 Notation

We follow the standard notation in this work. For example, \( \mathbb{R} \) will be used to denote the set of real numbers. Matrix \( A \) with \( m \) rows and \( n \) columns belong to \( \mathbb{R}^{m \times n} \), with \( A_{ij} \) denoting the real-valued element in row \( i \) and column \( j \). \( A^T \) will be used to denote the transpose of matrix \( A \), with \( \text{tr}(A) \), \( \det(A) \), and \( \text{diag}(A) \) denoting the trace, determinant, and diagonal of matrix \( A \) respectively. \( I_n \) will be used as the \( n \times n \) identity matrix.

Furthermore, we use vectors \( 1 = [1 \ldots 1]^T \) and \( e_i \) as the standard \( i \)-th basis vector. A function \( f : X \mapsto Y \) maps its domain, \( \text{dom}(f) \subseteq X \), into set \( Y \). As it is customary, we use \( \nabla f \) and \( \nabla^2 f \) to represent the gradient and Hessian of \( f \). Finally, \( \langle x, y \rangle = x^T y \) denotes the standard inner product and \( \| x \| = \sqrt{\langle x, x \rangle} \) as the euclidean (\( L_2 \)) norm. \( \lambda_{\text{min}}, \lambda_{\text{max}} \) are set to be the minimum and maximum nonzero eigenvalues of \( A^T A \) respectively. \( \| A \| \) is the spectral norm of the matrix \( A \) and \( \| A \|_F \) denote the Frobenius norm. Moreover, \( A^\dagger \) is the Moore-Penrose pseudoinverse of \( A \) and the corresponding compact singular value decomposition of \( A \in \mathbb{R}^{m \times n} \) as \( A = U \Sigma V^T \), where \( U, V \) are unitary matrices with appropriate size and \( \Sigma \) is the non-singular and diagonal matrix with singular value on the diagonal. Throughout the paper, we denote \( \zeta \) as the condition number of matrix \( A \). The notation \( P_{A,b}(x) \) denotes the Euclidean norm projection of \( x \) onto the feasible region of \( Ax \leq b \).

In this section, we review the proposed SKM algorithm in [9] (Algorithm 1) and then based on the motivation from the accelerated randomized Kaczmarz algorithm in [20] and accelerated extended Kaczmarz algorithm in [29], we develop ASKM algorithm.

Now, we propose to use the acceleration scheme discovered by Nesterov [31–35] in the SKM algorithm framework to achieve second order convergence rate as compared to the linear rate shown in [9]. The ASKM algorithm uses the acceleration procedure [33], which is more famous in the context of gradient descent algorithm. Note that, Nesterov’s acceleration scheme constructs sequences \( \{ y_k \} \) and \( \{ v_k \} \) using Eq. (3).
Algorithm 1 SKM Algorithm: $x_{k+1} = \text{SKM}(A, b, x_0, K, \delta, \beta)$

Initialize $k \leftarrow 0$;

while $k \leq K$

Choose a sample of $\beta$ constraints, $\tau_k$, uniformly at random from the rows of matrix $A$.

From these $\beta$ constraints, choose $i^* = \arg \max_{i \in \tau_k} \{a_i^T x_k - b_i \geq 0\};$

Update $x_{k+1} = x_k - \delta \frac{(a_{i^*}^T x_k - b_{i^*})}{\|a_{i^*}\|^2} a_{i^*};$

$k \leftarrow k + 1;$

end while

return $x$

Algorithm 2 ASKM Algorithm: $x_{k+1} = \text{ASKM}(A, b, x_0, K, \beta, \lambda_{\text{min}}, \zeta)$

Choose $\lambda \in [0, \lambda_{\text{min}}];$

Initialize $v_0 \leftarrow x_0, \gamma_{-1} \leftarrow 0, k \leftarrow 0$;

while $k \leq K$

Choose $\gamma_k$ to be the larger root of
\[ \gamma_k^2 - \frac{\zeta}{m} \gamma_k = \frac{d}{\beta} \left( 1 - \frac{\lambda \beta}{m} \gamma_k \right) \gamma_{k-1}^2 \]  \hspace{1cm} (4)

Update $\alpha_k$ and $\beta_k$ as follows:
\[ \alpha_k = \frac{\xi (m - \lambda \beta \gamma_k)}{\gamma_k (m^2 - \lambda \xi \beta)} \]  \hspace{1cm} (5)
\[ \beta_k = 1 - \frac{\lambda \beta}{m} \gamma_k \] \hspace{1cm} (6)

Update $y_k = \alpha_k v_k + (1 - \alpha_k) x_k$;

Choose a sample of $\beta$ constraints, $\tau_k$, uniformly at random from the rows of matrix $A$. From these $\beta$ constraints, choose $i^* = \arg \max_{i \in \tau_k} \{a_i^T x_k - b_i \geq 0\};$

Update $y_{k+1} = y_k - \zeta \nabla f(y_k)$
\[ v_{k+1} = \beta_k v_k + (1 - \beta_k) y_k - \gamma_k \frac{(a_{i^*}^T x_k - b_{i^*})}{\|a_{i^*}\|^2} a_{i^*}; \] \hspace{1cm} (8)

$k \leftarrow k + 1;$

end while

return $x$

In Eq. (3), $\nabla f$ is the gradient of the given function and $\theta_k$ is the step-size. The main contribution for the above scheme is that it uses appropriate values for the parameters $\alpha_k, \beta_k, \gamma_k$, which in turn yield better convergence in the context of standard gradient descent. Now, using
the general setup of Nesterov’s scheme [35] for coordinate descent and the idea in [20], we
developed ASKM algorithm shown above (Algorithm 2).

3 Convergence analysis

In this section, we analyze the convergence of the proposed ASKM algorithm 2. Throughout
the analysis, we make the assumptions: 1) \( \|a_i\| = 1 \) for any \( i \in m \), which implies \( \|A\|_F^2 = m \) and 2) \( P_{A,b} \) is full dimensional. The following convergence result was proven in [9] for the
SKM algorithm (Algorithm 1):

\[
E \left[ \|x_{k+1} - P_{A,b}(x_k)\|^2 \right] \leq \left( 1 - \frac{2\delta - \delta^2}{V_k L^2} \right) \|x_k - P_{A,b}(x_0)\|^2 \leq \left( 1 - \frac{2\delta - \delta^2}{mL^2} \right) \|x_k - P_{A,b}(x_0)\|^2 \quad (9)
\]

In the above equation, \( L \) is the Hoffman constant and \( V_k \) is defined in the proof of Lemma 3. For the ASKM algorithm (Algorithm 2) shown above, we prove a better convergence result as stated in Theorem 1 compared to the one in Eq. (9) (we consider the case \( \delta = 1 \)).

**Remark 1** This framework for convergence in the context of acceleration follows the general idea developed by Nesterov [32] for the Gradient Descent method. The proof of Theorem 1 follows the generalized sketch developed by Nesterov [35] for proving the convergence result of Coordinate Descent method. Due to the similarity of acceleration methods derived in [20] for the randomized Kaczmarz method and our proposed method, we will use the same standard notation on this subject. In addition to that, the following results generalize results for acceleration in Kaczmarz types methods (i.e. if we select \( \beta = 1 \) and use linear systems, we get the same results shown in [20]).

**Theorem 1** The ASKM algorithm defined above with \( \lambda \in [0, \lambda_{\text{min}}] \) and \( \sigma_1 = 1 + \frac{\sqrt{\lambda \beta \zeta}}{2m}, \sigma_2 = 1 - \frac{\sqrt{\lambda \beta \zeta}}{2m} \), then for all \( k \geq 0 \) we have the following:

\[
E \left[ \|v_{k+1} - x^*\|_{(AR)^t}^2 \right] \leq \frac{4\|x_0 - x^*\|_{(AR)^t}^2}{\sigma_1^{k+1} + \sigma_2^{k+1}} \quad (10)
\]

\[
E \left[ \|x_{k+1} - x^*\|_{(AR)^t}^2 \right] \leq \frac{4\lambda\|x_0 - x^*\|_{(AR)^t}^2}{\zeta (\sigma_1^{k+1} - \sigma_2^{k+1})} \quad (11)
\]

Here, \( x^* \in \mathbb{R}^n \) is a unique limit point of the ASKM iterates (for the uniqueness of \( x^* \) see Lemmas 2.2–2.4 in [9]), \( \zeta \) is the condition number of matrix \( A \) and \( \beta \) is the sample size of the random sampling process.

Before delving into Theorem 1, we start with the proof of some useful lemmas. For the expectation calculation of the random process described in our algorithm, we need a certain setup. Let, \( (Ax_k - b)^+ \) denote the \( (\beta + j) \)th smallest entry of the residual vector (if we order the entries of \( (Ax_k - b)^+ \) from smallest to largest, \( (Ax_k - b)^+ )_i \) is the entry in \( (\beta + j) \)th position). Now, if we consider the size of all the entries of the residual vector \( (Ax_k - b)^+ \), we can calculate the probability that a particular entry of the residual vector is selected.
this case, each sample has equal probability of selection (i.e., \( \frac{1}{m} \)). Moreover, the size of the residual vector controls the frequency that each entry of the residual vector will be expected to be selected (Algorithm 2, sample of constraints selection). For example, if we have only one sample then \( \beta \)th smallest entry will be selected and for the case of all samples, \( m \)th smallest entry will be selected. Therefore, if we expand the expectation of the residual (with respect to the probabilistic choice of sample constraints, \( \tau_j \), of size \( \beta \)), we get the following:

\[
E_{\beta_\ast} \left[ \left\| \left( a_i^T y - b_i \right)^{+}_{i_\ast} \right\|^2 \right] = \frac{1}{m} \sum_{k=0}^{m-\beta} \left( \frac{\beta - 1 + k}{\beta - 1} \right)^2 \left\| \frac{(a_i^T y - b_i)^{+}_{i_k}}{\beta} \right\|^2
\]

(12)

where \( E_{\beta_\ast} \) denotes the required expectation in accordance with the above sampling process (\( \beta \) is the sample size).

**Lemma 1** For any \( y \in \mathbb{R}^n \), we have the following:

\[
E_{\beta_\ast} \left[ \left\| a_{i_\ast}^T \left( a_i^T y - b_i \right)^{+}_{i_\ast} \right\|^2 \left( A^T A \right)^{\dagger} \right] \leq \frac{\beta}{m} \left\| (Ay - b)^{+} \right\|^2
\]

(13)

**Proof** Let us define the singular value decomposition of \( A \) as \( A = U \Sigma V^T \), where both \( U \) and \( V \) are unitary matrices of appropriate dimension and \( \Sigma \) is a positive diagonal. We can easily show that \( (A^T A)^{\dagger} = V \Sigma^{-2} V^T \). Then, with the defined orientation above, we have the following:

\[
E_{\beta_\ast} \left[ \left\| a_{i_\ast}^T \left( a_i^T y - b_i \right)^{+}_{i_\ast} \right\|^2 \left( A^T A \right)^{\dagger} \right] = \frac{1}{m} \sum_{k=0}^{m-\beta} \left( \frac{\beta - 1 + k}{\beta - 1} \right)^2 \left\| a_{i_k} \right\|^2 \left( a_i^T y - b_i \right)^{+}_{i_k}
\]

\[
\leq \frac{(m-1)}{m} \sum_{k=0}^{m-\beta} \left\| a_{i_k} \right\|^2 \left( a_i^T y - b_i \right)^{+}_{i_k}
\]

\[
\leq \frac{\beta}{m} \sum_{j=1}^{m} \left\| a_j \right\|^2 \left( a_j^T y - b_j \right)^{+}_{j}
\]

\[
= \frac{\beta}{m} \sum_{j=1}^{m} \left( (A^T A)^{\dagger} a_j \left( a_j^T y - b_j \right)^{+}_{j} , a_j \left( a_j^T y - b_j \right)^{+}_{j} \right)
\]

\[
= \frac{\beta}{m} \text{Tr} \left[ (A^T A)^{\dagger} a_j \left( a_j^T y - b_j \right)^{+}_{j} a_j \left( a_j^T y - b_j \right)^{+}_{j} \right]
\]

\[
= \frac{\beta}{m} \text{Tr} \left[ (A^T A)^{\dagger} A^T D^2 \left[ (Ay - b)^{+} \right] A \right]
\]

\[
= \frac{\beta}{m} \text{Tr} \left[ V \Sigma^{-2} V^T V \Sigma U^T D^2 \left[ (Ay - b)^{+} \right] U \Sigma V^T \right]
\]

\[
= \frac{\beta}{m} \text{Tr} \left[ U^T D^2 \left[ (Ay - b)^{+} \right] U \right]
\]

\[
= \frac{\beta}{m} \left\| D \left[ (Ay - b)^{+} \right] U \right\|^2_F
\]
\[= \frac{\beta}{m} \sum_{j=1}^{m} \left| \left( a_j^T y - b_j \right)^+ \right|^2 \| U_j \|_2^2 \]

\[\leq \frac{\beta}{m} \sum_{j=1}^{m} \left| \left( a_j^T y - b_j \right)^+ \right|^2 = \frac{\beta}{m} \| (Ay - b)^+ \|^2 \]

This proves Lemma 1. \(\square\)

**Lemma 2** For any \( y \in \mathbb{R}^n \), we have the following:

\[\mathbb{E}_{\beta_i^*} \left[ \left\| a_i^* \left( a_i^T y - b_i \right)^+ \right\|_2^2 \right] \leq \frac{\beta}{m} \| (Ay - b)^+ \|^2 \] \( (14) \)

**Proof** With the expression of expectation defined in (12) we have,

\[\mathbb{E}_{\beta_i^*} \left[ \left\| a_i^* \left( a_i^T y - b_i \right)^+ \right\|_2^2 \right] = \frac{1}{(m\beta)^k} \sum_{k=0}^{m-\beta} \left( \frac{\beta - 1 + k}{\beta - 1} \right) \left\| a_i_k \right\|^2 \left( \left( a_i^T y - b_i \right)^+_i \right)^2 \]

\[\leq \frac{\beta}{m} \sum_{k=0}^{m-\beta} \left( \left( a_i^T y - b_i \right)^+_i \right)^2 \]

This proves the Lemma 2. \(\square\)

**Lemma 3** For any \( y \in \mathbb{R}^n \) and \( x^* \) that satisfies \( Ax^* \leq b \), we have the following:

\[\frac{1}{m} \| (Ay - b)^+ \|^2 \leq \| y - x^* \|^2 - \mathbb{E}_{\beta_i^*} \left[ \left\| P_{a_i^*, b_i^*} (y) - x^* \right\|_2^2 \right] \] \( (15) \)

**Proof** Let us define \( P \) as the projection operator onto the feasible region \( P = \{ x \in \mathbb{R}^n \mid Ax \leq b \} \), and denote \( s_k \) as the number of zero entries in the residual \( (Ax_k - b)^+ \), which also corresponds to number of satisfied constraints. We also define \( V_j = \max \{ (m - s_j, m - \beta + 1) \} \). Now, from the update formula shown in Algorithm 2, we know that \( x_{k+1} = y_k - \frac{\left( A_{tk} y_k - b_{tk} \right)^+}{\| a_i^* \|^2} a_i^* \); where,

\[ i^* = \arg \max_{j \in \mathbb{T}_k} \{ a_j^T x_k - b_j, 0 \} = \arg \max_{j \in \mathbb{T}_k} \left( A_{tk} x_k - b_i \right)^+ \] \( (16) \)

Then we have,

\[\| x_{k+1} - P \|^2 = \| x_{k+1} - P(x_{k+1}) \|^2 \leq \| x_{k+1} - P(y_k) \|^2 \]

\[\| y_k - \frac{\left( A_{tk} y_k - b_{tk} \right)^+}{\| a_i^* \|^2} a_i^* - P(y_k) \|^2 \]

\[\| y_k - \left( a_i^* x_k - b_i \right) \|^2 + \left( \| a_i^* \|^2 a_i^* \right)^2 - 2 \left( A_{tk} y_k - b_{tk} \right)^+ \left( a_i^* y_k - P(y_k) \right) \]

\[\leq \| y_k - P \|^2 + \left( \| a_i^* \|^2 a_i^* \right)^2 - 2 \left( A_{tk} y_k - b_{tk} \right)^+ \left( a_i^* y_k - P(y_k) \right) \]

\[\| y_k - P \|^2 - \left( \| a_i^* \|^2 a_i^* \right)^2 = \| y_k - P \|^2 - \| \left( A_{tk} y_k - b_{tk} \right)^+ \|^2 \] \( (17) \)
Now, taking expectation in both sides of Eq. (17), we have,
\[
\mathbb{E}_{\beta_i^*} \left[ \| x_{k+1} - \mathcal{P} \|^2 \right] = \| y_k - x^* \|^2 - \mathbb{E}_{\beta_i^*} \left[ \| (A_{\tau_k} y_k - b_{\tau_k})_i^+ \|^2 \right]
\]
\[
= \| y_k - x^* \|^2 - \frac{1}{(m+\beta)} \sum_{j=0}^{m-\beta} \left( \beta - 1 + j \right) \left[ (Ay_k - b)_i^+ \right]^2
\]
\[
\leq \| y_k - x^* \|^2 - \frac{1}{m-\beta+1} \sum_{j=0}^{m-\beta} \left[ (Ay_k - b)_i^+ \right]^2
\]
\[
\leq \| y_k - x^* \|^2 - \frac{1}{m-\beta+1} \min \left\{ \frac{m - \beta + 1}{m - s_k}, 1 \right\} \sum_{i=1}^{m} |(Ay_k - b)_i^+|^2
\]
\[
\leq \| y_k - x^* \|^2 - \frac{1}{m} \| (Ay_k - b)^+ \|^2
\]
The expectation above proves the Lemma 3. \hfill \square

**Definition** Let us define a function \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) as follow:
\[
f(x) = \frac{1}{2} \left[ (Ax - b)^+ \right]^T A \left( (A^T A)^+ \right)^2 A^T \left( (Ax - b)^+ \right)
\]
where \( \sigma_{\max}(A) = \frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)} \).

**Lemma 4** For any \( x, y \in \mathbb{R}^n \) and condition number of A matrix \( \zeta = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)} = \frac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)} \), we have the following:
\[
\left\langle \nabla f(x), y - x \right\rangle \leq f(y) - f(x) + \frac{\zeta}{2} \| x - y \|^2
\]

**Proof** We first prove that \( \nabla f \) is Lipschitz continuous with the constant \( \zeta \). Using the definition given in (18), for any \( x, y \in \mathbb{R}^n \) we have,
\[
\| \nabla f(x) - \nabla f(y) \| = \| (A^T A)^+ A^T (Ax - b)^+ \| \leq \| (A^T A)^+ A^T \| \| (Ax - b)^+ - (Ay - b)^+ \| \leq \| A^+ \| \| A \| \| x - y \| = \zeta \| x - y \|
\]
The above equation shows that \( \nabla f \) is Lipschitz continuous with the constant \( \zeta \). Here, we use the common expression \( (A^T A)^+ A^T = (A)^+ \). Now using Lemma 1.2.3 proven in [33], as \( \nabla f \) is Lipschitz continuous, for any \( x, y \in \mathbb{R}^n \) we can write the following:
\[
| f(y) - f(x) - \langle \nabla f(x), y - x \rangle | \leq \frac{\zeta}{2} \| x - y \|^2
\]
Now, by simplifying (20), we get the following bound:
\[
\left\langle \nabla f(x), y - x \right\rangle \leq f(y) - f(x) + \frac{\zeta}{2} \| x - y \|^2
\]
The bound mentioned above proves the Lemma 4. \hfill \square
Lemma 5 For any \( m^2 > \zeta \lambda \beta \) and with the following definitions:

\[
\beta_k = 1 - \frac{\gamma_k \lambda \beta}{m}, \quad \alpha_k = \frac{\zeta (m - \lambda \beta \gamma_k)}{\gamma_k (m^2 - \zeta \lambda \beta)}
\]

both sequences \( \{\alpha_k\}, \{\beta_k\} \) lies in the interval \([0, 1]\) if and only if \( \gamma_k \) satisfies the following property:

\[
\frac{\zeta}{m} \leq \gamma_k \leq \frac{m}{\lambda \beta}
\]

Proof The proof of Lemma 5 is straightforward. If we consider the definitions of the sequences \( \{\alpha_k\}, \{\beta_k\} \) with the given condition, we find that \( \alpha_k, \beta_k \in [0, 1] \) implies that the following bound must hold:

\[
\frac{\zeta}{m} \leq \gamma_k \leq \frac{m}{\lambda \beta}
\]

Conversely, if we assume the bound holds for \( \gamma_k \), then we can easily find that it implies the sequences \( \{\alpha_k\} \) and \( \{\beta_k\} \) lies in the interval \([0, 1]\). \( \square \)

Lemma 6 For any \( d \geq \beta \), if \( \gamma_{k-1} \leq \sqrt{\frac{\zeta}{\lambda d}} \) holds, then \( \gamma_k \) satisfies the bound in Lemma 5 and also \( \gamma_k \) lies in the interval \( [\gamma_{k-1}, \sqrt{\frac{\zeta}{\lambda d}}] \).

Proof Let us define the function \( g : \mathbb{R} \rightarrow \mathbb{R} \) as follows:

\[
g(\gamma) = \gamma^2 + \frac{\gamma}{m} (\lambda d \gamma_{k-1}^2 - \zeta) - \frac{d}{\beta} \gamma_{k-1}^2
\]

As we know from the definition, \( \gamma_k \) is the largest root of \( g(\gamma) \), then it satisfies \( g(\gamma_k) = 0 \). Now we have,

\[
g\left(\frac{\zeta}{m}\right) = \frac{\zeta^2}{m^2} + \frac{d \zeta}{m^2} \lambda \gamma_{k-1}^2 - \frac{\zeta^2}{m^2} - \frac{d}{\beta} \gamma_{k-1}^2
\]

\[
= \frac{d \gamma_{k-1}^2}{\beta m^2} (\lambda \zeta \beta - m^2) \leq 0 \quad (m^2 > \zeta \lambda \beta)
\]

Similarly,

\[
g\left(\frac{m}{\lambda \beta}\right) = \frac{m^2}{\lambda^2 \beta^2} - \frac{\zeta m}{\lambda \beta} - \frac{d}{\beta} \gamma_{k-1}^2 + \frac{\lambda d \gamma_{k-1}^2}{m \lambda \beta}
\]

\[
= \frac{m^2 - \lambda \zeta \beta}{\lambda^2 \beta^2} \geq 0
\]

Therefore, we can write,

\[
\frac{\zeta}{m} \leq \gamma_k \leq \frac{m}{\lambda \beta}
\]

This proves the first part of the Lemma. For the second part, notice that, assuming \( \beta \leq d \) we have the following:

\[
g(\gamma_{k-1}) = \gamma_{k-1}^2 + \gamma_{k-1} \frac{1}{m} (\lambda d \gamma_{k-1}^2 - \zeta) - \frac{d}{\beta} \gamma_{k-1}^2
\]

\[
= \frac{\gamma_{k-1}^2}{\beta} (\beta - d) + \frac{\gamma_{k-1}}{m} (\lambda d \gamma_{k-1}^2 - \zeta) \leq 0
\]
Here, the last inequality follows from the assumed condition $\gamma_{k-1} \leq \sqrt{\frac{\zeta}{\lambda d}}$. In a similar fashion we have,

$$
g\left(\sqrt{\frac{\zeta}{\lambda d}}\right) = \frac{\zeta}{\lambda \beta} - \frac{\zeta}{m \sqrt{\frac{\zeta}{\lambda d}}} - \frac{d}{\beta} \gamma_{k-1}^2 + \frac{\lambda d}{m} \sqrt{\frac{\zeta}{\lambda d}} \gamma_{k-1}^2 \geq \frac{\zeta}{\lambda \beta} - \frac{d}{\beta} \gamma_{k-1}^2 + \frac{\lambda d \zeta}{m^2} \gamma_{k-1}^2 - \frac{\zeta^2}{m^2} = \frac{m^2 (\zeta - \lambda d \gamma_{k-1}^2) - \lambda \beta \zeta (\zeta - \lambda d \gamma_{k-1}^2)}{\lambda m^2 \beta} \geq 0
$$

In this case, we use the identity $\sqrt{\frac{\zeta}{\lambda d}} > \frac{\zeta}{m}$ and $\gamma_{k-1} \leq \sqrt{\frac{\zeta}{\lambda d}}$. This proves the statement, $\gamma_k \in [\gamma_{k-1}, \sqrt{\frac{\zeta}{\lambda d}}]$. \hfill $\square$

**Remark 2** Note that by taking limits as $\lambda \rightarrow 0^+$ in Theorem 1 we have,

$$
\lim_{\lambda \rightarrow 0^+} \frac{4\lambda \|x_0 - x^*\|_2^2}{\sigma_{1}^{k+1} - \sigma_{2}^{k+1}} = \lim_{\lambda \rightarrow 0^+} \frac{4\lambda \|x_0 - x^*\|_2^2}{\zeta \left(1 + (k + 1) \frac{\sqrt{\lambda \beta \zeta}}{2m} + o\left(\sqrt{\lambda \beta \zeta}\right)\right) - (1 - (k + 1) \frac{\sqrt{\lambda \beta \zeta}}{2m} + o\left(\sqrt{\lambda \beta \zeta}\right))} \geq \frac{4m^2 \|x_0 - x^*\|_2^2}{\beta^2 (k + 1)^2}
$$

Therefore, we can conclude that when $\lambda > 0$, the ASKM algorithm converges with a linear rate. When $\lambda = 0$, we get a sublinear convergence. But for the case of $\lambda \rightarrow 0^+$, we get a quadratic convergence, which is consistent with the convergence rate of the original accelerated algorithm of Nesterov [33] and also with the Accelerated Randomized Kaczmarz algorithm proposed in [20]. Furthermore, if we take $\beta = 1$, we get exactly the same convergence theorem proven in [20].

**Proof (Theorem 1)** The proof of theorem 1 is general in the context of acceleration. We follow the standard notation and steps shown in [20,35]. Using the definitions given in Lemma 5, we note that the following relation holds:

$$
\frac{1 - \alpha_k}{\alpha_k} = \frac{m \gamma_{k-1}^2}{\gamma_k} \quad (23)
$$
Now, let us define \( r^2_k = \|v_k - x^*\|_{(A^T A)^\dagger}^2 \). We can write,
\[
\begin{align*}
 r^2_{k+1} &= \|v_{k+1} - x^*\|_{(A^T A)^\dagger}^2 \\
 &= \|\beta_k v_k + (1 - \beta_k) y_k - y_k a_i^* (a_i^T y_k - b_i^*)^+ - x^*\|_{(A^T A)^\dagger}^2 \\
 &= \|\beta_k v_k + (1 - \beta_k) y_k - x^*\|_{(A^T A)^\dagger}^2 + \gamma_k^2 \parallel a_i^* (a_i^T y_k - b_i^*)^+ \parallel_{(A^T A)^\dagger}^2 \\
 &\quad - 2\gamma_k \beta_k v_k + (1 - \beta_k) y_k - x^*, (A^T A)^\dagger a_i^* (a_i^T y_k - b_i^*)^+ \\
\end{align*}
\]

(24)

Now, we divide the RHS of Eq. (24) into three parts and simplify them separately. Since \( \|\cdot\|_{(A^T A)^\dagger}^2 \) is a convex function and \( 0 \leq \beta_k \leq 1 \), 1\(^{st}\) part of (24) satisfies the following inequality:
\[
\begin{align*}
\|\beta_k v_k + (1 - \beta_k) y_k - x^*\|_{(A^T A)^\dagger}^2 \\
&\leq \beta_k \|v_k - x^*\|_{(A^T A)^\dagger}^2 + (1 - \beta_k) \|y_k - x^*\|_{(A^T A)^\dagger}^2 \\
&\leq \beta_k \|v_k - x^*\|_{(A^T A)^\dagger}^2 + \frac{(1 - \beta_k)}{\lambda} \|y_k - x^*\|^2 \\
\end{align*}
\]

(25)

Let us denote \( i(k) \) as the index which represents the random selection at iteration \( k \). And let \( I(k) \) denote all random indices occurred before or at iteration \( k \), i.e.,
\[
I(k) = \{i(k), i(k - 1), \ldots, i(0)\} 
\]

(26)

The sequences \( x_{k+1}, y_{k+1}, v_{k+1} \) are dependent on \( I(k) \). In the next part of the proof, we use \( E_{I(k)\mid I(k-1)[\cdot]} \) to represent the expectation of a random variable conditioned on \( I(k - 1) \) with respect to the index \( i(k) \). Note that,
\[
E_{I(k)[\cdot]} = E_{I(k-1)} \left[ E_{I(k)\mid I(k-1)[\cdot]} \right] 
\]

(27)

Also note that, from now on we use \( E \) instead (\( E_{\beta_1\cdot} \)) to denote the expectation. Now, based on the Lemmas 2 and 3, we can write the 2\(^{nd}\) part of (24) as follows:
\[
\begin{align*}
\mathbb{E}_{i(k)\mid I(k-1)} \left[ \|a_i^* (a_i^T y_k - b_i^*)^+\|_{(A^T A)^\dagger}^2 \right] \\
\leq \frac{\beta}{m} \|A y_k - b^+\|^2 \\
\leq \beta \|y_k - x^*\|^2 - \beta \mathbb{E} \left[ \|P_{a_i^* b_i^*} (y_k) - x^*\|^2 \right] \\
\leq \beta \|y_k - x^*\|^2 - \beta \mathbb{E}_{i(k)\mid I(k-1)} \left[ \|x_{k+1} - x^*\|^2 \right] \\
\end{align*}
\]

(28)

Now, by using the definitions of the sequences \( \{\alpha_k\}, \{\beta_k\} \) and \( \{\gamma_k\} \), we can simply show that the following identity holds:
\[
\frac{\zeta \beta}{m} 1 - \frac{\alpha_k}{\beta_k} \zeta \beta = d \frac{\alpha_k}{\beta_k} \frac{\gamma_{k-1}^2}{\gamma_k} 
\]

(29)
We use the identity of (29) in the next part of our proof. After taking expectation in the third term of equation (24), we get,

\[
\mathbb{E}_{i(k)\mid I(k-1)} \left[ 2\gamma_k \left( x^* - y_k + \frac{1 - \alpha_k}{\alpha_k} \beta_k (x_k - y_k), (A^T A)^\dagger a_i^* \left( a_{i*}^T y_k - b_{i*} \right) \right)^+ \right] \\
= 2\gamma_k \left( x^* - y_k + \frac{1 - \alpha_k}{\alpha_k} \beta_k (x_k - y_k), (A^T A)^\dagger \mathbb{E}_{i(k)\mid I(k-1)} \left[ a_{i*}^T (a_{i*}^T y_k - b_{i*}) \right]^+ \right) \\
= 2\gamma_k \left( x^* - y_k + \frac{1 - \alpha_k}{\alpha_k} \beta_k (x_k - y_k), (A^T A)^\dagger \sum_{j=0}^{m-\beta} \left( \beta - 1 + j \right) a_{ij} \left( a_{ij}^T y_k - b_{ij} \right)^+ \right) \\
\leq \frac{2\gamma_k \beta}{m} \left( x^* - y_k + \frac{1 - \alpha_k}{\alpha_k} \beta_k (x_k - y_k), (A^T A)^\dagger A^T (A y_k - b)^+ \right) \\
\tag{30}
\]

Using the definition of the function \( f(.) \) defined in (18) and denoting \( z_k = \beta_k \frac{1 - \alpha_k}{\alpha_k} \), we get,

\[
\left( x^* - y_k + z_k (x_k - y_k), (A^T A)^\dagger A^T (A y_k - b)^+ \right) \\
= \left( x^* + z_k (x_k - y_k) - y_k, \nabla f(y_k) \right) \\
\leq f \left( x^* + z_k (x_k - y_k) - y_k \right) - f(y_k) - \frac{\zeta}{2} \| x^* + z_k (x_k - y_k) - y_k \|^2 \\
\leq \frac{\zeta^2}{2} \left( (A x_k - A y_k)^+ \right)^T A (A^T A)^\dagger A^T (A x_k - A y_k)^+ + \zeta z_k \langle y_k - x^*, x_k - y_k \rangle \\
- \frac{\zeta}{2} z_k^2 \| x_k - y_k \|^2 - \frac{\zeta}{2} \| y_k - x^* \|^2 - \frac{1}{2} \| (A y_k - b)^+ \|^2 A (A^T A)^\dagger A^T (A y_k - b)^+ \\
\leq \frac{1}{2} \| x_k - y_k \|^2 \left[ \lambda_{\text{max}} z_k^2 - \zeta z_k^2 - \zeta z_k \right] + \frac{\zeta z_k}{2} \| x_k - x^* \|^2 \\
- \frac{1}{2} \| y_k - x^* \|^2 \left[ 1 + \zeta + \zeta z_k \right] \\
\leq - \frac{1}{2} \zeta z_k \| x_k - y_k \|^2 + \frac{\zeta}{2} z_k \| x_k - x^* \|^2 - \frac{1}{2} \| y_k - x^* \|^2 \left[ 1 + \zeta + \zeta z_k \right] \\
+ \frac{1}{2} \frac{\zeta \lambda_{\text{max}}}{\lambda_{\text{min}}} \| x_k - y_k \|^2 \left[ \lambda_{\text{min}}^2 - 1 \right] \\
\leq - \frac{\zeta}{2} z_k \| x_k - y_k \|^2 + \frac{\zeta}{2} z_k \| x_k - x^* \|^2 - \frac{1}{2} \| y_k - x^* \|^2 \left[ 1 + \zeta + \zeta z_k \right] \\
\tag{31}
\]

Now, substituting Eq. (31) in (30) with the known identity, \( \frac{\zeta \beta y_k}{m} z_k = d \beta_k y_{k-1}^2 \), we have,

\[
\mathbb{E}_{i(k)\mid I(k-1)} \left[ 2\gamma_k \left( x^* - y_k + \frac{1 - \alpha_k}{\alpha_k} \beta_k (x_k - y_k), (A^T A)^\dagger a_i^* \left( a_{i*}^T y_k - b_{i*} \right) \right)^+ \right] \\
\leq \frac{2\gamma_k \beta}{m} \left( x^* - y_k + \frac{1 - \alpha_k}{\alpha_k} \beta_k (x_k - y_k), (A^T A)^\dagger A^T (A y_k - b)^+ \right) \\
\leq - \frac{\zeta \beta y_k}{m} z_k \| x_k - y_k \|^2 + \frac{\zeta \beta y_k}{m} z_k \| x_k - x^* \|^2 - \frac{\beta y_k}{m} \| y_k - x^* \|^2 \left[ 1 + \zeta + \zeta z_k \right] \\
\leq d \beta_k \gamma_{k-1} \| x_k - x^* \|^2 - \frac{\beta y_k}{m} \| y_k - x^* \|^2 \left[ 1 + \zeta + \zeta z_k \right] \\
\tag{32}
\]
Now by substituting all three parts of (25), (28) and (32) in Eq. (24), we get,

\[
\mathbb{E}_{i(k)|I(k-1)}(r_{k+1}^2) = \beta_k \| v_k - x^* \|^2_{(A^T A)^+} + \frac{\beta}{m} \gamma_k \| y_k - x^* \|^2 + \beta \gamma_k^2 \| y_k - x^* \|^2 \\
- \beta \gamma_k^2 \mathbb{E}_{i(k)|I(k-1)}(\| x_{k+1} - x^* \|^2) + d \beta_k \gamma_{k-1}^2 \| x_k - x^* \|^2 \\
- \frac{\beta \gamma_k}{m} \| y_k - x^* \|^2 [1 + \zeta + \zeta z_k] \\
\leq \beta_k \| v_k - x^* \|^2_{(A^T A)^+} - \beta \gamma_k^2 \mathbb{E}_{i(k)|I(k-1)}(\| x_{k+1} - x^* \|^2) \\
+ \beta \left( \gamma_k^2 - \frac{\zeta}{m} \gamma_k - \frac{d}{\beta} \beta_k \gamma_{k-1}^2 \right) \| y_k - x^* \|^2 + d \beta_k \gamma_{k-1}^2 \| x_k - x^* \|^2 \\
= \beta_k \| v_k - x^* \|^2_{(A^T A)^+} + d \beta_k \gamma_{k-1}^2 \| x_k - x^* \|^2 \\
- \beta \gamma_k^2 \mathbb{E}_{i(k)|I(k-1)}(\| x_{k+1} - x^* \|^2) \\
(33)
\]

From now on, we will assume \( d = \beta \), which will simplify our algorithm. Let us define two sequences \( \{A_k\} \) and \( \{B_k\} \) as follows:

\[
A_k \geq 0, B_k \geq 0, B_0 \neq 0, B_{k+1}^2 = \frac{B_k^2}{\beta_k}, A_{k+1}^2 = \beta \gamma_k^2 B_{k+1}^2 \\
(34)
\]

Without loss of generality, we assume \( A_0 = 0 \) to be consistent with the definition \( \gamma_{-1} = 0 \). Also note that since \( \beta_k \in (0, 1) \), we have \( B_{k+1} \geq B_k \). Now using the definition of the sequence \( \{\gamma_k\}, \{\alpha_k\} \), we have,

\[
A_{k+1}^2 = \frac{\beta \gamma_k^2 B_k^2}{\beta_k} = \frac{\gamma_k^2}{\beta_k \gamma_{k-1}^2} A_k^2 = \frac{\gamma_k^2}{\gamma_{k-1}^2} - \frac{\zeta}{m} \gamma_k \\
(35)
\]

Equation (35) also implies that the sequence \( \{A_k\} \) is an increasing sequence. Now, it is straightforward to check that the following identities hold:

\[
\beta B_{k+1}^2 \gamma_k^2 = A_{k+1}^2, B_{k+1}^2 \beta_k = B_k^2 \quad \text{and} \quad B_{k+1}^2 \beta_k \gamma_{k-1}^2 = A_k^2 \\
(36)
\]

Now, multiplying both sides of (33) by \( B_{k+1}^2 \) and using the above identities we have,

\[
B_{k+1}^2 \mathbb{E}_{i(k)|I(k-1)}(r_{k+1}^2) + A_{k+1}^2 \mathbb{E}_{i(k)|I(k-1)}(\| x_{k+1} - x^* \|^2) \\
\leq B_k^2 r_k^2 + A_k^2 \| x_k - x^* \|^2 \\
(37)
\]
Furthermore, we have,

\[ E_I(k) \left[ B_{k+1}^2 r_{k+1}^2 + A_{k+1}^2 \|x_{k+1} - x^*\|^2 \right] = E_I(k) \left[ B_{k+1}^2 \sum_{i=1}^k E(i) \left( \|x_i - x^*\|^2 \right) \right] \]

\[ \leq E_I(k-1) \left[ B_k^2 r_k^2 + A_k^2 \|x_k - x^*\|^2 \right] \]

\[ \vdots \]

\[ \leq E_I(0) \left[ B_1^2 r_1^2 + A_1^2 \|x_1 - x^*\|^2 \right] \]

\[ \leq B_0^2 \|x_0 - x^*\|^2 \]

\[ = B_0^2 r_0^2 \quad (38) \]

Therefore, using (38) we can conclude the following bound,

\[ E \left[ r_{k+1}^2 \right] \leq \frac{B_0^2}{B_{k+1}} r_0^2 \quad \text{and} \quad E \left[ \|x_{k+1} - x^*\|^2 \right] \leq \frac{B_0^2}{A_{k+1}} r_0^2 \quad (39) \]

Now, we need to estimate the growth of the defined sequences \( \{A_k\} \) and \( \{B_k\} \). Here, we follow the proof for the Accelerated Coordinate Descent method of Nesterov [35] and accelerated randomized Kaczmarz algorithm by Wright et al. [20] as they are more general in the context of acceleration. We have,

\[ B_k^2 = \beta_k B_{k+1}^2 = \left( 1 - \frac{\lambda \beta}{m \gamma_k} \right) B_{k+1}^2 = \left( 1 - \frac{\lambda \beta}{m \sqrt{\beta} B_{k+1}} \right) B_{k+1}^2 \quad (40) \]

Simplifying (40) we get,

\[ \frac{\lambda \sqrt{\beta}}{m} A_{k+1} B_{k+1} = B_{k+1}^2 - B_k^2 = (B_{k+1} - B_k)(B_{k+1} + B_k) \]

\[ \leq 2B_{k+1}(B_{k+1} - B_k) \]

Here, we used the identity \( B_{k+1} \geq B_k \), which simplifies to:

\[ B_{k+1} \geq B_k + \frac{\lambda \sqrt{\beta}}{2m} A_{k+1} \geq B_k + \frac{\lambda \sqrt{\beta}}{2m} A_k \quad (41) \]

Similarly, note that,

\[ \frac{A_{k+1}^2}{B_{k+1}^2} - \frac{\zeta \sqrt{\beta}}{m} A_{k+1} = \frac{\beta \gamma_k^2}{m} \]

\[ = \beta \left( 1 - \frac{\lambda \beta}{m \gamma_k} \right) \gamma_{k-1}^2 \]

\[ = \beta \beta_k \gamma_{k-1}^2 = \beta \frac{A_k^2}{B_k} \frac{A_{k+1}^2}{B_{k+1}} \]

Above equation simplifies to the following:

\[ \frac{\zeta \sqrt{\beta}}{m} A_{k+1} B_{k+1} = A_{k+1}^2 - A_k^2 = (A_{k+1} - A_k)(A_{k+1} + A_k) \]

\[ \leq 2A_{k+1}(A_{k+1} - A_k) \]

\( \square \) Springer
In this case, we used the identity $A_{k+1} \geq A_k + \frac{\zeta \sqrt{\beta}}{2m} B_{k+1} \geq A_k + \frac{\zeta \sqrt{\beta}}{2m} B_k$ (42)

By combining the two expressions of (41) and (42) in a LS we get,

$$\begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} \geq \begin{bmatrix} 1 \\ \frac{\lambda \sqrt{\beta}}{2m} \\ \frac{\zeta \sqrt{\beta}}{2m} \end{bmatrix} \begin{bmatrix} A_k \\ B_k \end{bmatrix} \geq \cdots \geq \begin{bmatrix} 1 \\ \frac{\lambda \sqrt{\beta}}{2m} \\ \frac{\zeta \sqrt{\beta}}{2m} \end{bmatrix}^{k+1} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}$$ (43)

The Jordan decomposition of the matrix in the above expression is given by,

$$\begin{bmatrix} 1 \\ \frac{\lambda \sqrt{\beta}}{2m} \\ \frac{\zeta \sqrt{\beta}}{2m} \end{bmatrix}^{k+1} = \begin{bmatrix} -\sqrt{\frac{\zeta}{\lambda}} \sqrt{\frac{\kappa}{\lambda}} \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \sigma_2^{k+1} \\ 0 \\ \sigma_1^{k+1} \end{bmatrix} \begin{bmatrix} -\frac{1}{2} \sqrt{\frac{\zeta}{\lambda}} \\ \frac{1}{2} \sqrt{\frac{\zeta}{\lambda}} \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\kappa}{2m} B_0 \end{bmatrix}$$ (44)

Here, $\sigma_1 = 1 + \sqrt{\frac{\beta \beta}{2m}}$ and $\sigma_2 = 1 - \sqrt{\frac{\beta \beta}{2m}}$. Using $A_0 = 0$ and the decomposition of (44), from Eq. (43) we have,

$$\begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} \geq \begin{bmatrix} 1 \\ \frac{\lambda \sqrt{\beta}}{2m} \\ \frac{\zeta \sqrt{\beta}}{2m} \end{bmatrix}^{k+1} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}$$

$$= \begin{bmatrix} -\sqrt{\frac{\zeta}{\lambda}} \sqrt{\frac{\kappa}{\lambda}} \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} \sigma_2^{k+1} \\ 0 \\ \sigma_1^{k+1} \end{bmatrix} \begin{bmatrix} -\frac{1}{2} \sqrt{\frac{\zeta}{\lambda}} \\ \frac{1}{2} \sqrt{\frac{\zeta}{\lambda}} \end{bmatrix} \begin{bmatrix} 1 \\ \frac{\kappa}{2m} B_0 \end{bmatrix}$$

The above gives us the following growth bound for the sequences $\{A_k\}$ and $\{B_k\}$ as follows:

$$A_{k+1} \geq \frac{1}{2} \sqrt{\frac{\zeta}{\lambda}} \left( \sigma_1^{k+1} - \sigma_2^{k+1} \right) B_0$$ (45)

$$B_{k+1} \geq \frac{1}{2} \left( \sigma_1^{k+1} + \sigma_2^{k+1} \right) B_0$$ (46)

Substituting these above bounds of (45) and (46) in Eq. (39), we get the following bounds:

$$\mathbb{E} \left( r_{k+1}^2 \right) = \mathbb{E} \left[ \|v_k - x^*\|^2_{(A^T A)} \right] \leq \frac{B_0^2}{B_{k+1}^2} \leq \frac{4\|x_0 - x^*\|^2_{(A^T A)}}{\left( \sigma_1^{k+1} + \sigma_2^{k+1} \right)^2}$$ (47)

$$\mathbb{E} \left[ \|x_{k+1} - x^*\|^2 \right] \leq \frac{R_0^2}{A_{k+1}} \leq \frac{4\lambda \|x_0 - x^*\|^2_{(A^T A)}}{\zeta \left( \sigma_1^{k+1} - \sigma_2^{k+1} \right)^2}$$ (48)

The above equations complete the proof of Theorem 1.

4 Numerical experiments

We implemented the ASKM algorithm in MATLAB and performed the numerical experiments in a Dell Precision 7510 workstation with 32GB RAM, Intel Core i7-6820HQ CPU,
4.1 Comparison of SKM and ASKM on random data

We considered systems $Ax \leq b$ where the entries of $A$ and $b$ are chosen randomly from the corresponding distribution. To make sure that $b \in \mathcal{R}(A)$, we generated two vectors $x_1, x_2 \in \mathbb{R}^n$ at random from the corresponding distributions, then multiplied them by $A$ and set $b$ as a convex combination of those two vectors. We considered two types of random data sets: highly correlated systems and Gaussian systems. In the highly correlated systems, entries of $A$ are chosen uniformly at random between $[0.9, 1.0]$ and $b$ is chosen accordingly such that the system $Ax \leq b$ has a feasible solution. The entries of $A$ in the Gaussian systems are chosen from standard normal distribution and $b$ is chosen accordingly as before.

In Fig. 4.1, we provide a comparison between SKM and ASKM for three randomly generated correlated systems. We compare the average computational time necessary for SKM and ASKM with several choices of sample size $\beta$ to reach positive residual error $10^{-05}$ (i.e., $\| (Ax_k - b)^+ \|_2 \leq 10^{-05}$). We compare the two algorithms for the choice of $\delta = 1$. For the three test cases, we see that for any $1 \leq \beta \leq m$, ASKM significantly outperform SKM in terms of average computation time.

In Fig. 4.2, we show the same comparison experiments for randomly generated Gaussian systems. Similar to the correlated systems, ASKM algorithm solves the Gaussian systems much faster than the SKM algorithm. Notice that in Fig. 4.2b the computational time of SKM algorithm stays at 1000s for sample size $\beta \geq 2000$. This happens due to an additional terminating condition of maximum run time set at 1000s. While SKM algorithm fails to converge within the limiting time for larger sample sizes ($\beta \geq 2000$), ASKM algorithm finds a feasible solution for any sample size. Moreover, if we analyze the trend of ASKM’s average computational time in both figures (Figs. 4.1, 4.2), we see that ASKM accelerates the

\[\delta\] here is same as $\lambda$ in De Loera et al. [9].
Fig. 2 Average computation time of SKM and ASKM on Gaussian random systems to either reach residual error $\| (Ax_k - b)^+ \|_2 \leq 10^{-05}$ or a maximum run time limit of 1000 s

SKM algorithm and the nature of acceleration is quadratic which validates our claim of Theorem 1. In Figs. 3 and 4, we compare the positive residual error for SKM and ASKM for different sample sizes ($\beta = 1, 100, 1000$). We plot iteration versus residual error and time versus residual error for random Gaussian systems. Based on the findings of Figs. 3 and 4, we can conclude that irrespective of sample size selection, $\| (Ax_k - b)^+ \|_2$ converges to zero much more faster for ASKM than for SKM. The convergence of $\| (Ax_k - b)^+ \|_2$ for both ASKM and SKM are much slower for the choice of $\beta = 1$ as expected.

For $\beta = 100$ and $\beta = 1000$, the convergence rate of ASKM takes over SKM after a small amount of time. In addition, the convergence rate remains similar for both the test case problems (5000 $\times$ 1000 and 8000 $\times$ 2000). To investigate the solution quality of both SKM and ASKM, we measure the number of satisfied constraints after each iteration and the corresponding computational time for both algorithms. We summarize our findings in Figs. 5 and 6 for the above test sets. For simplification, we used the Fraction of satisfied constraints (FSC) as a measure of quality of the solution generated by both SKM and ASKM algorithms. After analyzing Figs. 5 and 6, we can conclude that the choice of $\beta = 1$ is the worst choice.
Fig. 4 Comparison of SKM and ASKM on a $8000 \times 2000$ random Gaussian system for sample size $\beta = 1, 100, 1000$

(a) Iterations versus residual error

(b) Time versus residual error

Fig. 5 Comparison of SKM and ASKM in terms of fraction of satisfied constraints on a $5000 \times 1000$ random Gaussian system

(a) Iterations versus FSC

(b) Time versus FSC

Fig. 6 Comparison of SKM and ASKM in terms of fraction of satisfied constraints on a $8000 \times 2000$ random Gaussian system

(a) Iterations versus FSC

(b) Time versus FSC
Fig. 7 Comparison of SKM and ASKM for real life data sets

as both SKM and ASKM takes much more data time to satisfy all the constraints. However, for the choice of $\beta = 100$ and $\beta = 1000$, ASKM takes much less time compared to SKM to find a solution within the error margin. For example, in Fig. 5, the choice of $\beta = 1000$ ASKM takes approximately 37 s to satisfy all the 5000 constraints whereas SKM takes up to 75 s.

4.2 Comparison of SKM and ASKM for real-world non-random data

In this subsection, we consider two real-world non-random problems. We consider Support Vector Machine (SVM) instances with linear classification and feasibility problems arising in benchmark libraries. We considered the standard test cases given in [9,37,38].

We compare SKM and ASKM methods to solve the linear classification problem with SVM for (1) Wisconsin (diagnostic) breast cancer data set and (2) Credit card data set. The breast cancer data set includes data points whose features are computed from digitized images. Each data point is classified either as malignant or as benign. Our goal is to find a solution of the homogeneous system of inequalities, $Ax \leq 0$ which represents the separating hyperplane between malignant and benign data points. The system of inequalities has 569 constraints (data points) and 30 variables (features). Since the data set is not separable, we set SKM and ASKM to find the minimized residual norm $\|Ax_k\|_2$. For our setup, We consider the threshold $\|Ax_k\|_2 \leq 0.0005$ and $10^{-6}$.

The credit card data set described in [9,39] consists of features describing the payment profile of user and binary variable for on-time or default payment in a certain billing cycle. Similar to the breast cancer data set, this problem can be solved by finding a solution to the corresponding homogeneous system of inequalities, $Ax \leq 0$ which represents the separating hyperplane between given on-time and default data points. The resulting system of inequalities has 30,000 constraints (30,000 user profiles) and 23 variables (22 profile features). Since the data set is not separable, we set SKM and ASKM to find the minimized residual norm $\|Ax_k\|_2$. For our setup, we considered the threshold as $\|Ax_k\|_2 / \|Ax_0\|_2 \leq 0.1$ and 0.001.

Based on the comparison graphs shown in Fig. 7, we can conclude ASKM performs much better than SKM for the breast cancer data set (Fig. 7a). For the credit card data set ASKM performs marginally better than SKM for smaller error. Also note that, the computation time curve for credit card data is not as smooth as previous curves, which we can attribute to the irregularity of the coefficients. Such irregularity in the coefficients creates a dependence bias between residual error and actual constraints.
Table 1  CPU time comparisons among MATLAB methods solving LP, and SKM and ASKM solving LF

| Instance     | Dimensions | ASKM | SKM | Interior Point | Active set | $\epsilon$ | $\beta$ |
|--------------|------------|------|-----|----------------|------------|-------------|--------|
| lp_brandy    | 1047 × 303 | 0.007| 0.0117| 16.97          | 63.11      | 0.1         | 50     |
| lp_blend     | 337 × 114  | 0.41 | 0.56 | 2.28           | 4.62       | 0.001       | 20     |
| lp_agg       | 2207 × 615 | 0.059| 0.088| 66.54*         | 315.91*    | 0.01        | 50     |
| lp_adlittle  | 389 × 138  | 0.0008| 0.002| 2.16           | 4.96       | 0.01        | 10     |
| lp_bandm     | 1555 × 472 | 0.28 | 0.24 | 14.57          | 529.43*    | 0.01        | 70     |
| lp_degen2    | 2403 × 757 | 8.29 | 10.16| 7.13           | 21,038     | 0.01        | 200    |
| lp_finnis    | 3123 × 1064| 0.13 | 0.15 | 66.16*         | 237,750*   | 0.005       | 100    |
| lp_recipe    | 591 × 204  | 0.19 | 0.27 | 0.89           | 63.24      | 0.002       | 30     |
| lp_scorpion  | 1709 × 466 | 6.83 | 11.86| 17.68          | 8.02       | 0.005       | 200    |
| lp_stocfor1  | 565 × 165  | 0.31 | 0.37 | 2.13           | 2.52       | 0.001       | 50     |

*Indicates that the solver did not solve the problem to the desired accuracy due to reaching an predetermined upper limit on function evaluations of 100,000

4.3 Comparison among SKM, ASKM and existing methods for Netlib LP

In this subsection, we investigate the comparative performance of the proposed ASKM algorithm with SKM and benchmark algorithms such as IPM and ASM on several Netlib LPs. For the implementation of SKM and ASKM to the Netlib LPs, we follow the framework given by De Loera et al. [9]. Each of these problems was formulated as a standard LP problem (min $c^T x$ subject to $Ax = b$, $l \leq x \leq u$ with optimum value $p^*$). In their work, De Loera et al. [9] transformed them into an equivalent LF problem $\bar{A}x \leq \bar{b}$, where $\bar{A} = [A - AI - I c^T]^T$ and $\bar{b} = [b - b u - l p^*]^T$. We used this setup for all the experiments on Netlib LPs.

In Table 1, we provide the performance behaviour (computation time in seconds) of ASKM, SKM, IPM and ASM on the Netlib LPs. For fair comparison, we coded ASKM, SKM in MATLAB and compared with the MATLAB Optimization Toolbox function `fmincon`. Note that `fmincon` allows us to select both IPM and ASM methods.

At first, we solve the feasibility problem ($\bar{A}x \leq \bar{b}$) using SKM and ASKM and recorded the CPU time in Table 1. But we didn’t solve the feasibility problem (min 0 s.t. $\bar{A}x \leq \bar{b}$) directly using `fmincon`'s IPM and ASM algorithms since both of these methods fail to solve the feasibility problem due to the fact that in IPM, the Karush Kuhn Tucker (KKT) condition system in each iteration becomes singular and similarly ASM halts in the initial step of finding a feasible point.

For fairness of comparison, in Table 1, we list the CPU time as follows: for SKM and ASKM method we used the feasibility problem ($\bar{A}x \leq \bar{b}$) and for the `fmincon` algorithms we used the original optimization LPs (min $c^T x$ s.t. $Ax = b$, $l \leq x \leq u$). As noted in [9], this is not an obvious comparison. For a better comparison, following [9] we set the halting criterion for SKM and ASKM as $\max(\bar{A}x - \bar{b}) \leq \epsilon$ and the halting criterion for the `fmincon`’s algorithms are set as $\max(Ax_0 - b) \leq \epsilon$ and $\frac{c^T x}{c^T x_0} \leq \epsilon$, where $\epsilon$ is listed in Table 1. For each problem, every method started with the same initial solution far from the feasible region.

The experiments show that the proposed ASKM method compares favorably with IPM and ASM methods. Notice that the improvement of ASKM over SKM method for some problems are marginal as the analyzed instances contain sparse matrices while our proposed
ASKM is explicitly designed for dense problems. Following the method provided by Liu and Wright [20], we believe one can develop a sparse version of ASKM algorithm. The trick is to aggregate several steps to reduce the calculation by using the sparsity of the instances. For example, after calculating $x_k$, $y_k$ and $v_k$, instead of updating to $x_{k+1}$, $y_{k+1}$ and $v_{k+1}$, for $T \gg 1$ we can update $x_{k+T}$, $y_{k+T}$ and $v_{k+T}$ using the recurrence relation which will reduce the computational effort significantly.

5 Conclusion

In this work, we have proposed an accelerated version of SKM algorithm for solving LF problem using the celebrated Nesterov acceleration of Gradient Descent method. The proposed algorithm also generalizes the accelerated randomized Kaczmarz algorithm for solving LS problem in the context of sample size $\beta$. We have performed a series of numerical experiments to show the performance and effectiveness of our proposed algorithm in comparison with IPM and ASM methods. ASKM algorithm performs favourably in comparison with SKM, IPM and ASM method for a wide range of test instances. The proposed algorithm as it is, including the convergence analysis, can be adopted effectively for both dense and sparse systems, however, we believe, a more efficient algorithm is possible for the sparse case. In the future, we plan to extend this work to solve large-scale real-world problems with greater sparsity on the constraint matrix. Furthermore, due to the introduction of the acceleration to the SKM algorithm, we have a set of parameters (i.e., $\beta$, $d$ etc.) which we plan to optimize based on the problem structure to further improve the efficiency of the proposed algorithm.

Acknowledgements

The authors are truly grateful to the anonymous referees and the editor for their valuable comments and suggestions in the revision process which have greatly improved the quality of this paper.

References

1. Strohmer, T., Vershynin, R.: A randomized Kaczmarz algorithm with exponential convergence. J. Fourier Anal. Appl. 15(2), 262 (2008)
2. Leventhal, D., Lewis, A.S.: Randomized methods for linear constraints: convergence rates and conditioning. Math. Oper. Res. 35(3), 641–654 (2010)
3. Needell, D.: Randomized Kaczmarz solver for noisy linear systems. BIT Numer. Math. 50(2), 395–403 (2010)
4. Drineas, P., Mahoney, M.W., Muthukrishnan, S., Sarlós, T.: Faster least squares approximation. Numer. Math. 117(2), 219–249 (2011)
5. Zouzias, A., Freris, N.M.: Randomized extended Kaczmarz for solving least squares. SIAM J. Matrix Anal. Appl. 34(2), 773–793 (2013)
6. Lee, Y.T., Sidford, A.: Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In: Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS’13, Washington, DC, USA. IEEE Computer Society, pp. 147–156 (2013)
7. Ma, A., Needell, D., Ramdas, A.: Convergence properties of the randomized extended gauss seidel and Kaczmarz methods. SIAM J. Matrix Anal. Appl. 36(4), 1590–1604 (2015)
8. Qu, Z., Richtarik, P., Takac, M., Fercoq, O.: SDNA: stochastic dual Newton ascent for empirical risk minimization. In: Proceedings of the 33rd International Conference on Machine Learning, vol. 48, New York, USA, 20–22 June 2016. PMLR, pp. 1823–1832 (2016)
9. De Loera, J., Haddock, J., Needell, D.: A sampling Kaczmarz–Motzkin algorithm for linear feasibility. SIAM J. Sci. Comput. 39(5), S66–S87 (2017)
10. Dembo, R.S., Eisenstat, S.C., Steihaug, T.: Inexact Newton methods. SIAM J. Numer. Anal. 19(2), 400–408 (1982)
11. Eisenstat, S.C., Walker, H.F.: Choosing the forcing terms in an inexact Newton method. SIAM J. Sci. Comput. 17(1), 16–32 (1996)
12. Bellavia, S.: Inexact interior-point method. J. Optim. Theory Appl. 96(1), 109–121 (1998)
13. Zhao, X.-Y., Sun, D., Toh, K.-C.: A Newton-CG augmented lagrangian method for semidefinite programming. SIAM J. Optim. 20(4), 1737–1765 (2010)
14. Wang, C., Aimin, X.: An inexact accelerated proximal gradient method and a dual Newton-CG method for the maximal entropy problem. J. Optim. Theory Appl. 157(2), 436–450 (2013)
15. Jiang, K., Sun, D., Toh, K.-C.: An inexact accelerated proximal gradient method for large scale linearly constrained convex SDP. SIAM J. Optim. 22(3), 1042–1064 (2012)
16. Gondzio, J.: Convergence analysis of an inexact feasible interior point method for convex quadratic programming. SIAM J. Optim. 23(3), 1510–1527 (2013)
17. Kaczmarz, S.: Angenaherte auflösung von systemen linearer gleichungen. Bull. Int. Acad. Pol. Sci. Lett. 35, 355–357 (1937)
18. Herman, G.T.: Fundamentals of Computerized Tomography: Image Reconstruction from Projections, 2nd edn. Springer, New York (2009)
19. Censor, Y.: Row-action methods for huge and sparse systems and their applications. SIAM Rev. 23(4), 444–466 (1981)
20. Liu, J., Wright, S.J.: An accelerated randomized Kaczmarz algorithm. Math. Comput. 85(297), 153–178 (2016)
21. Agamon, S.: The relaxation method for linear inequalities. Can. J. Math. 6, 382–392 (1954)
22. Motzkin, T.S., Schoenberg, I.J.: The relaxation method for linear inequalities. Can. J. Math. 6, 393–404 (1954)
23. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychol. Rev. 65, 65–386 (1958)
24. Ramdas, A., Peña, J.: Margins, kernels and non-linear smoothed perceptrons. In: Proceedings of the 31st International Conference on Machine Learning, vol. 32, Beijing, China, 22–24 June. PMLR, pp. 244–252 (2014)
25. Ramdas, A., Peña, J.: Towards a deeper geometric, analytic and algorithmic understanding of margins. Optim. Methods Softw. 31(2), 377–391 (2016)
26. Nutini, J., Sepehry, B., Laradji, I., Schmidt, M., Koepke, H., Virani, A.: Convergence rates for greedy Kaczmarz algorithms, and faster randomized Kaczmarz rules using the orthogonality graph. In: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI’16, Arlington, Virginia, USA, 2016. AUAI Press, pp. 547–556 (2016)
27. Petra, S., Popa, C.: Single projection Kaczmarz extended algorithms. Numer. Algorithms 73(3), 791–806 (2016)
28. Hoffman, A.J.: On approximate solutions of systems of linear inequalities. In: Selected Papers of Alan J Hoffman: With Commentary. World Scientific, pp. 174–176 (2003)
29. Xu Xiang, X., Liu, W.T., Dai, X.: An accelerated randomized extended Kaczmarz algorithm. J. Phys.: Conf. Ser. 814(1), 012017 (2017)
30. Morsched, M.S., Noor-E-Alam, M.: Generalized affine scaling algorithms for linear programming problems. Comput. Oper. Res. 114, 104807 (2020)
31. Nesterov, Y.: A method for solving the convex programming problem with convergence rate $o(1/k^2)$. Soviet Mathematics Doklady, vol. 27, pp. 372–376 (1983)
32. Nesterov, Y.: Gradient methods for minimizing composite functions. Math. Program. 140(1), 125–161 (2013)
33. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course, 1st edn. Springer, New York (2014)
34. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. 103(1), 127–152 (2005)
35. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. 22(2), 341–362 (2012)
36. Gordon, R., Bender, R., Herman, G.T.: Algebraic reconstruction techniques (art) for three-dimensional electron microscopy and x-ray photography. J. Theor. Biol. 29(3), 471–481 (1970)
37. Calafiore, G., El Ghoui, L.: Optimization Models. Control Systems and Optimization Series. Cambridge University Press, Cambridge (2014)
38. Lichman, M.: UCI machine learning repository. (2013). http://archive.ics.uci.edu/ml
39. Yeh, I.-C., Lien, C.: The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Syst. Appl. 36(2), 2473–2480 (2009)