On Momentum Acceleration for Randomized Coordinate Descent in Matrix Completion

Matthew Callahan[1], Trung Vu[2], and Raviv Raich[1]
[1]School of EECS, Oregon State University, Corvallis, OR 97331-5501, USA
[2] Department of CSEE, University of Maryland, Baltimore County, MD 21250-0002, USA
{callamat, raich}@oregonstate.edu trungvv@umbc.edu

***Abstract*—Matrix completion plays an important role in machine learning and signal processing, with applications ranging from recommender systems to image inpainting. Many approaches have been considered to solve the problem and some offer computationally efficient solutions. In particular, a highly-efficient random coordinate descent approach reduces the per-epoch computation dramatically. This paper is concerned with further improvement of computational efficiency to expand the range of problem sizes and conditions that can be solved. Momentum acceleration is a well-known method to improve the efficiency of iterative algorithms, but applying it to random coordinate descent methods without increasing the computational complexity is non-trivial. To address this challenge, we introduce a momentum-accelerated randomized coordinate descent for matrix completion approach that does not increase computational complexity by accelerating at the level of epochs. Additionally, we propose an analysis-driven, tuning-free method for step size selection. To that end, we offer a convergence rate analysis for the algorithm. Using numerical evaluations, we demonstrate the competitiveness of the method and verify the theoretical analysis.**

***Index Terms*—coordinate descent, asymptotic convergence analysis, matrix completion, randomized methods.**

I. Introduction

Matrix completion addresses the recovery of a rank-r matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ from a subset of s entries, $\Omega = \{(i,j) \in [m] \times [n] \mid M_{ij} \text{ is observed}\}$. One application is recommendation systems in which the objective is to estimate the full set of ratings for every user for every product, $\boldsymbol{M}$, using only a small subset of user-product ratings, Ω [1]. Numerous additional applications in signal processing include image inpainting [2] and device localization [3]. Using a rank-r factorization, the matrix completion problem can be reformulated as finding an $m \times r$ matrix $\boldsymbol{A}$ and an $n \times r$ matrix $\boldsymbol{B}$ such that $(\boldsymbol{A}\boldsymbol{B}^\top)_{ij} = M_{ij}$ for all $(i,j) \in \Omega$ where $|\Omega| = s$. The following optimization is used to formalize matrix completion:

$$\min_{\boldsymbol{A}\in\mathbb{R}^{m\times r},\boldsymbol{B}\in\mathbb{R}^{n\times r}} \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{A}\boldsymbol{B}^\top - \boldsymbol{M})\|_F^2, \tag{1}$$

where the projection $\mathcal{P}_\Omega : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is defined as

$$[\mathcal{P}_\Omega(\boldsymbol{X})]_{ij} = \begin{cases} X_{ij} & (i,j) \in \Omega, \\ 0 & (i,j) \notin \Omega. \end{cases} \tag{2}$$

There exists a long line of work focused on first-order iterative methods for low-rank matrix completion, including gradient descent and its variations [4]–[6], iterative hard-thresholding [7], [8], and alternating minimization [9], [10]. Recently, we proposed a simple yet competitive approach for solving matrix completion via random coordinate descent (RCD) [11]. A theoretical rate of convergence analysis was performed and confirmed via empirical evaluation. Moreover, the approach benefits from a low per-epoch computational complexity of $O(sr)$. However, to extend the applicability of the approach, we are interested in finding methods that can improve the rate of convergence without increasing the per-epoch computational complexity.

The acceleration of RCD methods has been studied in [12]–[14]. These methods apply acceleration with the update of each coordinate. This is efficient under the assumption that calculating one entry of the gradient is on the order of the number of coordinates. However, for matrix completion via factorization, this calculation is of a lower order than the number of unknown entries, since only one column of $\boldsymbol{A}$ or of $\boldsymbol{B}$ is needed. Hence, a momentum acceleration step is of a higher computational complexity order than a single coordinate update, and care needs to be taken as to how often this step is performed.

Contributions To address the aforementioned challenge, we introduce a per-epoch momentum acceleration of factor-based randomized coordinate descent for matrix completion. Additionally, we provide a rate of convergence analysis on the mean error vector, which we confirm using numerical experiments. Using this analysis we introduce a method for determining the momentum coefficient a priori, rendering our method tuning-free. Moreover, we demonstrate empirically that the proposed method outperforms state-of-the-art methods, achieving a superior convergence rate while maintaining the same per-epoch computational efficiency as without the acceleration.

Notation. Throughout the paper, we use the notations $\|\cdot\|_F$ and $\|\cdot\|$ to denote the Frobenius norm of a matrix and the Euclidean norm of a vector, respectively. Boldfaced symbols are reserved for vectors and matrices, while the elements of a vector/matrix are unbold. In addition, $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix. The notation $\boldsymbol{e}_i^{(n)}$ denotes the ith vector in the natural basis of $\mathbb{R}^n$. We use $\otimes$ to denote the Kronecker product between two matrices and $\mathrm{vec}(\boldsymbol{X})$ to denote the vectorization of $\boldsymbol{X}$ by stacking its columns on top of one another. We use $\boldsymbol{A}^{(k)}$ to denote iterations of coordinate updates, and $\boldsymbol{A}_k$ to denote iterations where acceleration may occur.

II. Background

To set the ground for understanding the momentum-based acceleration of RCD for matrix completion, we begin with a brief overview of momentum acceleration for fixed-point iterations and of the RCD algorithm and its convergence rate analysis.

A. Momentum-based acceleration for fixed-point iterations

Consider a real-valued vector function $\boldsymbol{f}$ of a vector variable and given a point $\boldsymbol{x}_0$ in the domain of $\boldsymbol{f}$, the fixed-point iteration is given by

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k), \quad \text{for} \quad k = 0, 1, 2, \ldots,$$

which yields a sequence of vectors $\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$. If $\boldsymbol{f}$ is a contraction map, this sequence converges to a fixed point $\boldsymbol{x}^*$ such that $\boldsymbol{f}(\boldsymbol{x}^*) = \boldsymbol{x}^*$. If $\boldsymbol{f}$ is twice differentiable, then one can derive the update equation for the error $\boldsymbol{\epsilon}_k = \boldsymbol{x}_k - \boldsymbol{x}^*$ as a recursion

$$\boldsymbol{\epsilon}_{k+1} = \boldsymbol{T}\boldsymbol{\epsilon}_k + \boldsymbol{q}(\boldsymbol{\epsilon}_k), \tag{3}$$

where $\boldsymbol{T} = \nabla \boldsymbol{f}(\boldsymbol{x}^*)$ and $\boldsymbol{q}$ is the residual linear approximation error satisfying $\|\boldsymbol{q}(\boldsymbol{\epsilon})\| \leq q\|\boldsymbol{\epsilon}\|^2$ for any $\boldsymbol{\epsilon}$ [15].

In [16], Vu and Raich showed that if $\boldsymbol{T}$ is diagonalizable, then for arbitrarily small $\bar{\epsilon} > 0$ it holds that $\|\boldsymbol{\epsilon}_k\| < \bar{\epsilon},\ \forall k \geq N$, where N is the smallest integer satisfying

$$N \geq \frac{\log(1/\bar{\epsilon})}{\log(1/\rho(\boldsymbol{T}))} + c\left(\rho(\boldsymbol{T}), \frac{q\|\boldsymbol{\epsilon}_0\|}{1-\rho(\boldsymbol{T})}\right), \tag{4}$$

where $0 \leq \rho(\boldsymbol{T}) < 1$ is the spectral radius of matrix $\boldsymbol{T}$, $q = \sup_{\|\boldsymbol{\delta}\| \leq \|\boldsymbol{\epsilon}_0\|} \frac{\|\boldsymbol{q}(\boldsymbol{\delta})\|}{\|\boldsymbol{\delta}\|^2}$, and $c(\cdot)$ is a function independent of $\bar{\epsilon}$. This suggests that the error norm decreases exponentially at rate $\rho = \rho(\boldsymbol{T})$ as $\|\boldsymbol{\epsilon}_k\| \leq C\rho^k$ (linear convergence).

To accelerate the convergence of $\boldsymbol{x}_k$ towards $\boldsymbol{x}_*$, a momentum-accelerated version of the fixed-point iteration can be considered

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k) + \beta(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}).$$

The resulting error update equation is then

$$\boldsymbol{\epsilon}_{k+1} = \boldsymbol{T}\boldsymbol{\epsilon}_k + \beta(\boldsymbol{\epsilon}_k - \boldsymbol{\epsilon}_{k-1}) + \boldsymbol{q}(\boldsymbol{\epsilon}_k). \tag{5}$$

By stacking $\boldsymbol{\epsilon}_{k+1}$ on top of $\boldsymbol{\epsilon}_k$, this equation can be rewritten as

$$\begin{bmatrix}\boldsymbol{\epsilon}_{k+1}\\ \boldsymbol{\epsilon}_k\end{bmatrix} = \boldsymbol{H}(\beta)\begin{bmatrix}\boldsymbol{\epsilon}_k\\ \boldsymbol{\epsilon}_{k-1}\end{bmatrix} + \begin{bmatrix}\boldsymbol{q}(\boldsymbol{\epsilon}_k)\\ \boldsymbol{q}(\boldsymbol{\epsilon}_{k-1})\end{bmatrix},\ \boldsymbol{H}(\beta) = \begin{bmatrix}\boldsymbol{T}+\beta\boldsymbol{I} & -\beta\boldsymbol{I}\\ \boldsymbol{I} & \boldsymbol{0}\end{bmatrix}.$$

Then, from (4), the number of iterations N required to achieve an error of $\bar{\epsilon}$ is the smallest integer N such that

$$N \geq \frac{\log(1/\bar{\epsilon})}{\log(1/\rho(\boldsymbol{H}(\beta)))} + c\left(\rho(\boldsymbol{H}(\beta)), \frac{q\|\boldsymbol{\epsilon}_0\|}{1-\rho(\boldsymbol{H}(\beta))}\right),$$

for which the rate is $\rho = \rho(\boldsymbol{H}(\beta))$. If $\boldsymbol{T}$ has all real eigenvalues between 0 and 1, then by setting β to the optimal value $\beta^* = (1 - \sqrt{1-\rho(\boldsymbol{T})})^2$, the fastest rate of convergence (i.e., the smallest ρ) can be achieved and is $\rho(\boldsymbol{H}(\beta^*)) = \sqrt{\beta^*} = 1 - \sqrt{1-\rho(\boldsymbol{T})}$ [17].

The rate of slowly converging unaccelerated fixed-point algorithms can be expressed as $\rho(\boldsymbol{T}) = \rho(\boldsymbol{I} - \boldsymbol{Q}) = 1 - \delta$, where $\delta \ll 1$. The number of steps to achieve $\bar{\epsilon}$ error is proportional (up to an additive constant) to $1/\log(1/\rho) \approx 1/\delta$. On the other hand, the rate for the optimally-accelerated algorithm is $\rho(\boldsymbol{H}) = 1 - \sqrt{\delta}$ and consequently the number of steps would be proportional to $1/\sqrt{\delta}$. This suggests that momentum acceleration with an optimal step-size selection can reduce the number of iterations to a number proportional to the square root of the number of iterations in the unaccelerated case.

B. Randomized coordinate descent for matrix completion

1) Algorithm: In [11], a regularization-free randomized coordinate descent algorithm on the factors is presented. To handle factor ambiguity, a refactorization is included. The algorithm performs iteration k by selecting a coordinate uniformly from the set $\{A_{11}, \ldots, A_{mr}, B_{11}, \ldots, B_{nr}\}$ and minimizing the error (1) with respect to that coordinate while holding the remaining coordinates fixed. During an entry update, both the single entry of $\boldsymbol{A}^{(k)}$ or $\boldsymbol{B}^{(k)}$ and the corresponding row or column of the residual error matrix $\boldsymbol{R}^{(k)} = \mathcal{P}_\Omega(\boldsymbol{A}^{(k)}(\boldsymbol{B}^{(k)})^\top - \boldsymbol{M})$ are updated as in Algorithm 1.

The rate of convergence of the error between $\boldsymbol{X}^{(k)} = \boldsymbol{A}^{(k)}\boldsymbol{B}^{(k)\top}$ and $\boldsymbol{M}$ depends on the normalized columns of $\boldsymbol{A}$ and $\boldsymbol{B}$ at the solution. Hence, one way to maintain a predictable rate is to construct a unique factorization (up to permutation/sign). This is the approach taken by [11]. Given arbitrary factors $\boldsymbol{A}$ and $\boldsymbol{B}$, new factors $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ are produced such that $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}^\top = \boldsymbol{A}\boldsymbol{B}^\top$ and $\tilde{\boldsymbol{A}}^\top\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{B}}^\top\tilde{\boldsymbol{B}} = \boldsymbol{\Sigma}$ where $\boldsymbol{\Sigma}$ is the matrix of singular values of $\boldsymbol{X}$. This procedure follows Lines 1-7 of Algorithm 2. Assuming distinct values of $\boldsymbol{\Sigma}$, this procedure produces unique factors $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$ up to a consistent permutation and/or sign change of paired columns. This refactorization procedure is performed every $r(n+m)$ element updates, namely, *an epoch*. This allows the computational complexity of the entire algorithm to be $O(sr)$ per epoch. Recall that $s = |\Omega|$ is the number of known entries.

Algorithm 1: One epoch of Randomize Coordinate Descent (RCD) for Matrix Completion modified from [11]

Input: Initial value $\boldsymbol{A} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times r}$, $\mathcal{P}_\Omega(\boldsymbol{M})$, Ω, direction vector $\boldsymbol{s} \in \mathbb{R}^m$
Output: Normalized factors $\tilde{\boldsymbol{A}} \in \mathbb{R}^{m\times r}$ and $\tilde{\boldsymbol{B}} \in \mathbb{R}^{n\times r}$

$\boldsymbol{R} \leftarrow \mathcal{P}_\Omega(\boldsymbol{A}\boldsymbol{B}^\top - \boldsymbol{M})$ for all $(i,j) \in \Omega$
for $k = 1, 2, \ldots, (m+n)r$ **do**
  Sample $p \in \{1, \ldots, m+n\}$ uniformly at random
  if $p \in [1, m]$ **then** ▷ Perform the A-update
    Sample $(i,j) \in \{1, \ldots, m\} \times \{1, \ldots, r\}$ uniformly
    $\gamma = \frac{\sum_{l:(i,l)\in\Omega} R_{il}B_{lj}}{\sum_{l:(i,l)\in\Omega} B_{lj}^2}$ ▷ $O(s/m)$
    $A_{ij} \leftarrow A_{ij} - \gamma$ ▷ $O(1)$
    for $h : (i,h) \in \Omega$ **do** ▷ $O(s/m)$
      $R_{ih} \leftarrow R_{ih} - \gamma B_{hj}$
  else ▷ Perform the B-update
    Sample $(i,j) \in \{1, \ldots, n\} \times \{1, \ldots, r\}$ uniformly
    $\gamma = \frac{\sum_{l:(l,i)\in\Omega} R_{li}A_{lj}}{\sum_{l:(l,i)\in\Omega} A_{lj}^2}$ ▷ $O(s/n)$
    $B_{ij} \leftarrow B_{ij} - \gamma$ ▷ $O(1)$
    for $h : (h,i) \in \Omega$ **do** ▷ $O(s/n)$
      $R_{hi} \leftarrow R_{hi} - \gamma A_{hj}$
$(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}) = \text{Refactor}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{s})$ ▷ $O\big((m+n+r)r^2\big)$

2) Error Analysis: In [11], an error analysis is presented on $\boldsymbol{\delta}^{(k)} = (\boldsymbol{Z}^\top\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{Z})^{1/2}\boldsymbol{Z}^\top \operatorname{vec}(\boldsymbol{A}^{(k)}(\boldsymbol{B}^{(k)})^\top - \boldsymbol{M})$, a projected version of the error between $\boldsymbol{X}^{(k)} = \boldsymbol{A}^{(k)}\boldsymbol{B}^{(k)\top}$ and $\boldsymbol{M}$. Specifically, it is shown that the error follows

$$\boldsymbol{\delta}^{(k)} = \boldsymbol{T}^{(k)}\boldsymbol{\delta}^{(k-1)} + \mathbf{q}(\boldsymbol{\delta}^{(k-1)}),$$

where $\boldsymbol{q}(\boldsymbol{\delta}) \leq c\|\boldsymbol{\delta}\|^2$ for some $c > 0$, while $\boldsymbol{T}^{(k)}$ is a random matrix whose mean is given by $\boldsymbol{T} = \mathbb{E}[\boldsymbol{T}^{(k)}] = \boldsymbol{I} - \boldsymbol{Q}$. The analysis yields

$$\boldsymbol{Q} = \frac{1}{(m+n)r}\sum_{i=1}^{m}\sum_{j=1}^{r}\frac{\boldsymbol{G}^{1/2}\boldsymbol{Z}^\top(\boldsymbol{v}_j \otimes \boldsymbol{e}_i^{(m)})(\boldsymbol{v}_j \otimes \boldsymbol{e}_i^{(m)})^\top\boldsymbol{Z}\boldsymbol{G}^{1/2}}{\|\boldsymbol{G}^{1/2}\boldsymbol{Z}^\top(\boldsymbol{v}_j \otimes \boldsymbol{e}_i^{(m)})\|^2} + \frac{1}{(m+n)r}\sum_{i=1}^{n}\sum_{j=1}^{r}\frac{\boldsymbol{G}^{1/2}\boldsymbol{Z}^\top(\boldsymbol{e}_i^{(n)} \otimes \boldsymbol{u}_j)(\boldsymbol{e}_i^{(n)} \otimes \boldsymbol{u}_j)^\top\boldsymbol{Z}\boldsymbol{G}^{1/2}}{\|\boldsymbol{G}^{1/2}\boldsymbol{Z}^\top(\boldsymbol{e}_i^{(n)} \otimes \boldsymbol{u}_j)\|^2}, \tag{6}$$

with $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ the reduced singular value decomposition of $\boldsymbol{M}$, $\boldsymbol{Z} \in \mathbb{R}^{mn\times(m+n-r)r}$ an orthonormal basis of $\boldsymbol{I}_{mn} - \boldsymbol{P}_{\boldsymbol{V}_\perp} \otimes \boldsymbol{P}_{\boldsymbol{U}_\perp}$, $\boldsymbol{u}_j = \boldsymbol{U}\boldsymbol{e}_j^{(r)}$, $\boldsymbol{v}_j = \boldsymbol{V}\boldsymbol{e}_j^{(r)}$, $\boldsymbol{S}$ a selection matrix such that $\boldsymbol{S}\boldsymbol{S}^\top\operatorname{vec}(X) = \operatorname{vec}(\mathcal{P}_\Omega(\boldsymbol{X}))$, and $\boldsymbol{G} = \boldsymbol{Z}^\top\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{Z}$. The expected error based on the previous iteration is then

$$\mathbb{E}[\boldsymbol{\delta}^{(k+1)}] = (\boldsymbol{I} - \boldsymbol{Q})\mathbb{E}[\boldsymbol{\delta}^{(k)}] + O(\|\boldsymbol{\delta}^{(k)}\|^2).$$

However, as in [18], since the index is selected i.i.d. between iterations, the above equation can be applied repeatedly to obtain the expected error after t epochs as

$$\mathbb{E}[\boldsymbol{\delta}^{(k+(n+m)rt}] = (\boldsymbol{I} - \boldsymbol{Q})^{(n+m)rt}\mathbb{E}[\boldsymbol{\delta}^{(k)}] + O(\|\boldsymbol{\delta}^{(k)}\|^2). \tag{7}$$

III. Proposed Algorithm

In this section, we introduce momentum acceleration to matrix completion by random coordinate descent. Our proposed routine is as follows:

1) Initialize $\boldsymbol{A}_0$, $\boldsymbol{B}_0$
2) Set $\boldsymbol{A}_1 = \boldsymbol{A}_0$, $\boldsymbol{B}_1 = \boldsymbol{B}_0$
3) for $k = 1, 2, \ldots$
   a) Generate $(\tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{B}}_k)$ from iterated RCD:

$$(\tilde{\boldsymbol{A}}_k, \tilde{\boldsymbol{B}}_k) = \underbrace{\text{RCD}(\text{RCD}(\ldots \text{RCD}(\boldsymbol{A}_k, \boldsymbol{B}_k)))}_{t \text{ nested functions}}$$

   b) Update the factors as follows:

$$\boldsymbol{A}_{k+1} = \tilde{\boldsymbol{A}}_k + \beta(\boldsymbol{A}_k - \boldsymbol{A}_{k-1}), \tag{8}$$

$$\boldsymbol{B}_{k+1} = \tilde{\boldsymbol{B}}_k + \beta(\boldsymbol{B}_k - \boldsymbol{B}_{k-1}). \tag{9}$$

A pseudo-code for the routine is provided in Algorithm 3. Unlike other random coordinate descent acceleration methods in the literature, we delay the acceleration step to occur after t epochs in order to preserve the computational complexity.

Remark 1. *Applying acceleration more frequently than once per epoch increases the complexity. Accelerating individual coordinates, as in [14], increases the per-epoch computational complexity from $O(sr)$ to $O(sr) + O((n+m)^2 r^2)$ since the acceleration operation is $O((n+m)r)$ while each coordinate update is $O\left(\frac{s}{n+m}\right)$.*

Modified refactorization Due to the refactorization procedure from [11], columns of $\boldsymbol{A}_k$ are ordered and scaled versions of the left singular vectors of $\boldsymbol{X}_k$. In our analysis, we expect the column subspace to change gradually from one iteration to the next. However, a sign ambiguity issue may cause a column of $\boldsymbol{A}_k$ to change rapidly, meaning $\boldsymbol{A}_k$ and $\boldsymbol{X}_k$ would change at different rates, and the algorithm could fail. Hence, we modify the refactorization procedure to enforce a consistent sign in the factors. First, we perform the procedure in [11] (Lines 1-7 of Algorithm 2). Let the factors produced by this process be

$$(\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}) = \text{Refactor}(\boldsymbol{A}, \boldsymbol{B}) \qquad \text{(Lines 1-7)}.$$

Using an arbitrary vector $\boldsymbol{s} \in \mathbb{R}^m$ selected at the initialization stage, we adjust the sign of the columns of $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$ as follows

$$\tilde{\boldsymbol{A}} = \hat{\boldsymbol{A}}\ \text{diag}(\text{sign}(\hat{\boldsymbol{A}}^\top \boldsymbol{s})),$$
$$\tilde{\boldsymbol{B}} = \hat{\boldsymbol{B}}\ \text{diag}(\text{sign}(\hat{\boldsymbol{A}}^\top \boldsymbol{s})).$$

The column order ambiguity is addressed by ordering the singular vectors in the descending order of their respective singular values.

Theorem 1. *The optimal momentum step for acceleration in Algorithm 3 after every t epochs is*

$$\beta = \beta^* = (1 - \sqrt{1 - \rho_t})^2, \tag{10}$$

where the corresponding t-epoch unaccelerated rate $\rho_t = \rho((\boldsymbol{I} - \boldsymbol{Q})^{(n+m)rt})$ with $\boldsymbol{Q}$ from (6). Furthermore, the expected error is bounded by

$$\|\mathbb{E}[\boldsymbol{A}_k \boldsymbol{B}_k^\top - \boldsymbol{M}]\|_F^2 \leq C(\beta^*)^k, \tag{11}$$

for some finite constant $C > 0$ depending only on $\|\boldsymbol{A}_0 \boldsymbol{B}_0^\top - \boldsymbol{M}\|_F^2$ and $\boldsymbol{M}$.

A sketch of the proof of Theorem 1 can be found in Section IV. The optimal momentum parameter in (10) depends on the t-epoch rate which is defined by the matrix $\boldsymbol{Q}$ from (6). Since we do not have the solution before the algorithm is executed, we propose to estimate the true rate as a mean of random problems $\boldsymbol{M}_i$ with the same observation matrix $\boldsymbol{\Omega}$. This average can be performed on either the eigenvalue, $\hat{\mathbb{E}}[(1-\lambda_{min}(\boldsymbol{Q}(\boldsymbol{M}_i, \Omega)))^{(n+m)rt}]$, or the rate, $(\hat{\mathbb{E}}[1-\lambda_{min}(\boldsymbol{Q}(\boldsymbol{M}_i, \Omega))])^{(n+m)rt}$.

Algorithm 2: Sign-Aligned Refactor, modified from [11]

Input: Factors $\boldsymbol{A} \in \mathbb{R}^{m \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times r}$, Sign direction $\boldsymbol{s} \in \mathbb{R}^m$
Output: Normalized factors $\tilde{\boldsymbol{A}} \in \mathbb{R}^{m \times r}$ and $\tilde{\boldsymbol{B}} \in \mathbb{R}^{n \times r}$

1: $\boldsymbol{G} = (\boldsymbol{B}^\top \boldsymbol{B})(\boldsymbol{A}^\top \boldsymbol{A})$ ▷ $O\big((m+n+r)r^2\big)$
2: Perform the EVD: $\boldsymbol{G} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^{-1}$ ▷ $O(r^3)$
3: $\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^{1/2}$ ▷ $O(r)$
4: $\boldsymbol{D} = \big(\boldsymbol{\Sigma}^{-1}\boldsymbol{P}^\top \boldsymbol{A}^\top \boldsymbol{A}\boldsymbol{P}\big)^{1/2}$ ▷ $O(r^3)$
5: $\boldsymbol{Q} = \boldsymbol{D}\boldsymbol{P}^{-1}$, $\boldsymbol{Q}^{-1} = \boldsymbol{P}\boldsymbol{D}^{-1}$ ▷ $O(r^2)$
6: $\hat{\boldsymbol{A}} = \boldsymbol{A}\boldsymbol{Q}^{-1}$ ▷ $O(mr^2)$
7: $\hat{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{Q}^\top$ ▷ $O(nr^2)$
8: $\tilde{\boldsymbol{A}} = \hat{\boldsymbol{A}}\,\text{diag}(\text{sign}(\hat{\boldsymbol{A}}^\top \boldsymbol{s}))$ ▷ $O(r^2 + mr)$
9: $\tilde{\boldsymbol{B}} = \hat{\boldsymbol{B}}\,\text{diag}(\text{sign}(\hat{\boldsymbol{A}}^\top \boldsymbol{s}))$ ▷ $O(nr)$

Remark 2. *As the matrix dimensions increase, our preliminary analysis of random matrix $\boldsymbol{Q}$ suggests that its eigenvalues approach a limiting distribution with finite support. Moreover, for large matrices this distribution correctly predicts the rate of the algorithm. Further, the support of the limiting eigenvalue distribution of $\boldsymbol{Q}$ depends only on $\rho_s = \frac{s}{nm}$, $\rho_r = 1 - \sqrt{\left(1 - \frac{r}{n}\right)\left(1 - \frac{r}{m}\right)}$, and n/m.*

While we maintain that our analysis of the mean error is correct for all $t \geq 1$, we find that in some cases for small matrices and small values of t that the mean is not the only parameter relevant to the rate of convergence of typical sequences. Increasing t allows analysis of the mean to translate to analysis of particular executions of the algorithm.

IV. Error Analysis

In this section, we provide a proof sketch for Theorem 1 in Section III. We begin with an argument that the direct update on the factors $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$ leads to an asymptotically similar update on $\boldsymbol{X}_k = \boldsymbol{A}_k \boldsymbol{B}_k{}^\top$:

$$\boldsymbol{X}_{k+1} = \tilde{\boldsymbol{X}}_k + \beta(\boldsymbol{X}_k - \boldsymbol{X}_{k-1}) + O(\|\boldsymbol{X}_k - \boldsymbol{X}_*\|_F^2). \tag{12}$$

This follows from the expansion of $\boldsymbol{A}_k$, $\boldsymbol{B}_k$, and $\boldsymbol{X}_k$ around their fixed points: $\boldsymbol{A}_k = \boldsymbol{A}_* + \Delta\boldsymbol{A}_k$, $\boldsymbol{B}_k = \boldsymbol{B}_* + \Delta\boldsymbol{B}_k$, and $\boldsymbol{X}_k = \boldsymbol{X}_* + \boldsymbol{\Delta}\boldsymbol{X}_k$. Further perturbation analysis is left to the reader due to space constraints.

By projecting the error $\boldsymbol{X}_k - \boldsymbol{X}_*$ and taking the expectation as in (7), we can define $\boldsymbol{v}_k = \mathbb{E}[\boldsymbol{G}^{1/2}\boldsymbol{Z}^\top(\boldsymbol{X}_k - \boldsymbol{M})]$ and obtain

$$\boldsymbol{v}_{k+1} = (\boldsymbol{I} - \boldsymbol{Q})^{((m+n)rt)}\boldsymbol{v}_k + \beta(\boldsymbol{v}_k - \boldsymbol{v}_{k-1}) + O(\|\boldsymbol{v}_k\|^2),$$

where we can replace $\boldsymbol{X}_*$ with the explicit form of the minimizer $\boldsymbol{M}$. This equation takes the form of (5) from Section II-A where $\boldsymbol{T} = (\boldsymbol{I} - \boldsymbol{Q})^{(n+m)rt}$ and $\boldsymbol{q}_k$ is $O(\|\boldsymbol{v}_k\|^2)$ so by selecting

$$\beta = (1 - \sqrt{1 - \rho_t})^2,$$

$\boldsymbol{v}_k$ converges at the rate of $\sqrt{\beta}$. Since this transformation $\boldsymbol{G}^{1/2}\boldsymbol{Z}^\top$ is full rank, equality of norms leads to the bound in Theorem 1.

Remark 3. *If we expect $\rho_t = (1-\delta)^t$ for $\delta \ll 1$, then $\rho_t \approx 1 - t\delta$, and the expected number of epochs needed to reach a precision $\bar{\epsilon}$ is proportional to $\frac{t}{-\log(\sqrt{\beta})} \approx \sqrt{t/\delta}$. By contrast, the unaccelerated algorithm requires the number of epochs proportional to $1/\delta$.*

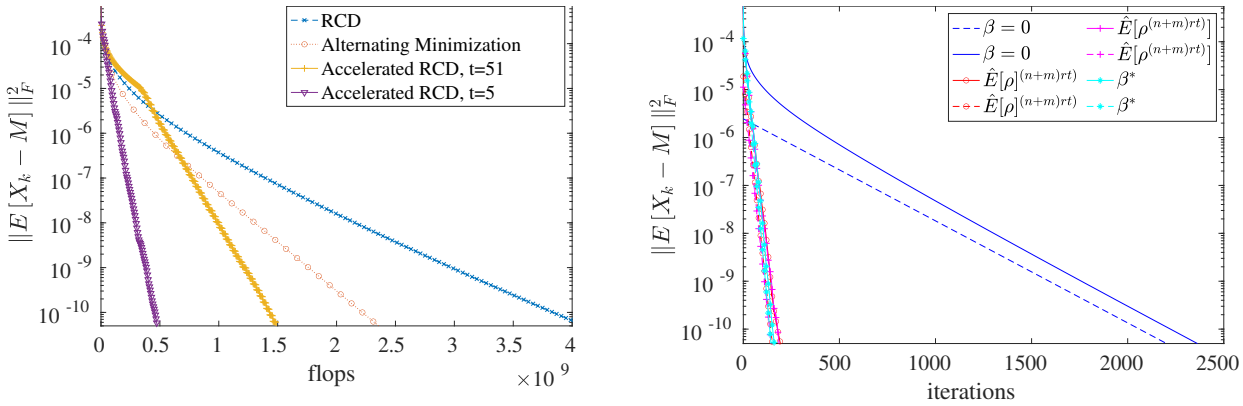

Fig. 1: Squared Frobenius error of mean distance for an 80×80 matrix. (Left) vs. flops for different algorithms. The algorithms shown are the unaccelerated RCD ($*$), optimally accelerated RCD ($+$ and $\triangledown$), and alternating minimization ($\circ$). (Right) vs. the number of iterations for $\beta = 0$ (blue) for β estimated in the two ways described ($\circ$ and $+$), and for β^* ($*$). The dashed straight lines are the predicted asymptotic rate, and the solid lines are the results generated from executing the algorithms.

Algorithm 3: Momentum Accelerated RCD for Matrix Completion

Input: $\mathcal{P}_\Omega(\boldsymbol{M})$, β, $\boldsymbol{A}_0$, $\boldsymbol{B}_0$, t
Output: $\{\boldsymbol{A}, \boldsymbol{B}\}$

Initialize $\boldsymbol{A}_c \leftarrow \boldsymbol{A}_0$, $\boldsymbol{A}_o \leftarrow \boldsymbol{A}_0$ and $\boldsymbol{B}_c \leftarrow \boldsymbol{B}_0$, $\boldsymbol{B}_o \leftarrow \boldsymbol{B}_0$
Choose random $\boldsymbol{s} \in \mathbb{R}^m$
$\boldsymbol{A}_n \leftarrow \boldsymbol{A}_c$, $\boldsymbol{B}_n \leftarrow \boldsymbol{B}_c$
for $l = 1, 2, \ldots$ **do**
  for $i = 1, 2, \ldots, t$ **do**
    $(\boldsymbol{A}_n, \boldsymbol{B}_n) \leftarrow \text{RCD}(\boldsymbol{A}_n, \boldsymbol{B}_n, \boldsymbol{s})$ ▷ $O(sr)$
  $\boldsymbol{A}_n \leftarrow \boldsymbol{A}_n + \beta(\boldsymbol{A}_c - \boldsymbol{A}_o)$ ▷ $O(mr)$
  $\boldsymbol{B}_n \leftarrow \boldsymbol{B}_n + \beta(\boldsymbol{B}_c - \boldsymbol{B}_o)$ ▷ $O(nr)$
  $\boldsymbol{A}_o \leftarrow \boldsymbol{A}_c$, $\boldsymbol{B}_o \leftarrow \boldsymbol{B}_c$
  $\boldsymbol{A}_c \leftarrow \boldsymbol{A}_n$, $\boldsymbol{B}_c \leftarrow \boldsymbol{B}_n$
$\boldsymbol{A} = \boldsymbol{A}_c$, $\boldsymbol{B} = \boldsymbol{B}_c$

V. Numerical Results

We conduct simulations to verify our analytic expression for the rate of convergence, show that our method is competitive with state-of-the-art, and confirm that choosing β via the solution-agnostic manner in Section III reaches near optimal performance.

1) Experimental settings: We selected $\boldsymbol{s} = \boldsymbol{e}_1^{(m)}$ to produce these figures. The figures use an 80×80 matrix $\boldsymbol{M}$ and Ω that were generated with $\rho_r = 0.15$ and $\rho_s = 0.29$ and the initial values $\boldsymbol{A}^0$ and $\boldsymbol{B}^0$ were generated from the true factors with random additive noise. The acceleration step was performed every t epochs as marked in Fig. 1 (left) and every $t = 5$ epochs in Fig. 1 (right). Each figure shows the squared mean error obtained by averaging over 5 runs of the same $\boldsymbol{M}$, Ω, $\boldsymbol{A}_0$, and $\boldsymbol{B}_0$.

Fig. 1 (left) shows the performance of the proposed method using the optimal β^* calculated from the true $\boldsymbol{Q}$ for the problem for each value of t compared in the number of flops to the performance of the unaccelerated version and to alternating minimization. We considered alternating minimization due to its computational efficiency [9], [19], and selected an implementation that minimizes each of the rows of each factor separately and is $O(sr^2)$. The Lightspeed Matlab toolbox[1] was used to track the number of flops for each operation of each algorithm; we used r^3 to be the exact number of flops for the eigenvalue decomposition, but this is not the dominant term.

Fig. 1 (right) uses β calculated based on the rate estimated by 10 random solution matrices using the same Ω, with both estimation methods suggested in Section III (before Remark 2). It also compares these to β^* from the true $\boldsymbol{Q}$ in terms of number of iterations, each of which is t epochs. The asymptotic rates were produced from the spectrum of the true $\boldsymbol{Q}$ based on the β selected from the estimates.

2) Results: In Fig. 1 (left) we see that the accelerated algorithm is the most computationally efficient algorithm and outperforms alternating minimization even when the unaccelerated method does not due to the small size of the problem. In addition, the figure demonstrates that various values of t allow convergence, and increasing values of t yield smoother convergence. Fig. 1 (right) shows that exact knowledge of ρ_t is not required to achieve performance gains with momentum since the accelerated versions achieve a faster rate than the unaccelerated version (blue) despite generating the value of the momentum parameter without reference to the rate of the unaccelerated algorithm. In addition, the rates in Fig. 1 (right) approach the asymptotic bounds support our theory of the asymptotic rate.

VI. Conclusion

In this paper, we present an acceleration method for randomized coordinate descent matrix completion. We avoid increasing the computational complexity via an epoch-level acceleration step. We present a convergence analysis of the proposed algorithm resulting in a closed-form expression for the asymptotic linear convergence rate. This leads to a closed-form expression for the optimal momentum parameter. Numerical experiments establish the correctness of the rate expression and establish the effectiveness of the acceleration method. In future work, we plan to derive an expression for the rate of convergence that can be expressed in terms independent of the solution, such as the ratio of known entries to the matrix size, the shape of the matrix, and the relative rank. This would ease determination of the optimal momentum parameter. In addition, this would allow an a priori prediction of the total number of flops required to achieve a certain accuracy, allowing for a comprehensive comparison between methods for matrix completion.

[1]The toolbox can be found at Tom Minka's Github: https://github.com/tminka/lightspeed

REFERENCES

[1] Nathan Srebro and Tommi Jaakkola, "Weighted low-rank approximations," in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 720–727.

[2] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye, "Tensor completion for estimating missing values in visual data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 208–220, 2013.

[3] Luong Trung Nguyen, Junhan Kim, Sangtae Kim, and Byonghyo Shim, "Localization of iot networks via low-rank matrix completion," *IEEE Trans. Commun.*, vol. 67, no. 8, pp. 5833–5847, 2019.

[4] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen, "Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval and matrix completion," in *Proc. Int. Conf. Mach. Learn.* PMLR, 2018, pp. 3345–3354.

[5] Trung Vu and Raviv Raich, "On the asymptotic linear convergence of gradient descent for non-symmetric matrix completion," in *Asilomar Conf. Signals Syst. Comput.* IEEE, 2023, pp. 1049–1053.

[6] Tian Tong, Cong Ma, and Yuejie Chi, "Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent," *Journal of Machine Learning Research*, vol. 22, no. 150, pp. 1–63, 2021.

[7] Lijun Ding and Yudong Chen, "Leave-one-out approach for matrix completion: Primal and dual analysis," *IEEE Trans. Inf. Theory*, vol. 66, no. 11, pp. 7274–7301, 2020.

[8] Trung Vu, Evgenia Chunikhina, and Raviv Raich, "On local linear convergence rate of iterative hard thresholding for matrix completion," *IEEE Trans. Signal Process.*, vol. 70, pp. 5940–5953, 2022.

[9] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proc. Annu. ACM Symp. Theory Comput.*, 2013, pp. 665–674.

[10] Moritz Hardt, "Understanding alternating minimization for matrix completion," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, 2014, pp. 651–660.

[11] Matthew Callahan, Trung Vu, and Raviv Raich, "Provable randomized coordinate descent for matrix completion," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.* IEEE, 2024, pp. 9421–9425.

[12] Yurii Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM J. Optim.*, vol. 22, no. 2, pp. 341–362, 2012.

[13] Filip Hanzely and Peter Richtarik, "Accelerated coordinate descent with arbitrary sampling and best rates for minibatches," in *Proc. Int. Conf. Artif. Intell. Stat.* 2019, pp. 304–312, PMLR.

[14] Qin Wang, Weiguo Li, Wendi Bao, and Feiyu Zhang, "Accelerated randomized coordinate descent for solving linear systems," *Mathematics*, vol. 10, no. 22, pp. 4379, 2022.

[15] David G Luenberger and Yinyu Ye, *Appendix A*, p. 507–514, Springer, 3 edition, 2008.

[16] Trung Vu and Raviv Raich, "A closed-form bound on the asymptotic linear convergence of iterative methods via fixed point analysis," *Optim. Lett.*, vol. 1, pp. 1–14, 2022.

[17] Boris T Polyak, *Multistep Methods*, pp. 65–67, New York, Optimization Software,, 1987.

[18] Robert M Gower and Peter Richtárik, "Randomized iterative methods for linear systems," *SIAM J. Matrix Anal. Appl.*, vol. 36, no. 4, pp. 1660–1690, 2015.

[19] Dongha Lee, Jinoh Oh, and Hwanjo Yu, "OCAM: Out-of-core coordinate descent algorithm for matrix completion," *Inf. Sci.*, vol. 514, pp. 587–604, 2020.