SIAM J. MATRIX ANAL. APPL.
Vol. 42, No. 4, pp. 1451–1479

AN ALTERNATING RANK-k NONNEGATIVE LEAST SQUARES FRAMEWORK (ARkNLS) FOR NONNEGATIVE MATRIX FACTORIZATION[*]

DELIN CHU[†], WENYA SHI[‡], SRINIVAS ESWAR[§], AND HAESUN PARK[§]

Abstract. Nonnegative matrix factorization (NMF) is a prominent technique for data dimensionality reduction that has been widely used for text mining, computer vision, pattern discovery, and bioinformatics. In this paper, a framework called ARkNLS (alternating rank-k nonnegativity-constrained least squares) is proposed for computing NMF. First, a recursive formula for the solution of the rank-k nonnegativity-constrained least squares (NLS) is established. This recursive formula can be used to derive the closed-form solution for the rank-k NLS problem for any integer $k \geq 1$. As a result, each subproblem for an alternating rank-k nonnegative least squares framework can be obtained based on this closed-form solution. Assuming that all matrices involved in rank-k NLS in the context of NMF computation are of full rank, two of the currently best NMF algorithms HALS (hierarchical alternating least squares) and ANLS-BPP (alternating NLS based on block principal pivoting) can be considered as special cases of ARkNLS with $k = 1$ and $k = r$ for rank-r NMF, respectively. This paper then focuses on the framework with $k = 3$, which leads to a new algorithm for NMF via the closed-form solution of the rank-3 *NLS* problem. Furthermore, a new strategy that efficiently overcomes the potential singularity problem in rank-3 *NLS* within the context of NMF computation is also presented. Extensive numerical comparisons using real and synthetic data sets demonstrate that the proposed algorithm provides state-of-the-art performance in terms of computational accuracy and CPU time.

Key words. nonnegative matrix factorization, nonnegative least squares, rank-k residue iteration, block coordinate descent method

AMS subject classifications. 65F15, 65F60, 65F10

DOI. 10.1137/20M1352405

1. Introduction.

Nonnegative matrix factorization (NMF) [34], which performs a constrained low-rank approximation of a matrix, is a commonly used effective method for data dimensionality reduction and other related tasks. Given $A \in \mathbb{R}^{m\times n}$ and a desired low-rank $r < \min\{m, n\}$ for approximation, NMF aims at finding two low-rank nonnegative matrices $U^* \in \mathbb{R}^{m\times r}$ and $V^* \in \mathbb{R}^{n\times r}$ such that

$$(U^*, V^*) = \arg\{\min \|A - UV^T\|_F^2,\ U \in \mathbb{R}^{m\times r},\ V \in \mathbb{R}^{n\times r},\ U \geq 0,\ V \geq 0\}, \tag{1.1}$$

where $X \geq 0$ means that all elements of a matrix X are nonnegative. The NMF problem was first proposed in [41] as positive matrix factorization and popularized due to [34]. By now it has become a powerful tool for data dimensionality reduction and has found important applications in many fields such as clustering [10, 30, 32, 38, 14, 19, 16], data mining [43, 52, 13], signal processing [5], computer vision [2, 22, 18],

[*]Received by the editors July 13, 2020; accepted for publication (in revised form) by July 2, 2021; published electronically October 19, 2021.
https://doi.org/10.1137/20M1352405

Funding: The work of the first author was partially supported by the NUS research grant R-146-000-187-112.

[†]Department of Mathematics, National University of Singapore, Singapore, 119076, Singapore (matchudl@nus.edu.sg).
[‡]School of Mathematics, China University of Mining and Technology, Xuzhou, 221116, Jiangsu, People's Republic of China (shiwenyaer@163.com).
[§]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0765 USA (seswar3@cc.gatech.edu, hpark@cc.gatech.edu).

bioinformatics [4, 11, 24], blind source separation [9], topic modeling [15, 52, 31, 43], spectral data analysis [42], and many others.

NMF problem (1.1) has been studied extensively, and many numerical methods are currently available. Some of the successful methods alternatingly compute the unknown low-rank factors U and V iteratively, partitioning the unknowns (U, V) into two blocks. These existing methods include the projected gradient method [36, 54], the interior point method [39], the projected quasi-Newton method [23, 53], the active-set method [3, 48, 25, 33], the active-set-like method [28, 29], and the alternating nonnegative least squares based on block principal pivoting (ANLS-BPP) method [29]. There also exist many variants of NMF (1.1) that add constraints and/or penalty terms on U and V for better interpretation and representation of the characteristics of the tasks [1, 50] including sparse NMF [9, 46, 24], orthogonal NMF [35], semi-NMF [44], joint NMF [14], nonnegative tensor factorization [7, 9, 26, 27, 45], manifold NMF [51], kernel NMF [55], regularized NMF [46, 49], symmetric NMF [20, 47, 32], integer constrained NMF [12], and so on. A comprehensive review of solving NMF can be found in [27, 50]. Some other NMF algorithms that compute the solution by partitioning the unknowns into vector blocks include the multiplicative updates method [34] and the hierarchical alternating least squares (HALS) method [7, 8, 9] (which is also called the rank-1 residue iteration (RRI) [21]). More recently, random shuffling [6] and randomized sampling techniques [17] were used to accelerate HALS/RRI, respectively.

In [27], it has been shown that most existing NMF algorithms can be explained using the block coordinate descent (BCD) framework. Among the BCD framework-based algorithms, the ANLS-BPP method [29] and HALS/RRI method [7, 8, 9, 21] have been shown to be the most effective in most situations [27]. A rank-2 residue iteration method (RTRI) is proposed, which is similar to HALS/RRI. In these methods, all subproblems that the NMF algorithm encounters are nonnegativity-constrained least squares with a matrix with one column in case of HALS/RRI or a matrix with two columns in case of RTRI.

In this paper, we establish a new framework for computing NMF where the low-rank factors U and V are partitioned into blocks where each block consists k columns where k can be any integer with $1 \leq k \leq r$. We also present a recursive formula for the solution of the rank-k nonnegativity-constrained least squares (NLS). This recursive formula can be used to derive the closed-form solution of the rank-k NLS problem for any integer $k \geq 1$. As a result, we provide a framework called ARkNLS (alternating rank-k nonnegative least squares) for NMF. Based on the framework with $k = 3$, we present a new algorithm for NMF via the closed-form solution for the rank-3 NLS problem. When $k = 1$ our framework produces the HALS/RRI. When $k = r$, the framework can be reduced to the alternating NLS method, where the subproblems can be solved using the closed-form solution based on recursion. However, as will be seen in the next section, as k becomes larger, the recursion for the closed-form solution gets significantly complicated, incurring high computational demand. Accordingly, we conclude that computation of the NMF based on ARkNLS stays efficient when k is relatively small, such as $k = 1, 2$, or 3, and for larger k, methods like ANLS-BPP are much more efficient.

In the BCD-based methods, the matrices that appear in all NLS subproblems are assumed to have full rank. In case any of these is rank deficient, it then requires a special remedy. In this paper, we will call this problem the *singularity* problem. It is well known that the closed-form solution of the rank-k NLS problem may run into a singularity problem in the HALS/RRI. Typically some small values are added to avoid zero columns in a NLS subproblem in HALS, but numerically the solution produced by HALS/RRI has been known to be very sensitive to this small value. In order to solve this singularity problem, in [37], instead of using the cyclic strategy of updating two adjacent columns in U or V, two columns of U which most violate the optimality conditions are selected in terms of the reduced gradients

$$H_\rho(U) := U - [U - \rho(UV^TV - AV)]_+, \quad H_\rho(V) := V - [V - \rho(V - \rho(VU^TU - A^TU)]_+$$

as follows: let

$$h = \left[\begin{array}{ccc} \|H_\rho(U)(:,1)\|^2 & \cdots & H_\rho(U)(:,r)\|^2 \end{array} \right],$$

set $(\hat{h}, s) = \max(h)$, $h(s) = 0$, and $(\tilde{h}, t) = \max(h)$; then $\mathbf{v}_s$ and $\mathbf{v}_t$ are updated. For the details we refer to [37]. However, theoretically this new strategy cannot completely overcome the singularity problem, since $[\begin{array}{cc} \mathbf{u}_s & \mathbf{u}_t \end{array}]$ or $[\begin{array}{cc} \mathbf{v}_s & \mathbf{v}_t \end{array}]$ can still be rank deficient in some stage of iterations. In addition, a parameter $0 < \rho \leq 1$ is involved. It is not clear how this ρ can be selected appropriately and how it affects the computed results since indices s and t depend on the value of this parameter ρ. Moreover, the idea used in [37] cannot be used to develop methods for NMF based on the rank-k residue iteration for $k \geq 3$. We present a new strategy that efficiently overcomes the potential singularity problem within the context of NMF computation for $k = 1, 2$, or 3.

Some notations and definition used in this paper are as follows. An NLS problem where the coefficient matrix has k columns and is of full rank will be called rank-k NLS. A lowercase letter, such as x, denotes a scalar; a boldface lowercase letter, such as $\mathbf{x}$, denotes a vector; a boldface uppercase, such as X, denotes a matrix. For a matrix X, $X(i,:)$, $X(:,j)$, and $X(i,j)$ denote its ith row, jth column, and (i,j)th element of X, respectively. We also let $\mathbf{x}(i)$ denote the ith element of $\mathbf{x}$. For simplicity, $X \geq 0$ indicates that all the elements of X are nonnegative, $[X]_+ = \max\{X, \mathbf{0}\}$, and $\det(X)$ is the determinant of X.

This paper is organized as follows. In section 2 the ARkNLS framework for NMF is developed. The recursive formula for rank-k NLS problem is established in section 3. Then in section 4, this framework with $k = 3$ is specifically highlighted which leads to an new algorithm ARkNLS(k=3) for NMF. Numerical experiments are provided in section 5 on some synthetic as well as real data sets to illustrate the numerical behavior of our new algorithms compared with HALS/RRI, RTRI, and ANLS-BPP. Finally some concluding remarks are given in section 6.

2. ARkNLS: A rank-*k* NLS-based NMF framework.

In this section, we present a framework called ARkNLS for NMF (1.1). ARkNLS represents a set of BCD methods for NMF where a block consists of k columns of U or k columns of V. Specifically, for NMF (1.1), suppose $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ are partitioned into q blocks each as follows:

$$\begin{aligned} &U = \left[\begin{array}{ccc} U_1 & \cdots & U_q \end{array} \right], \qquad V = \left[\begin{array}{ccc} V_1 & \cdots & V_q \end{array} \right], \\ &U_1, \ldots, U_q \in \mathbb{R}^{m\times k}, \quad V_1, \ldots, V_q \in \mathbb{R}^{n\times k}. \end{aligned} \tag{2.1}$$

For simplicity of discussion, let us assume that $r/k = q$ is an integer, for now. Later, we will show how the cases can be handled when r is not divisible by k. We have

$$f(U,V) = f(U_1, \ldots, U_q, V_1, \ldots, V_q) := \|A - UV^T\|_F^2 = \|U_1V_1^T + \cdots + U_qV_q^T - A\|_F^2.$$

Following the BCD scheme [27], f can be minimized by iteratively solving the following problems:
for $i = 1, \ldots, q$,

$$\begin{aligned} V_i &= \arg\min_{\mathcal{Y}\geq 0} f(U_1, \ldots, U_q, V_1, \ldots, V_{i-1}, \mathcal{Y}, V_{i+1}, \ldots, V_q) \\ &= \arg\min_{\mathcal{Y}\geq 0} \|U_i\mathcal{Y}^T - \left(A - \sum_{l\neq i} U_l V_l^T\right)\|_F^2 \end{aligned} \tag{2.2}$$

and for $i = 1, \ldots, q$,

$$\begin{aligned} U_i &= \arg\min_{\mathcal{Y}\geq 0} f(U_1, \ldots, U_{i-1}, \mathcal{Y}, U_{i+1}, \ldots, U_q, V_1, \ldots, V_q) \\ &= \arg\min_{\mathcal{Y}\geq 0} \|V_i\mathcal{Y}^T - \left(A - \sum_{l\neq i} U_l V_l^T\right)^T\|_F^2. \end{aligned} \tag{2.3}$$

The above yields the ARkNLS framework for NMF (1.1) which is summarized in Algorithm 2.1. The proposed ARkNLS is a general framework where we can choose any integer k. When $k = 1$, it represents a $2r$ block BCD and is reduced to HALS/RRI. When $k = r$, it represents a 2 block BCD and ANLS-BPP is one of such algorithms.

Algorithm 2.1 ARkNLS: Alternating Rank-k nonnegative least squares framework for NMF.

```
1. Assume $A \in \mathbb{R}^{m\times n}$ and $r \leq \min(m, n)$ are given.
   Initialize $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ with $\{U, V\} \geq 0$.
   Partition as in (2.1) where each block $U_i$ and $V_i$ has $k$ columns for some integer $k$
   such that $qk = r$.
   Normalize the columns of $U$.
2. Repeat
3.     For $i = 1, \ldots, q$,
           update $V_i$ by solving rank-$k$ NLS problems (2.2)
4.     For $i = 1, \ldots, q$,
           update $U_i$ by solving rank-$k$ NLS problems (2.3)
5. Until a stopping criterion is satisfied
```

The subproblems (2.2) and (2.3) are NLS with multiple right-hand sides of the form

$$\min_{Y\geq 0} \|GY - B\|_F^2, \tag{2.4}$$

where $G = U_i$ and $B = A - \sum_{l\neq i} U_l V_l^T$ for $i = 1, \ldots, q$ in (2.2). Likewise $G = V_i$ and $B = (A - \sum_{l\neq i} U_l V_l^T)^T$ for $i = 1, \ldots, q$ in (2.3). Therefore, an NLS with multiple right-hand-side vectors (2.4) is the core problem in ARkNLS, which is our focus in this section.

As illustrated in the following theorem, a significant aspect of the ARkNLS framework is that the NLS (2.4) and, accordingly, the subproblems (2.2) and (2.3), have closed-form solutions.

THEOREM 2.1. *Assume that* $B \in \mathbb{R}^{m\times n}$, $G \in \mathbb{R}^{m\times k}$, *and* $\mathbf{g}_{k+1} \in \mathbb{R}^m$ *are given, where rank*$(G) = k$ *and rank*$([\begin{array}{cc} G & \mathbf{g}_{k+1} \end{array}]) = k + 1$. *Denote the unique solution of* *the rank-k NLS problem* (2.4) *by* $S(G,B) \in \mathbb{R}^{k\times n}$. *Then the unique solution of the rank-*$(k+1)$ *NLS problem*

$$\begin{bmatrix} Y^\star \\ \mathbf{y}^\star_{k+1} \end{bmatrix} = \arg \min_{Y\geq 0, \mathbf{y}_{k+1}\geq 0} \| \begin{bmatrix} G & \mathbf{g}_{k+1} \end{bmatrix} \begin{bmatrix} Y \\ \mathbf{y}_{k+1} \end{bmatrix} - B\|_F^2 \tag{2.5}$$

is given by

$$\begin{cases} \mathbf{y}^\star_{k+1} & = \frac{1}{\|\mathbf{g}_{k+1}\|^2}\Big[\mathbf{g}_{k+1}^T\Big(B - G\cdot S\Big(G - \frac{\mathbf{g}_{k+1}\mathbf{g}_{k+1}^T}{\|\mathbf{g}_{k+1}\|^2}G, B - \frac{\mathbf{g}_{k+1}\mathbf{g}_{k+1}^T}{\|\mathbf{g}_{k+1}\|^2}B\Big)\Big)\Big]_+ \in \mathbb{R}^{1\times n}, \\ Y^\star & = S(G, B - \mathbf{g}_{k+1}\mathbf{y}^*_{k+1}). \end{cases}$$

Proof. Theorem 2.1 follows trivially from Theorem 3.1 which is proved in the next section. ∎

Theorem 2.1 enables us to derive the closed-form solution of the rank-k NLS problem (2.4) and accordingly for the subproblems (2.2) and (2.3). In addition, according to Theorem 1 in [27], which characterizes the convergence property of the BCD scheme for NMF, the convergence property of ARkNLS can be stated as follows.

THEOREM 2.2. *If U_i and V_i, for $i = 1,\ldots,q$, are of full column rank throughout all the iterations and the unique minimums in* (2.2) *and* (2.3) *are attained at each updating step, every limit point of the sequence $\{(U,\ V)^{(i)}\}$ generated by ARkNLS algorithm is a stationary point of the NMF* (1.1). *Note that this uniqueness condition is not needed when $k = r$, i.e., $q = 1$.*

It is important to note that for the above theorem to be applicable, we need to have a unique solution for each subproblem when $q > 2$ [33]. When any U_i or V_i is rank deficient, then the uniqueness of the solution will be violated, and therefore, the above theorem cannot be applied for proof of convergence to a stationary point. In case of HALS, $q = r$, and the uniqueness of the solution for all subproblems cannot be guaranteed when a block U_i or V_i (in case of HALS, these will consist of one vector) becomes rank deficient (zero vectors). For NMF algorithms based on the BCD scheme with 2 blocks like ANLS-BPP (where updates are alternated between blocks U and V), the uniqueness is not required and the convergence result of the above theorem is applicable. However, in case of NMF, due to the nonuniqueness of the NMF solution (U^*, V^*), we can modify the subproblems so that the subproblems are always of full rank and therefore, the solution for NLS is unique. More detailed discussions are presented in subsection 4.2.

3. Recursive formula for the solution of the rank-*k* NLS problem.

Problem (2.4) can be decoupled into independent NLS problems with single right-hand side vector as

$$\min_{Y(:,j)\in\mathbb{R}^{k\times 1}, Y(:,j)\geq 0} \|GY(:,j) - B(:j)\|_F^2.$$

Accordingly, in order to derive the closed-form solution of the rank-k NLS problem (2.4) (and so problems (2.2) and (2.3)), we first establish the recursive formula for the solution of the following rank-k NLS problem:

$$\min_{\mathbf{y}\geq\mathbf{0}} \|G\mathbf{y} - \mathbf{b}\|, \tag{3.1}$$

where $\mathbf{b} \in \mathbb{R}^m$, $G \in \mathbb{R}^{m\times k}$, and $rank(G) = k$.

LEMMA 1. *Given a continuous and convex function $f(\mathbf{z})$, and two nonempty closed convex sets $\mathcal{T}$ and $\mathcal{C}$ satisfying $\mathcal{T} \cap \mathcal{C} \neq \emptyset$, assume*

$$\widetilde{\mathbf{z}} = \arg\min_{\mathbf{z}\in\mathcal{T}} f(\mathbf{z}),$$

and $\widetilde{\mathbf{z}}$ is finite. Assume further that the constrained optimization problem

$$\min_{\mathbf{z}\in\mathcal{T}\cap\mathcal{C}} f(\mathbf{z}) \tag{3.2}$$

has a finite solution.
(1) If $\widetilde{\mathbf{z}} \in \mathcal{C}$, then $\mathbf{z}^* = \widetilde{\mathbf{z}} = \arg\min_{\mathbf{z}\in\mathcal{T}\cap\mathcal{C}} f(\mathbf{z})$.
(2) If $\widetilde{\mathbf{z}} \notin \mathcal{C}$, then there exists a $\mathbf{z}^* \in \mathcal{T} \cap \mathcal{C}_{edge}$ satisfying $\mathbf{z}^* = \arg\min_{\mathbf{z}\in\mathcal{T}\cap\mathcal{C}} f(\mathbf{z})$, where $\mathcal{C}_{edge}$ denotes the boundary of $\mathcal{C}$.

Proof. Part (1) is obvious. In the following we prove part (2).

Let $\widehat{\mathbf{z}} \in \mathcal{T} \cap \mathcal{C}$ be finite and

$$\widehat{\mathbf{z}} = \arg\min_{\mathcal{T}\cap\mathcal{C}} f(\mathbf{z}). \tag{3.3}$$

It is clear that part (2) follows with $\mathbf{z}^* = \widehat{\mathbf{z}}$ if $\widehat{\mathbf{z}} \in \mathcal{T} \cap \mathcal{C}_{edge}$. Otherwise, suppose that $\widehat{\mathbf{z}} \in \mathcal{T} \cap \mathcal{C}_{int}$, where $\mathcal{C}_{int}$ is the interior of $\mathcal{C}$. Note that $\widetilde{\mathbf{z}} \notin \mathcal{C}$, $\widetilde{\mathbf{z}} \in \mathcal{T}$ and $\widehat{\mathbf{z}} \in \mathcal{T} \cap \mathcal{C}_{int}$. Therefore, there exists $\mathbf{z}^* \in \mathcal{T} \cap \mathcal{C}_{edge}$ such that for some $t \in (0, 1)$, we have

$$\mathbf{z}^* = (1-t)\widetilde{\mathbf{z}} + t\widehat{\mathbf{z}} \in \mathcal{T} \cap \mathcal{C}_{edge}.$$

Furthermore, $f(\widetilde{\mathbf{z}}) \leq f(\widehat{\mathbf{z}})$. Hence, $f(\mathbf{z}^*) \leq (1-t)f(\widetilde{\mathbf{z}}) + tf(\widehat{\mathbf{z}}) \leq (1-t)f(\widehat{\mathbf{z}}) + tf(\widehat{\mathbf{z}}) = f(\widehat{\mathbf{z}})$, which with (3.3) yields that $f(\mathbf{z}^*) = f(\widehat{\mathbf{z}})$ and $\mathbf{z}^* = \arg\min_{\mathbf{z}\in\mathcal{T}\cap\mathcal{C}} f(\mathbf{z})$. ◻

LEMMA 2. *Assume $G \in \mathbb{R}^{m\times k}$, $rank(G) = k$, and $\mathbf{b} \in \mathbb{R}^m$. Then the solution of the rank-k NLS problem* (3.1) *is unique.*

Proof. The proof is trivial and we refer it to [33]. ◻

Now, we establish the recursive formula for the solution of the rank-k NLS problem (3.1).

THEOREM 3.1. *Assume $G \in \mathbb{R}^{m\times k}$, $\mathbf{g}_{k+1} \in \mathbb{R}^m$, $\mathbf{b} \in \mathbb{R}^m$ are given, where G and $\begin{bmatrix} G & \mathbf{g}_{k+1} \end{bmatrix}$ are of full column rank. Denote the unique solution of the rank-k NLS problem* (3.1) *by $s(G, \mathbf{b}) \in \mathbb{R}^k$. Then the unique solution of the rank-$(k+1)$ NLS problem*

$$\begin{bmatrix} \mathbf{y}^\star \\ y_{k+1}^\star \end{bmatrix} = \arg\min_{\mathbf{y}\geq\mathbf{0}, y_{k+1}\geq 0} \left\| \begin{bmatrix} G & \mathbf{g}_{k+1} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ y_{k+1} \end{bmatrix} - \mathbf{b} \right\| \tag{3.4}$$

is given by

$$\begin{cases} y_{k+1}^\star &= \frac{1}{\|\mathbf{g}_{k+1}\|^2}\left[\mathbf{g}_{k+1}^T\left(\mathbf{b} - G\cdot s\left(G - \frac{\mathbf{g}_{k+1}\mathbf{g}_{k+1}^T}{\|\mathbf{g}_{k+1}\|^2}G, \mathbf{b} - \frac{\mathbf{g}_{k+1}\mathbf{g}_{k+1}^T}{\|\mathbf{g}_{k+1}\|^2}\mathbf{b}\right)\right)\right]_+, \\ \mathbf{y}^\star &= s(G, \mathbf{b} - \mathbf{g}_{k+1}y_{k+1}^*). \end{cases}$$

Proof. First let us consider the optimization problem

$$\begin{bmatrix} \widetilde{\mathbf{y}} \\ \widetilde{y}_{k+1} \end{bmatrix} = \arg\min_{\mathbf{y}\geq\mathbf{0}, y_{k+1}\in\mathbb{R}} \left\| \begin{bmatrix} G & \mathbf{g}_{k+1} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ y_{k+1} \end{bmatrix} - \mathbf{b} \right\|. \tag{3.5}$$

For any given $\mathbf{y}$, the solution y_{k+1} to the optimization problem

$$\min_{y_{k+1}\in\mathbb{R}} \|\mathbf{g}_{k+1}y_{k+1} - (\mathbf{b} - G\mathbf{y})\|$$

is uniquely given by

$$y_{k+1} = \frac{\mathbf{g}_{k+1}^T(\mathbf{b} - G\mathbf{y})}{\|\mathbf{g}_{k+1}\|^2}. \tag{3.6}$$

Accordingly, the optimization problem (3.5) can be reduced to

$$\widetilde{\mathbf{y}} = \arg\min_{\mathbf{y}\geq\mathbf{0}} \|(G\mathbf{y} + \mathbf{g}_{k+1}\frac{\mathbf{g}_{k+1}^T(\mathbf{b} - G\mathbf{y})}{\|\mathbf{g}_{k+1}\|^2}) - \mathbf{b}\| = \arg\min_{\mathbf{y}\geq\mathbf{0}} \|\widetilde{G}\mathbf{y} - \widetilde{\mathbf{b}}\|, \tag{3.7}$$

where

$$\widetilde{\mathbf{b}} = \mathbf{b} - \frac{\mathbf{g}_{k+1}\mathbf{g}_{k+1}^T}{\|\mathbf{g}_{k+1}\|^2}\mathbf{b}, \quad \widetilde{G} = G - \frac{\mathbf{g}_{k+1}\mathbf{g}_{k+1}^T}{\|\mathbf{g}_{k+1}\|^2}G.$$

Moreover, it holds that

$$k+1 = rank\left(\begin{bmatrix} G & \mathbf{g}_{k+1} \end{bmatrix}\right) = rank(\mathbf{g}_{k+1}) + rank\left(\left(I - \frac{\mathbf{g}_{k+1}\mathbf{g}_{k+1}^T}{\|\mathbf{g}_{k+1}\|^2}\right)G\right) = 1 + rank(\widetilde{G}),$$

i.e., $rank(\widetilde{G}) = k$ and $\widetilde{G}$ is of full column rank. Note that according to our notation we have

$$s(\widetilde{G}, \widetilde{\mathbf{b}}) = \widetilde{\mathbf{y}} = \arg\min_{\mathbf{y}\geq\mathbf{0}} \|\widetilde{G}\mathbf{y} - \widetilde{\mathbf{b}}\|;$$

thus, it follows from (3.6) that

$$\widetilde{y}_{k+1} = \frac{\mathbf{g}_{k+1}^T(\mathbf{b} - G\widetilde{\mathbf{y}})}{\|\mathbf{g}_{k+1}\|^2} = \frac{\mathbf{g}_{k+1}^T(\mathbf{b} - G\cdot s(\widetilde{G}, \widetilde{\mathbf{b}}))}{\|\mathbf{g}_{k+1}\|^2}. \tag{3.8}$$

Consequently, the optimization problem (3.5) becomes

$$\widetilde{\mathbf{y}} = \arg\min_{\mathbf{y}\geq\mathbf{0}} \|G\mathbf{y} - (\mathbf{b} - \mathbf{g}_{k+1}\widetilde{y}_{k+1})\|,$$

which, according to our notation again, can be rewritten as

$$\widetilde{\mathbf{y}} = s(G, \mathbf{b} - \mathbf{g}_{k+1}\widetilde{y}_{k+1}). \tag{3.9}$$

Let

$$\mathcal{T} := \left\{\begin{bmatrix} \mathbf{y} \\ y_{k+1} \end{bmatrix} \mid \mathbf{y}\in\mathbb{R}^k,\ \mathbf{y}\geq 0\right\}, \qquad \mathcal{C} := \left\{\begin{bmatrix} \mathbf{y} \\ y_{k+1} \end{bmatrix} \mid \mathbf{y}\in\mathbb{R}^k,\ \mathbf{y}\geq 0,\ y_{k+1}\geq 0\right\}.$$

By Lemma 1, (3.8), and (3.9),

- if $\widetilde{y}_{k+1} \geq 0$, then

$$\begin{bmatrix} \mathbf{y}^\star \\ y_{k+1}^\star \end{bmatrix} \text{with}$$

$$y_{k+1}^* = \widetilde{y}_{k+1} = \frac{\mathbf{g}_{k+1}^T(\mathbf{b} - G\cdot s(\widetilde{G}, \widetilde{\mathbf{b}}))}{\|\mathbf{g}_{k+1}\|^2} = \frac{[\mathbf{g}_{k+1}^T(\mathbf{b} - G\cdot s(\widetilde{G}, \widetilde{\mathbf{b}}))]_+}{\|\mathbf{g}_{k+1}\|^2},$$
$$\mathbf{y}^\star = \widetilde{\mathbf{y}} = s(G, \mathbf{b} - \mathbf{g}_{k+1}\widetilde{y}_{k+1}) = s(G, \mathbf{b} - \mathbf{g}_{k+1}y_{k+1}^*)$$

  is the unique solution of the problem (3.4);

- if $\widetilde{y}_{k+1} < 0$, then the unique solution y_{k+1} of the problem (3.4) is

$$y_{k+1}^{\star} = 0 = [\widetilde{y}_{k+1}]_{+} = \frac{[\mathbf{g}_{k+1}^T(\mathbf{b} - G \cdot s(\widetilde{G}, \widetilde{\mathbf{b}}))]_{+}}{\|\mathbf{g}_{k+1}\|^2},$$

  and consequently, the unique solution $\mathbf{y}$ of the problem (3.4) is given by

$$\mathbf{y}^{\star} = \arg\min_{\mathbf{y}\geq 0} \|G\mathbf{y} - \mathbf{b}\| = \arg\min_{\mathbf{y}\geq 0} \|G\mathbf{y} - (\mathbf{b} - \mathbf{g}_{k+1}y_{k+1}^{\star})\|,$$

  which yields that

$$\mathbf{y}^{\star} = s(G, \mathbf{b} - \mathbf{g}_{k+1}y_{k+1}^{*}).$$

□

Theorem 3.1 can be used to derive the closed-form solution of the rank-k NLS problem (3.1) for any integer $k \geq 1$. We derive the closed-form solutions of the rank-2 NLS and rank-3 NLS (without recursion) in the following two corollaries.

COROLLARY 3.2. *Assume that* $G = [\ \mathbf{g}_1 \quad \mathbf{g}_2\ ] \in \mathbb{R}^{m\times 2}$ *and rank*$(G) = 2$. *Then the unique solution of the rank-*2 *NLS problem*

$$\begin{bmatrix} y_1^* \\ y_2^* \end{bmatrix} = \arg\min_{\mathbf{y}\geq 0} \|G\mathbf{y} - \mathbf{b}\| = \arg\min_{\{y_1,y_2\}\geq 0} \|y_1\mathbf{g}_1 + y_2\mathbf{g}_2 - \mathbf{b}\| \tag{3.10}$$

is given by

$$\begin{cases} y_2^* &= \frac{1}{\|\mathbf{g}_2\|^2}\left[\mathbf{b}^T\mathbf{g}_2 - \mathbf{g}_2^T\mathbf{g}_1\left[\frac{\|\mathbf{g}_2\|^2\mathbf{b}^T\mathbf{g}_1 - \mathbf{b}^T\mathbf{g}_2\cdot\mathbf{g}_2^T\mathbf{g}_1}{\|\mathbf{g}_1\|^2\|\mathbf{g}_2\|^2 - (\mathbf{g}_1^T\mathbf{g}_2)^2}\right]_{+}\right]_{+} \\ y_1^* &= \frac{1}{\|\mathbf{g}_1\|^2}[\mathbf{b}^T\mathbf{g}_1 - (\mathbf{g}_2^T\mathbf{g}_1)y_2^*]_{+}. \end{cases} \tag{3.11}$$

Proof. Since the solution of the rank-1 NLS problem $\min_{y_1\geq 0}\|y_1\mathbf{g}_1 - \mathbf{b}\|$ is given by $s(\mathbf{g}_1, \mathbf{b}) = \frac{[\mathbf{g}_1^T\mathbf{b}]_{+}}{\|\mathbf{g}_1\|^2}$, according to Theorem 3.1, the solution y_2^* to the rank-2 NLS problem (3.10) is

$$y_2^* = \frac{\left[\mathbf{g}_2^T(\mathbf{b} - \mathbf{g}_1\cdot s(\mathbf{g}_1 - \frac{\mathbf{g}_2\mathbf{g}_2^T}{\|\mathbf{g}_2\|^2}\mathbf{g}_1, \mathbf{b} - \frac{\mathbf{g}_2\mathbf{g}_2^T}{\|\mathbf{g}_2\|^2}\mathbf{b})\right]_{+}}{\|\mathbf{g}_2\|^2}.$$

Since

$$\begin{aligned} s\left(\mathbf{g}_1 - \frac{\mathbf{g}_2\mathbf{g}_2^T}{\|\mathbf{g}_2\|^2}\mathbf{g}_1, \mathbf{b} - \frac{\mathbf{g}_2\mathbf{g}_2^T}{\|\mathbf{g}_2\|^2}\mathbf{b}\right) &= \frac{\left[\left(\mathbf{g}_1 - \mathbf{g}_2\frac{\mathbf{g}_2^T\mathbf{g}_1}{\|\mathbf{g}_2\|^2}\right)^T\left(\mathbf{b} - \mathbf{g}_2\frac{\mathbf{g}_2^T\mathbf{b}}{\|\mathbf{g}_2\|^2}\right)\right]_{+}}{\left\|\mathbf{g}_1 - \mathbf{g}_2\frac{\mathbf{g}_2^T\mathbf{g}_1}{\|\mathbf{g}_2\|^2}\right\|^2} \\ &= \frac{\left[\|\mathbf{g}_2\|^2\mathbf{b}^T\mathbf{g}_1 - \mathbf{b}^T\mathbf{g}_2\cdot\mathbf{g}_2^T\mathbf{g}_1\right]_{+}}{\|\mathbf{g}_1\|^2\|\mathbf{g}_2\|^2 - (\mathbf{g}_1^T\mathbf{g}_2)^2}, \end{aligned}$$

(3.11) holds. □

COROLLARY 3.3. *Assume that* $G = [\ \mathbf{g}_1 \quad \mathbf{g}_2 \quad \mathbf{g}_3\ ] \in \mathbb{R}^{m\times 3}$ *and rank*$(G) = 3$. *Then the unique solution of the rank-*3 *NLS problem*

$$\begin{bmatrix} y_1^* \\ y_2^* \\ y_3^* \end{bmatrix} = \arg\min_{\mathbf{y}\geq 0} \|G\mathbf{y} - \mathbf{b}\| = \arg\min_{\{y_1,y_2,y_3\}\geq 0} \|y_1\mathbf{g}_1 + y_2\mathbf{g}_2 + y_3\mathbf{g}_3 - \mathbf{b}\|$$

is given by

(3.12)
$$\begin{cases} y_3^* &= \frac{1}{\|\mathbf{g}_3\|^2}[\mathbf{b}^T\mathbf{g}_3 - (\mathbf{g}_3^T\mathbf{g}_2)p - (\mathbf{g}_3^T\mathbf{g}_1)\widetilde{p}]_+ \\ y_2^* &= \frac{1}{\|\mathbf{g}_2\|^2}\Bigg[\mathbf{b}^T\mathbf{g}_2 - (\mathbf{g}_3^T\mathbf{g}_2)y_3^* \\ & \qquad -\mathbf{g}_2^T\mathbf{g}_1\left[\frac{(\mathbf{b}^T\mathbf{g}_1\|\mathbf{g}_2\|^2 - \mathbf{b}^T\mathbf{g}_2\cdot\mathbf{g}_2^T\mathbf{g}_1) - (\mathbf{g}_3^T\mathbf{g}_1\|\mathbf{g}_2\|^2 - \mathbf{g}_3^T\mathbf{g}_2\cdot\mathbf{g}_2^T\mathbf{g}_1)y_3^*}{\|\mathbf{g}_1\|^2\|\mathbf{g}_2\|^2 - (\mathbf{g}_2^T\mathbf{g}_1)^2}\right]_+\Bigg]_+ \\ y_1^* &= \frac{1}{\|\mathbf{g}_1\|^2}[\mathbf{b}^T\mathbf{g}_1 - (\mathbf{g}_3^T\mathbf{g}_1)y_3^* - (\mathbf{g}_2^T\mathbf{g}_1)y_2^*]_+, \end{cases}$$

where

$$p = \Bigg[\frac{\mathbf{b}^T\mathbf{g}_2\cdot\|\mathbf{g}_3\|^2 - \mathbf{b}^T\mathbf{g}_3\cdot\mathbf{g}_3^T\mathbf{g}_2}{\|\mathbf{g}_2\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_2)^2} - \frac{\mathbf{g}_2^T\mathbf{g}_1\cdot\|\mathbf{g}_3\|^2 - \mathbf{g}_3^T\mathbf{g}_2\cdot\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_2\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_2)^2}\left[\frac{\det([\mathbf{b},\mathbf{g}_2,\mathbf{g}_3]^TG)}{\det(G^TG)}\right]_+\Bigg]_+$$

and

$$\widetilde{p} = \left[\frac{\mathbf{b}^T\mathbf{g}_1\cdot\|\mathbf{g}_3\|^2 - \mathbf{b}^T\mathbf{g}_3\cdot\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_1\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_1)^2} - \frac{\mathbf{g}_2^T\mathbf{g}_1\cdot\|\mathbf{g}_3\|^2 - \mathbf{g}_3^T\mathbf{g}_2\cdot\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_1\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_1)^2}p\right]_+.$$

Proof. The proof is similar to Corollary 3.2 and can be found in Appendix A. □

4. ARkNLS with $k = 3$.

Based on Theorem 2.1 and Collorarie 3.2 and 3.3, ARkNLS with $k = 2$ and $k = 3$ can produce practical numerical methods for NMF. Although there is a closed-form solution for the rank-k NLS problem for any k including when $k \geq 4$, expansion of recursion to obtain a closed-form solution itself gets very complicated, and the closed-form solution becomes computationally messy. Therefore, we will focus on ARkNLS with $k = 3$ in the rest of the paper since the $k = 2$ case can be easily derived in a similar way. In addition, we will propose methods for handling the possible singularity problem for $k = 1, 2$, and 3.

In the following the closed-form solution of the rank-k NLS problem (2.2) with $k = 3$ is derived first, and then a strategy for avoiding rank deficient rank-k NLS in NMF iteration is provided. Finally these closed-form solutions and the proposed strategy lead to the efficient algorithm ARkNLS(k=3).

4.1. The closed-form solution of the rank-k NLS problem (2.2) with $k = 3$.

In this subsection, we derive an efficient algorithm to solve the rank-k NLS problem (2.2) (solving the problem (2.3) will be analogous) with $k = 3$. The efficient algorithm is derived generalizing the result presented in Corollary 3.3 to the case of NLS with multiple right-hand-side vectors which are $(A - \Sigma_{l\neq i}U_lV_l^T)$ in problem (2.2) (and $(A - \Sigma_{l\neq i}U_lV_l^T)^T$in problem (2.3)) and exploiting the special structure of these multiple right-hand-side vectors, so that redundant computations are identified and avoided.

Assume $q = r/3$ is an integer, and partition U and V into q blocks as follows:

$$U = \begin{bmatrix} U_1 & \cdots & U_q \end{bmatrix}, \qquad V = \begin{bmatrix} V_1 & \cdots & V_q \end{bmatrix},$$
$$U_1, \ldots, U_q \in \mathbb{R}^{m\times 3}, \quad V_1, \ldots, V_q \in \mathbb{R}^{n\times 3}.$$

For notational simplicity, we denote the three columns of the ith blocks U_i and V_i as follows, respectively, without additional subscripts that correspond to the columns of U and V:

$$U_i = \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix} \quad \text{and} \quad V_i = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}.$$

THEOREM 4.1. *Assume $U_i \in \mathbb{R}^{m\times 3}$ and rank$(U_i) = 3$. Then the unique solution of the rank-3 NLS problem*

$$V_i^* = \begin{bmatrix} \mathbf{v}_1^* & \mathbf{v}_2^* & \mathbf{v}_3^* \end{bmatrix} = \arg\min_{V_i\in R^{n\times 3}, V_i\geq \mathbf{0}} \|U_iV_i^T - (A - \Sigma_{l\neq i}U_lV_l^T)\|_F^2 \tag{4.1}$$

is given by

$$\begin{cases} \mathbf{v}_3^* &= \left[\mathbf{v}_3 + \frac{\mathbf{r}_3}{\|\mathbf{u}_3\|^2} + \frac{\mathbf{u}_3^T\mathbf{u}_1}{\|\mathbf{u}_3\|^2}(\mathbf{v}_1 - \widetilde{\mathbf{p}}) + \frac{\mathbf{u}_3^T\mathbf{u}_2}{\|\mathbf{u}_3\|^2}(\mathbf{v}_2 - \mathbf{p})\right]_+, \\ \mathbf{v}_2^* &= \left[\mathbf{v}_2 + \frac{\mathbf{r}_2}{\|\mathbf{u}_2\|^2} + \frac{\mathbf{u}_2^T\mathbf{u}_1}{\|\mathbf{u}_2\|^2}(\mathbf{v}_1 - \mathbf{z}) + \frac{\mathbf{u}_3^T\mathbf{u}_2}{\|\mathbf{u}_2\|^2}(\mathbf{v}_3 - \mathbf{v}_3^*)\right]_+, \\ \mathbf{v}_1^* &= \left[\mathbf{v}_1 + \frac{\mathbf{r}_1}{\|\mathbf{u}_1\|^2} + \frac{\mathbf{u}_2^T\mathbf{u}_1}{\|\mathbf{u}_1\|^2}(\mathbf{v}_2 - \mathbf{v}_2^*) + \frac{\mathbf{u}_3^T\mathbf{u}_1}{\|\mathbf{u}_1\|^2}(\mathbf{v}_3 - \mathbf{v}_3^*)\right]_+, \end{cases} \tag{4.2}$$

where

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix} = A^TU_i - VU^TU_i, \tag{4.3}$$

$$a = \mathbf{u}_2^T\mathbf{u}_1 \cdot \|\mathbf{u}_3\|^2 - \mathbf{u}_3^T\mathbf{u}_2 \cdot \mathbf{u}_3^T\mathbf{u}_1, \tag{4.4}$$

$$b = \mathbf{u}_3^T\mathbf{u}_1 \cdot \|\mathbf{u}_2\|^2 - \mathbf{u}_3^T\mathbf{u}_2 \cdot \mathbf{u}_2^T\mathbf{u}_1, \tag{4.5}$$

$$d_{12} = \|\mathbf{u}_1\|^2\|\mathbf{u}_2\|^2 - (\mathbf{u}_2^T\mathbf{u}_1)^2, \tag{4.6}$$

$$d_{13} = \|\mathbf{u}_1\|^2\|\mathbf{u}_3\|^2 - (\mathbf{u}_3^T\mathbf{u}_1)^2, \tag{4.7}$$

$$d_{23} = \|\mathbf{u}_2\|^2\|\mathbf{u}_3\|^2 - (\mathbf{u}_3^T\mathbf{u}_2)^2, \tag{4.8}$$

$$\mathbf{p} = \left[\mathbf{v}_2 + \frac{\|\mathbf{u}_3\|^2\mathbf{r}_2 - \mathbf{u}_3^T\mathbf{u}_2 \cdot \mathbf{r}_3}{d_{23}} + \frac{a}{d_{23}}\left(\mathbf{v}_1 - \left[\frac{d_{23}\mathbf{r}_1 - a\mathbf{r}_2 - b\mathbf{r}_3}{\det(U_i^TU_i)} + \mathbf{v}_1\right]_+\right)\right]_+, \tag{4.9}$$

$$\widetilde{\mathbf{p}} = \left[\mathbf{v}_1 + \frac{\|\mathbf{u}_3\|^2\mathbf{r}_1 - \mathbf{u}_3^T\mathbf{u}_1 \cdot \mathbf{r}_3}{d_{13}} + \frac{a}{d_{13}}(\mathbf{v}_2 - \mathbf{p})\right]_+, \tag{4.10}$$

$$\mathbf{z} = \left[\mathbf{v}_1 + \frac{\|\mathbf{u}_2\|^2\mathbf{r}_1 - \mathbf{u}_2^T\mathbf{u}_1 \cdot \mathbf{r}_2}{d_{12}} + \frac{b}{d_{12}}(\mathbf{v}_3 - \mathbf{v}_3^*)\right]_+. \tag{4.11}$$

Proof. Recall that

$$A - \Sigma_{l\neq i}U_lV_l^T = A - UV^T + U_iV_i^T.$$

Then we have

$$\begin{aligned} (A - UV^T + U_iV_i^T)^T\mathbf{u}_1 &= A^T\mathbf{u}_1 - VU^T\mathbf{u}_1 + \|\mathbf{u}_1\|^2\mathbf{v}_1 \\ &\quad + \mathbf{u}_2^T\mathbf{u}_1 \cdot \mathbf{v}_2 + \mathbf{u}_3^T\mathbf{u}_1 \cdot \mathbf{v}_3 \\ &= \mathbf{r}_1 + \|\mathbf{u}_1\|^2\mathbf{v}_1 + \mathbf{u}_2^T\mathbf{u}_1 \cdot \mathbf{v}_2 + \mathbf{u}_3^T\mathbf{u}_1 \cdot \mathbf{v}_3, \end{aligned} \tag{4.12}$$

$$(A - UV^T + U_iV_i^T)^T\mathbf{u}_2 = \mathbf{r}_2 + \mathbf{u}_2^T\mathbf{u}_1 \cdot \mathbf{v}_1 + \|\mathbf{u}_2\|^2\mathbf{v}_2 + \mathbf{u}_3^T\mathbf{u}_2 \cdot \mathbf{v}_3, \tag{4.13}$$

$$(A - UV^T + U_iV_i^T)^T\mathbf{u}_3 = \mathbf{r}_3 + \mathbf{u}_3^T\mathbf{u}_1 \cdot \mathbf{v}_1 + \mathbf{u}_3^T\mathbf{u}_2 \cdot \mathbf{v}_2 + \|\mathbf{u}_3\|^2\mathbf{v}_3, \tag{4.14}$$

where $\mathbf{r}_i$ for $i = 1, 2, 3$ are defined in (4.3). In addition,

$$\begin{aligned} &(A - UV^T + U_iV_i^T)^T\mathbf{u}_2 \cdot \|\mathbf{u}_3\|^2 - (A - UV^T + U_iV_i^T)^T\mathbf{u}_3 \cdot \mathbf{u}_3^T\mathbf{u}_2 \\ &\quad = \|\mathbf{u}_3\|^2\mathbf{r}_2 - \mathbf{u}_3^T\mathbf{u}_2\mathbf{r}_3 + \mathbf{u}_2^T\mathbf{u}_1\|\mathbf{u}_3\|^2 \cdot \mathbf{v}_1 \\ &\quad\quad + \|\mathbf{u}_2\|^2\|\mathbf{u}_3\|^2 \cdot \mathbf{v}_2 + \mathbf{u}_3^T\mathbf{u}_2\|\mathbf{u}_3\|^2 \cdot \mathbf{v}_3 \\ &\quad\quad - \mathbf{u}_3^T\mathbf{u}_1 \cdot \mathbf{u}_3^T\mathbf{u}_2 \cdot \mathbf{v}_1 - (\mathbf{u}_3^T\mathbf{u}_2)^2\mathbf{v}_2 - \|\mathbf{u}_3\|^2\mathbf{u}_3^T\mathbf{u}_2 \cdot \mathbf{v}_3 \end{aligned} \tag{4.15}$$

$$
\begin{aligned}
&= \|\mathbf{u}_3\|^2\mathbf{r}_2 - \mathbf{u}_3^T\mathbf{u}_2\mathbf{r}_3 + (\mathbf{u}_2^T\mathbf{u}_1\|\mathbf{u}_3\|^2 - \mathbf{u}_3^T\mathbf{u}_1\cdot\mathbf{u}_3^T\mathbf{u}_2)\mathbf{v}_1\\
&\quad + (\|\mathbf{u}_2\|^2\|\mathbf{u}_3\|^2 - (\mathbf{u}_3^T\mathbf{u}_2)^2)\mathbf{v}_2\\
&= d_{23}\mathbf{v}_2 + \|\mathbf{u}_3\|^2\mathbf{r}_2 - \mathbf{u}_3^T\mathbf{u}_2\mathbf{r}_3 + a\mathbf{v}_1,
\end{aligned}
$$

$$
\begin{aligned}
&(A - UV^T + U_iV_i^T)^T\mathbf{u}_1\cdot\|\mathbf{u}_3\|^2 - (A - UV^T + U_iV_i^T)^T\mathbf{u}_3\cdot\mathbf{u}_3^T\mathbf{u}_1\\
&\quad = d_{13}\mathbf{v}_1 + \|\mathbf{u}_3\|^2\mathbf{r}_1 - \mathbf{u}_3^T\mathbf{u}_1\mathbf{r}_3 + a\mathbf{v}_2,
\end{aligned}
\tag{4.16}
$$

$$
\begin{aligned}
&(A - UV^T + U_iV_i^T)^T\mathbf{u}_1\|\mathbf{u}_2\|^2 - (A - UV^T + U_iV_i^T)^T\mathbf{u}_2\cdot\mathbf{u}_2^T\mathbf{u}_1\\
&\quad = d_{12}\mathbf{v}_1 + \|\mathbf{u}_2\|^2\mathbf{r}_1 - \mathbf{u}_2^T\mathbf{u}_1\mathbf{r}_2 + b\mathbf{v}_3,
\end{aligned}
\tag{4.17}
$$

$$
\begin{aligned}
&(A - UV^T + U_iV_i^T)^T\mathbf{u}_1\cdot d_{23} - (A - UV^T + U_iV_i^T)^T\mathbf{u}_2\cdot a\\
&\quad - (A - UV^T + U_iV_i)^T\mathbf{u}_3\cdot b\\
&\quad = d_{23}\mathbf{r}_1 - a\mathbf{r}_2 - b\mathbf{r}_3 + \det\big(U_i^TU_i\big)\mathbf{v}_1,
\end{aligned}
\tag{4.18}
$$

where a, b, d_{23}, d_{13}, and d_{12} are defined in (4.4)–(4.6). Also from (4.12)–(4.18), we have

$$
\begin{aligned}
\mathbf{p} &= \Bigg[\frac{(A - UV^T + U_iV_i^T)^T\mathbf{u}_2\cdot\|\mathbf{u}_3\|^2 - (A - UV^T + U_iV_i^T)^T\mathbf{u}_3\cdot\mathbf{u}_3^T\mathbf{u}_2}{d_{23}}\\
&\quad - \frac{a}{d_{23}}\left[\frac{(A - UV^T + U_iV_i^T)^T\mathbf{u}_1\cdot d_{23} - (A - UV^T + U_iV_i^T)^T\mathbf{u}_2\cdot a - (A - UV^T + U_iV_i^T)^T\mathbf{u}_3\cdot b}{\det(U_i^TU_i)}\right]_+\Bigg]_+\\
&= \left[\mathbf{v}_2 + \frac{\|\mathbf{u}_3\|^2\mathbf{r}_2 - \mathbf{u}_3^T\mathbf{u}_2\cdot\mathbf{r}_3}{d_{23}} + \frac{a}{d_{23}}\left(\mathbf{v}_1 - \left[\frac{d_{23}\mathbf{r}_1 - a\mathbf{r}_2 - b\mathbf{r}_3}{\det(U_i^TU_i)} + \mathbf{v}_1\right]_+\right)\right]_+,\\
\widetilde{\mathbf{p}} &= \left[\frac{(A - UV^T + U_iV_i^T)^T\mathbf{u}_1\cdot\|\mathbf{u}_3\|^2 - (A - UV^T + U_iV_i^T)^T\mathbf{u}_3\cdot\mathbf{u}_3^T\mathbf{u}_1}{d_{13}} - \frac{a}{d_{13}}\mathbf{p}\right]_+\\
&= \left[\mathbf{v}_1 + \frac{\|\mathbf{u}_3\|^2\mathbf{r}_1 - \mathbf{u}_3^T\mathbf{u}_1\cdot\mathbf{r}_3}{d_{13}} + \frac{a}{d_{13}}(\mathbf{v}_2 - \mathbf{p})\right]_+.
\end{aligned}
$$

Then, according to Corollary 3.3, the solution $\mathbf{v}_3^*$ of problem (2.2) is

$$
\begin{aligned}
\mathbf{v}_3^* &= \frac{1}{\|\mathbf{u}_3\|^2}[(A - UV^T + U_iV_i^T)^T\mathbf{u}_3 - (\mathbf{u}_3^T\mathbf{u}_1)\widetilde{\mathbf{p}} - (\mathbf{u}_3^T\mathbf{u}_2)\mathbf{p}]_+\\
&= \left[\mathbf{v}_3 + \frac{\mathbf{r}_3}{\|\mathbf{u}_3\|^2} + \frac{\mathbf{u}_3^T\mathbf{u}_1}{\|\mathbf{u}_3\|^2}(\mathbf{v}_1 - \widetilde{\mathbf{p}}) + \frac{\mathbf{u}_3^T\mathbf{u}_2}{\|\mathbf{u}_3\|^2}(\mathbf{v}_2 - \mathbf{p})\right]_+.
\end{aligned}
$$

Furthermore, letting

$$
\begin{aligned}
\mathbf{z} &= \left[\frac{((A - UV^T + U_iV_i^T)^T\mathbf{u}_1\|\mathbf{u}_2\|^2 - (A - UV^T + U_iV_i^T)^T\mathbf{u}_2\cdot\mathbf{u}_2^T\mathbf{u}_1)}{d_{12}} - \frac{b}{d_{12}}\mathbf{v}_3^*\right]_+\\
&= \left[\mathbf{v}_1 + \frac{\|\mathbf{u}_2\|^2\mathbf{r}_1 - \mathbf{u}_2^T\mathbf{u}_1\cdot\mathbf{r}_2}{d_{12}} + \frac{b}{d_{12}}(\mathbf{v}_3 - \mathbf{v}_3^*)\right]_+,
\end{aligned}
$$

the solution $\mathbf{v}_2^*$ of problem (2.2) is

$$
\begin{aligned}
\mathbf{v}_2^* &= \frac{1}{\|\mathbf{u}_2\|^2}\left[(A - UV^T + U_iV_i^T)^T\mathbf{u}_2 - (\mathbf{u}_3^T\mathbf{u}_2)\mathbf{v}_3^* - \mathbf{u}_2^T\mathbf{u}_1\mathbf{z}\right]_+\\
&= \left[\mathbf{v}_2 + \frac{\mathbf{r}_2}{\|\mathbf{u}_2\|^2} + \frac{\mathbf{u}_2^T\mathbf{u}_1}{\|\mathbf{u}_2\|^2}(\mathbf{v}_1 - \mathbf{z}) + \frac{\mathbf{u}_3^T\mathbf{u}_2}{\|\mathbf{u}_2\|^2}(\mathbf{v}_3 - \mathbf{v}_3^*)\right]_+,
\end{aligned}
$$

and the solution $\mathbf{v}_1^*$ of problem (2.2) is

$$\begin{aligned}\mathbf{v}_1^* &= \frac{1}{\|\mathbf{u}_1\|^2}[(A - UV^T + U_iV_i^T)^T\mathbf{u}_1 - (\mathbf{u}_3^T\mathbf{u}_1)\mathbf{v}_3^* - (\mathbf{u}_2^T\mathbf{u}_1)\mathbf{v}_2^*]_+ \\ &= \left[\mathbf{v}_1 + \frac{\mathbf{r}_1}{\|\mathbf{u}_1\|^2} + \frac{\mathbf{u}_2^T\mathbf{u}_1}{\|\mathbf{u}_1\|^2}(\mathbf{v}_2 - \mathbf{v}_2^*) + \frac{\mathbf{u}_3^T\mathbf{u}_1}{\|\mathbf{u}_1\|^2}(\mathbf{v}_3 - \mathbf{v}_3^*)\right]_+ . \qquad \square\end{aligned}$$

Remark 1. Theorem 4.1 holds only when the matrix U_i has full column rank. When $rank(U_i) \neq 3$, then one or more of the following values which occur in the denominators will be zero:

- $\|\mathbf{u}_1\|$, $\|\mathbf{u}_2\|$, or $\|\mathbf{u}_3\|$ when a column of U_i is zero;
- $d_{12} = \|\mathbf{u}_1\|^2\|\mathbf{u}_3\|^2 - (\mathbf{u}_3^T\mathbf{u}_1)^2$, $d_{13} = \|\mathbf{u}_1\|^2\|\mathbf{u}_2\|^2 - (\mathbf{u}_2^T\mathbf{u}_1)^2$, or $d_{23} = \|\mathbf{u}_2\|^2\|\mathbf{u}_3\|^2 - (\mathbf{u}_3^T\mathbf{u}_2)^2$ when two of the columns of U_i are linearly dependent;
- $\det(U_i^TU_i)$.

Any of the above cases will make some of the operations not valid. In the next subsection, we propose a remedy to handle these cases.

Remark 2. Theorem 4.1 is based on the assumption that $r/3$ is an integer. If $r/3$ is not an integer, the following methods can be used. Let $q = [\frac{r}{3}]$, and denote

$$\begin{cases} V = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_r \end{bmatrix} = \begin{bmatrix} V_1 & \cdots & V_q & \mathbf{v}_r \end{bmatrix}, \\ \qquad V_1, \ldots, V_q \in \mathbb{R}^{n\times 3} \quad \text{if } r = 3q+1, \\ V = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_r \end{bmatrix} = \begin{bmatrix} V_1 & \cdots & V_q & \mathbf{v}_{r-1} & \mathbf{v}_r \end{bmatrix}, \\ \qquad V_1, \ldots, V_q \in \mathbb{R}^{n\times 3} \quad \text{if } r = 3q+2, \end{cases}$$

where $[x]$ means the nearest integer less than or equal to x. Then $V_1, \ldots, V_q$ are computed by Theorem 4.1. For the computation of $\mathbf{v}_r$ when $r = 3q+1$ or $\mathbf{v}_{r-1}$ and $\mathbf{v}_r$, there are two choices:

(a) Let
$$V_{q+1} = \begin{bmatrix} \mathbf{v}_{r-2} & \mathbf{v}_{r-1} & \mathbf{v}_r \end{bmatrix}.$$
Then V_{q+1} is computed by Theorem 4.1.

(b) $\mathbf{v}_r$ is computed by HALS/RRI when $r = 3q+1$, and $\mathbf{v}_{r-1}$ and $\mathbf{v}_r$ are computed when $r = 3q+2$ by ARkNLS with $k = 2$ when $r = 3q+2$.

In our implementation, we adopted the choice (a) above due to uniformness of singularity checking. However, even for choice (b), we can easily implement the proposed method to avoid singularity which is discussed in the next section.

4.2. Avoiding rank deficient ARkNLS in NMF iteration. The low-rank factor matrices U and V in NMF (1.1) play important roles in applications. For example, in the blind source separation, the matrix A stands for the observation matrix, the matrix U plays the role of mixing matrix, and the matrix V expresses source signals. In topic modeling [15], where A is a term-document matrix, the normalized columns of U can be interpreted as topics, and the corresponding columns of V provide the topic distribution for the documents. If the matrix V has zero columns, this implies that some of source signals may be lost through the process. In addition, if any of the columns of these factor matrices are computed as zeros, then not only does the interpretation of the result become difficult but also the computed reduced rank becomes lower than the prespecified reduced rank r which may make the approximation less accurate. More importantly, ARkNLS iteration assumes that the matrix U_i or V_i that plays the role of the coefficient matrix in each iteration of NLS has full column rank, and when this is not the case, the algorithm will break down. Hence this singularity problem must be overcome for more meaningful solutions as well as for more robust algorithms. This situation is well known especially in HALS/RRI for NMF (1.1) which is based on the RRI method where a typical problem of NLS is

$$\min_{\mathbf{v}_i\in\mathbb{R}^{n\times 1},\mathbf{v}_i\geq\mathbf{0}} \|\mathbf{u}_i\mathbf{v}_i^T-(A-\Sigma_{l\neq i}\mathbf{u}_l\mathbf{v}_l^T)\|_F^2. \tag{4.19}$$

The optimal solution vector $\mathbf{v}_i$ is given by $\mathbf{v}_i=\frac{[(A-\Sigma_{l\neq i}\mathbf{u}_l\mathbf{v}_l^T)^T\mathbf{u}_i]_+}{\|\mathbf{u}_i\|^2}$, and thus it will be zero if $(A-\Sigma_{l\neq i}\mathbf{u}_l\mathbf{v}_l^T)^T\mathbf{u}_i\leq 0$, and when this zero vector becomes the coefficient matrix in a later step, the iteration will break down. When $k\geq 1$, due to an analogous rank deficiency of the coefficient matrix U_i or V_i, the results in Theorem 4.1 cannot be applied to solve the NLS problem (2.2). When the NLS problem in the ARkNLS context involves a rank deficient matrix, there is a singularity problem.

Fortunately, in the context of NMF, we can modify the involved NLS problem during rank-k NLS iteration to avoid such a singularity problem. This is achieved by adjusting the columns of U_i and V_i such that the "new" U_i and V_i satisfy that U_i is of full column rank and that the value of

$$U_iV_i^T=\mathbf{u}_1\mathbf{v}_1^T+\mathbf{u}_2\mathbf{v}_2^T+\mathbf{u}_3\mathbf{v}_3^T$$

remains unchanged. The proposed adjustment is motivated by the monotonicity property of our algorithm ARkNLS with $k=3$. Denote the new U_i and V_i by $\mathcal{U}_i$ and $\mathcal{V}_i$, respectively, satisfying $U_iV_i=\mathcal{U}_i\mathcal{V}_i$. Define $\mathcal{V}_i^*$ as

$$\mathcal{V}_i^*=\arg\min_{\mathcal{V}\in\mathbb{R}^{n\times 3},\mathcal{V}\geq\mathbf{0}}\left\|\mathcal{U}_i\mathcal{V}^T-\left(A-\sum_{l\neq i}U_lV_l^T\right)\right\|_F^2.$$

Then we have

$$\begin{aligned}&\left\|\mathcal{U}_i(\mathcal{V}_i^*)^T-\left(A-\sum_{l\neq i}U_lV_l^T\right)\right\|_F\\ &\quad\leq\left\|\mathcal{U}_i\mathcal{V}_i^T-\left(A-\sum_{l\neq i}U_lV_l^T\right)\right\|_F=\left\|U_iV_i^T-\left(A-\sum_{l\neq i}U_lV_l^T\right)\right\|_F,\end{aligned}$$

which preserves the monotonicity property of our algorithm ARkNLS with $k=3$.

Assume that we are concerned with the blocks U_i and V_i in the iteration. Let

$$U_i=\begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3\end{bmatrix}\quad\text{and}\quad V_i=\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3\end{bmatrix}.$$

Remark 1 in the previous subsection listed the cases when U_i is rank deficient. In the following, we show how the factors U_i and V_i can be adjusted when U_i is rank deficient. Here, we adjust the matrix U_i checking the linear independence of its columns from left and right, and adjust V_i as needed.

Step (i) In the first step, if $\mathbf{u}_1=0$, then we replace this $\mathbf{u}_1$ with a nonzero vector while keeping the value of $\mathbf{u}_1\mathbf{v}_1^T$ unchanged. Assume $\mathbf{u}_1=0$, and so $\mathbf{u}_1\mathbf{v}_1^T=0$. Then set $\mathbf{u}_1(3i-2)=1,\ \mathbf{v}_1=\mathbf{0}$. Since the first column $\mathbf{u}_1$ of

U_i is the $(3i-2)$th column of U, $\mathbf{u}_1(3i-2)$ is the $(3i-2)$th element of $\mathbf{u}_1$. Now elements of $\mathbf{u}_1$ are all zeros except it has only one nonzero element $\mathbf{u}_1(3i-2) = 1$. Obviously, $\mathbf{u}_1 \neq 0$ now, and the value of $\mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T + \mathbf{u}_3\mathbf{v}_3^T$ remains unchanged.

Step (ii) Now we have $\mathbf{u}_1 \neq 0$. But if $\mathbf{u}_1$ and $\mathbf{u}_2$ are linearly dependent, then we have

$$\mathbf{u}_2 = \alpha\mathbf{u}_1, \qquad \mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T = \mathbf{u}_1(\mathbf{v}_1^T + \alpha\mathbf{v}_2^T) = \mathbf{u}_1(\mathbf{v}_1 + \alpha\mathbf{v}_2)^T,$$

where $\alpha = \frac{\|\mathbf{u}_2\|}{\|\mathbf{u}_1\|}$. We set

$$\mathbf{v}_1 = \mathbf{v}_1 + \alpha\mathbf{v}_2, \quad \mathbf{v}_2 = \mathbf{0}, \quad \text{and} \quad \mathbf{u}_2 = \mathbf{0}. \tag{4.20}$$

Note that the second column $\mathbf{u}_2$ of U_i is the $(3i-1)$th column of U; we further adjust $\mathbf{u}_2$ to ensure $\mathbf{u}_1$ and $\mathbf{u}_2$ are linearly independent:

$$\text{if } \mathbf{u}_1(3i-2) \neq 0, \text{ set } \mathbf{u}_2(3i-1) = 1; \tag{4.21}$$

$$\text{otherwise, set } \mathbf{u}_2(3i-2) = 1. \tag{4.22}$$

Clearly, the adjusted $\mathbf{u}_1$ and $\mathbf{u}_2$ are linearly independent and $\mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T$ remains unchanged, and therefore $\mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T + \mathbf{u}_3\mathbf{v}_3^T$ remains unchanged.

Step (iii) Now $\mathbf{u}_1$ and $\mathbf{u}_2$ are linearly independent. Finally, if $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ are linearly dependent in the following way:

$$rank(\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix}) = rank(\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 \end{bmatrix}) = 2, \tag{4.23}$$

then we have

$$\mathbf{u}_3 = \widetilde{\alpha}\mathbf{u}_1 + \widetilde{\beta}\mathbf{u}_2,$$

where

$$\widetilde{\alpha} = \frac{\det(\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix}^T \begin{bmatrix} \mathbf{u}_3 & \mathbf{u}_2 \end{bmatrix})}{\det(\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix}^T \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix})} \tag{4.24}$$

and

$$\widetilde{\beta} = \frac{\det(\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix}^T \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_3 \end{bmatrix})}{\det(\begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix}^T \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{bmatrix})}. \tag{4.25}$$

Actually, $\widetilde{\alpha}$ and $\widetilde{\beta}$ cannot be both negative as $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ are all nonnegative, $\mathbf{u}_1 \neq 0$, and $\mathbf{u}_2 \neq 0$. But when only one of them is negative, we need to permute the index list $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ to the list $\mathcal{I} = \begin{bmatrix} \mathcal{I}(1) & \mathcal{I}(2) & \mathcal{I}(3) \end{bmatrix}$ and change the values of $\widetilde{\alpha}$ and $\widetilde{\beta}$ such that

$$\mathbf{u}_{\mathcal{I}(3)} = \widetilde{\alpha}\mathbf{u}_{\mathcal{I}(1)} + \widetilde{\beta}\mathbf{u}_{\mathcal{I}(2)}, \quad \widetilde{\alpha} \geq 0, \quad \widetilde{\beta} \geq 0.$$

At the same time, $\mathbf{u}_{\mathcal{I}(1)}$ and $\mathbf{u}_{\mathcal{I}(2)}$ must be linearly independent. So we can adjust $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ easily, and meanwhile we only need to adjust $\mathbf{u}_{\mathcal{I}(3)}$ such that the adjusted $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ are linearly independent, and the value of $U_iV_i^T = \mathbf{u}_1\mathbf{v}_1 + \mathbf{u}_2\mathbf{v}_2 + \mathbf{u}_3\mathbf{v}_3$ remains unchanged. The permuted index list $\mathcal{I} = \begin{bmatrix} \mathcal{I}(1) & \mathcal{I}(2) & \mathcal{I}(3) \end{bmatrix}$ can be obtained as follows where $\mathcal{J}$ denotes the index list in the entire matrix U or V:

– if $\widetilde{\alpha}\widetilde{\beta} \geq 0$, then $\widetilde{\alpha} \geq 0$, $\widetilde{\beta} \geq 0$, and $\mathbf{u}_3 = \widetilde{\alpha}\mathbf{u}_1 + \widetilde{\beta}\mathbf{u}_2$. Let

$$\mathcal{I} = \left[\begin{array}{ccc} 1 & 2 & 3 \end{array}\right], \quad \mathcal{J} = \left[\begin{array}{ccc} 3i-2 & 3i-1 & 3i \end{array}\right];$$

– if $\widetilde{\alpha} < 0$, $\widetilde{\beta} > 0$, then $\mathbf{u}_2 = -\frac{\widetilde{\alpha}}{\widetilde{\beta}}\mathbf{u}_1 + \frac{1}{\widetilde{\beta}}\mathbf{u}_3$ with $-\frac{\widetilde{\alpha}}{\widetilde{\beta}} > 0$ and $\frac{1}{\widetilde{\beta}} > 0$. Let

$$\widetilde{\alpha} = -\widetilde{\alpha}/\widetilde{\beta} > 0, \quad \widetilde{\beta} = 1/\widetilde{\beta} > 0, \quad \mathcal{I} = \left[\begin{array}{ccc} 1 & 3 & 2 \end{array}\right],$$
$$\mathcal{J} = \left[\begin{array}{ccc} 3i-2 & 3i & 3i-1 \end{array}\right];$$

– if $\widetilde{\alpha} > 0$, $\widetilde{\beta} < 0$, then $\mathbf{u}_1 = -\frac{\widetilde{\beta}}{\widetilde{\alpha}}\mathbf{u}_2 + \frac{1}{\widetilde{\alpha}}\mathbf{u}_3$ with $-\frac{\widetilde{\beta}}{\widetilde{\alpha}} > 0$ and $\frac{1}{\widetilde{\alpha}} > 0$. Let

$$\widetilde{\alpha} = -\widetilde{\beta}/\widetilde{\alpha} > 0, \quad \widetilde{\beta} = 1/\widetilde{\alpha} > 0, \quad \mathcal{I} = \left[\begin{array}{ccc} 2 & 3 & 1 \end{array}\right],$$
$$\mathcal{J} = \left[\begin{array}{ccc} 3i-1 & 3i & 3i-2 \end{array}\right].$$

Now we have

$$\mathbf{u}_{\mathcal{I}(3)} = \widetilde{\alpha}\mathbf{u}_{\mathcal{I}(1)} + \widetilde{\beta}\mathbf{u}_{\mathcal{I}(2)}, \quad \widetilde{\alpha} \geq 0,\ \widetilde{\beta} \geq 0,$$

and $\mathbf{u}_{\mathcal{I}(j)}$ is the $\mathcal{J}(j)$th column of U, $j = 1, 2, 3$. Moreover, we also have

$$\begin{aligned}\mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T + \mathbf{u}_3\mathbf{v}_3^T &= \mathbf{u}_{\mathcal{I}(1)}\mathbf{v}_{\mathcal{I}(1)}^T + \mathbf{u}_{\mathcal{I}(2)}\mathbf{v}_{\mathcal{I}(2)}^T + \mathbf{u}_{\mathcal{I}(3)}\mathbf{v}_{\mathcal{I}(3)}^T \\ &= \mathbf{u}_{\mathcal{I}(1)}(\mathbf{v}_{\mathcal{I}(1)} + \widetilde{\alpha}\mathbf{v}_{\mathcal{I}(3)})^T + \mathbf{u}_{\mathcal{I}(2)}(\mathbf{v}_{\mathcal{I}(2)} + \widetilde{\beta}\mathbf{v}_{\mathcal{I}(3)})^T.\end{aligned}$$

Thus, we set

$$\mathbf{v}_{\mathcal{I}(1)} = \mathbf{v}_{\mathcal{I}(1)} + \widetilde{\alpha}\mathbf{v}_{\mathcal{I}(3)}, \quad \mathbf{v}_{\mathcal{I}(2)} = \mathbf{v}_{\mathcal{I}(2)} + \widetilde{\beta}\mathbf{v}_{\mathcal{I}(3)}, \quad \mathbf{v}_{\mathcal{I}(3)} = \mathbf{0}, \quad \mathbf{u}_{\mathcal{I}(3)} = \mathbf{0}.$$

Then $\mathbf{v}_{\mathcal{I}(1)}$ and $\mathbf{v}_{\mathcal{I}(2)}$ are both nonnegative vectors, $\mathbf{u}_{\mathcal{I}(1)}$ and $\mathbf{u}_{\mathcal{I}(2)}$ are linearly independent, and the value of $\mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T + \mathbf{u}_3\mathbf{v}_3^T$ remains unchanged.

Now we adjust $\mathbf{u}_{\mathcal{I}(3)}$ so that $\mathbf{u}_1, \mathbf{u}_2$, $\mathbf{u}_3$ are linearly independent and the value of $\mathbf{u}_1\mathbf{v}_1^T + \mathbf{u}_2\mathbf{v}_2^T + \mathbf{u}_3\mathbf{v}_3^T$ remains unchanged:

$$\text{If } \mathbf{u}_{\mathcal{I}(1)}(\mathcal{J}(1))\mathbf{u}_{\mathcal{I}(2)}(\mathcal{J}(2)) - \mathbf{u}_{\mathcal{I}(1)}(\mathcal{J}(2))\mathbf{u}_{\mathcal{I}(2)}(\mathcal{J}(1)) \neq 0, \quad \text{set } \mathbf{u}_{\mathcal{I}(3)}(\mathcal{J}(3)) = 1; \tag{4.26}$$

$$\text{else, if } \mathbf{u}_{\mathcal{I}(1)}(\mathcal{J}(1)) + \mathbf{u}_{\mathcal{I}(2)}(\mathcal{J}(1)) = 0, \text{ set } \mathbf{u}_{\mathcal{I}(3)}(\mathcal{J}(1)) = 1; \tag{4.27}$$

$$\text{else, if } \mathbf{u}_{\mathcal{I}(1)}(\mathcal{J}(1)) + \mathbf{u}_{\mathcal{I}(2)}(\mathcal{J}(1)) \neq 0, \text{ set } \mathbf{u}_{\mathcal{I}(3)}(\mathcal{J}(2)) = 1. \tag{4.28}$$

Through the above three steps, U_i is of full column rank, and so Theorem 4.1 can be applied.

4.3. Algorithm ARkNLS with $k = 3$.

Denote

$$H = A^T U_i, \qquad M = U^T U.$$

Note that for solving (4.1) via (4.2), only two matrix-matrix products H and M are needed, and all of $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$, a, b, d_{12}, d_{13}, and d_{23} can be invoked via H and M as shown in Theorem 4.1. Moreover, the singularity problem discussed in the subsection above can be detected via matrices H and M which will have to be computed anyway for other parts of the computation.

In the following we illustrate the implementation of adjustment of vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$ first and then present algorithm ARkNLS(k=3).

- In step (i), whether $\mathbf{u}_1 = 0$ can be checked by checking whether $\mathbf{u}_1^T\mathbf{u}_1 = M(3i-2, 3i-2)$ is zero. After $\mathbf{u}_1$ is adjusted, the elements of H and M should also be adjusted:

$$H(:,1) = A(3i{-}2,:)^T, \quad M(:,3i{-}2) = U(3i{-}2,:)^T, \quad M(3i{-}2,:) = U(3i{-}2,:).$$

- In step (ii), the linear dependence of $\mathbf{u}_1$ and $\mathbf{u}_2$ can be checked by checking whether $\det([\ \mathbf{u}_1 \quad \mathbf{u}_2\ ]^T[\ \mathbf{u}_1 \quad \mathbf{u}_2\ ]) = \det(M(3i-2 : 3i-1, 3i-2 : 3i-1))$ is zero. Moreover,

$$\alpha = \frac{\|\mathbf{u}_2\|}{\|\mathbf{u}_1\|} = \sqrt{\frac{M(3i-1,3i-1)}{M(3i-2,3i-2)}}.$$

Furthermore, after $\mathbf{u}_2$ is adjusted, the elements of H and M should also be adjusted:

  – Corresponding to (4.21),

$$H(:,2) = A(3i-1,:)^T,\ M(:,3i-1) = U(3i-1,:)^T,$$
$$M(3i-1,:) = U(3i-1,:),$$

  – Corresponding to (4.22),

$$H(:,2) = A(3i-2,:)^T,\ M(:,3i-1) = U(3i-2,:)^T,$$
$$M(3i-1,:) = U(3i-2,:).$$

- In step (iii), the condition (4.23) can be checked from the relationship

$$\det([\ \mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3\ ]^T[\ \mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3\ ]) = \det(M(3i-2 : 3i, 3i-2 : 3i)) = 0.$$

In addition,

$$\begin{aligned}\widetilde{\alpha} &= \frac{\det([\ \mathbf{u}_1 \quad \mathbf{u}_2\ ]^T[\ \mathbf{u}_3 \quad \mathbf{u}_2\ ])}{\det([\ \mathbf{u}_1 \quad \mathbf{u}_2\ ]^T[\ \mathbf{u}_1 \quad \mathbf{u}_2\ ])} \\ &= \frac{M(3i-1,3i-1)M(3i-2,3i) - M(3i-1,3i)M(3i-2,3i-1)}{M(3i-1,3i-1)M(3i-2,3i-2) - (M(3i-1,3i-2))^2},\end{aligned}$$

and

$$\begin{aligned}\widetilde{\beta} &= \frac{\det([\ \mathbf{u}_1 \quad \mathbf{u}_2\ ]^T[\ \mathbf{u}_1 \quad \mathbf{u}_3\ ])}{\det([\ \mathbf{u}_1 \quad \mathbf{u}_2\ ]^T[\ \mathbf{u}_1 \quad \mathbf{u}_2\ ])} \\ &= \frac{M(3i-2,3i-2)M(3i-1,3i) - M(3i-1,3i-2)M(3i-2,3i)}{M(3i-1,3i-1)M(3i-2,3i-2) - (M(3i-1,3i-2))^2}.\end{aligned}$$

After $\mathbf{u}_{\mathcal{I}(3)}$ is adjusted, the elements of H and M should also be adjusted:

  – Corresponding to (4.26),

$$H(:,\mathcal{I}(3)) = A(\mathcal{J}(3),:)^T, \quad M(:,\mathcal{J}(3)) = U(\mathcal{J}(3),:)^T,$$
$$M(\mathcal{J}(3),:) = U(\mathcal{J}(3),:).$$

  – Corresponding to (4.27),

$$H(:,\mathcal{I}(3)) = A(\mathcal{J}(1),:)^T, \quad M(:,\mathcal{J}(3)) = U(\mathcal{J}(1),:)^T,$$
$$M(\mathcal{J}(3),:) = U(\mathcal{J}(1),:).$$

– Corresponding to (4.28),

$$H(:,\mathcal{I}(3)) = A(\mathcal{J}(2),:)^T, \quad M(:,\mathcal{J}(3)) = U(\mathcal{J}(2),:)^T,$$
$$M(\mathcal{J}(3),:) = U(\mathcal{J}(2),:).$$

Theorem 4.1 and discussions above lead to the algorithm ARkNLS(k=3) for NMF problem, which is summarized in Algorithm 4.1.

4.4. Computational complexity of ARkNLS with $k = 3$. We briefly discuss the per iteration computational complexity of Algorithm 4.1. The matrix multiplications needed to compute $M = U^TU$ and $H = A^TU$ cost $2mr^2$ flops and $2mnr$ flops, respectively. Adjusting the rank deficiency of U requires only $\approx 6n$ of flops and data movement per U_i to replace entries of M and H. Solving for V_i using Theorem 4.1 involves calculating $A^TU_i - VU^TU_i$ which can be computed using H and M using $6nr + 3n$ flops. Finally solving for V_i involves a constant number of vector

Algorithm 4.1 Alternating rank-3 NLS for NMF (ARkNLS(k=3)).

```
1. Given an $m$-by-$n$ nonnegative matrix $A$, initialize $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ with
   nonnegative elements and normalized the columns of $U$. Let $q = \mathtt{floor}(\frac{r}{3})$, where
   $\mathtt{floor}(\frac{r}{3})$ rounds $\frac{r}{3}$ to the nearest integer less than or equal to $\frac{r}{3}$.
2. **Repeat**
3.   **For** $i = 1 : q$ **do**    % $U_i = [\,\mathbf{u}_1 \ \mathbf{u}_2 \ \mathbf{u}_3\,]$, $\mathbf{u}_1 = U(:,3i-2)$,
                                   $\mathbf{u}_2 = U(:,3i-1)$, $\mathbf{u}_3 = U(:,3i)$.
                                 % $V_i = [\,\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3\,]$, $\mathbf{v}_1 = V(:,3i-2)$,
                                   $\mathbf{v}_2 = V(:,3i-1)$, $\mathbf{v}_3 = V(:,3i)$.
4.     Adjust the columns of $U_i$ and $V_i$ such that the adjusted $U_i$ is of full column rank
       and the value of $U_iV_i^T$
         remains unchanged.
5.     Compute $V_i$ for (4.1) by (4.2).
6.   **End for**
7.   **If** mod$(r,3) = 1$ or $2$ **then** compute $\mathbf{v}_{r-2}$, $\mathbf{v}_{r-1}$, $\mathbf{v}_r$ via the above steps 4–5. **End if**
       % mod$(r,3)$ returns the remainder after division of r by 3.

       % In the case mod$(r,3) = 1$, $\mathbf{u}_{r-2}$ and $\mathbf{u}_{r-1}$ must be linearly independent
       % after step 5, we only need to check whether $\mathbf{u}_{r-2}$, $\mathbf{u}_{r-1}$ and $\mathbf{u}_r$ are
       % linearly independent before updating the last three columns of $V$.

       % In the case mod$(r,3) = 2$, $\mathbf{u}_{r-2} \neq 0$ after step 5, we only need to check
       % whether $\mathbf{u}_{r-2}$ and $\mathbf{u}_{r-1}$, and $\mathbf{u}_{r-2}$, $\mathbf{u}_{r-1}$ and $\mathbf{u}_r$ are linearly independent
       % before updating the last three columns of $V$.
8.   Replace $U$ with $V$ and $V$ with $U$, $A$ with $A^T$, repeat steps 3–7.
9. **Until** a stopping criterion is satisfied
```

operations taking $\approx 52n$ flops. These solves are updated for every V_i block giving us a total time for updating V to be $2mnr+2mr^2+\frac{r}{3}(6nr+61n) = 2mnr+O\left((n+m)r^2\right)$. Performing a similar analysis for updating U we get the overall per iteration computational complexity of Algorithm 4.1 to be $4mnr + O\left((m+n)r^2\right)$ flops. If r is small, the computation is dominated by the matrix multiplication operations involving A.

5. Numerical experiments.

In this section we provide numerical experiments on synthetic data sets and real world text and image data sets. All methods were implemented in MATLAB (version R2017a), and the experiments were conducted on a server with two Intel(R) Xeon(R) CPU ES-2680 v3 CPUs and 377GB RAM. We compared the following algorithms for NMF.

1. (ARk) ARkNLS with $k = 3$, the method proposed in this paper.
2. (HALS) Cichocki and Phan's hierarchical alternating least squares algorithm [7].
3. (BPP) Kim and Park's block principal pivoting method [24].
4. (RTRI) Liu and Zhou's rank-2 residual iteration method [37].

Prior work [27, 7, 9] has shown that HALS and BPP are two of the most effective methods for NMF. RTRI is an algorithm with a similar style as our proposed method and thus serves as good benchmark as well. We implemented the RTRI algorithm and utilized the BPP and HALS implementation provided by Kim, He, and Park [27]. In all our experiments $A \in \mathbb{R}_+^{m\times n}$ refers to the input matrix with r being the approximation rank.

5.1. Comparison criteria: Computational complexity.

The computational cost of each iteration for various methods is shown in Table 5.1. Here, each iteration refers to updating all entries of U and V once. The algorithms we compared use the BCD framework for NMF, ranging from 2 blocks (where each block is the entire matrix U or V) in the case of BPP to $2r$ blocks (where each block is a column of U or V) in the case of HALS. ARk and RTRI lie somewhere in between these two extremes. It is important to note that the underlying problems for one complete updating of U or V in these algorithms are very different. For example, updating every column of U once in HALS is not equivalent to updating the entire U all at once in BPP and may not even be related. This is easy to see since updating U in BPP requires an NLS solution but we do not reach an NLS solution by updating each column of U once for the entire U as in HALS. In addition, the outer iteration for the larger problem of NMF can take very different paths to the solution, and the total number of iterations would depend on the problems involved in the inner iteration. For the above reasons, any comparison across the algorithms based on flop count of each iteration for updating the matrix U or V once would be meaningless. However, a fair comparison of speed across the NMF algorithms can come from measuring the objective function decrease (or other relevant measures) over time. Nevertheless, we provide the complexity of updating one factor (i.e., U or V) once for each algorithm in Table 5.1. The purpose of the table is to give the reader a sense of the work performed in each iteration of each method, but it will not be meaningful to compare the flop counts per iteration across the methods.

The complexity in Table 5.1 for each method is divided into 3 terms. The first term is interaction with input A. The second is the computation of the Gramian of U or V. The final term is the update rule for a column/row of U or V and is often simple in nature. In BPP, the update rule is for the entire matrix U or V and can be exponential in the worst case. However, the algorithm is often fast in practize and is a standard benchmark for many NMF applications [27]. In RTRI, we need to apply the input matrix twice to every column of U and V, resulting in the leading

Table 5.1
Computational complexity of updating $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ once.

Algorithm	FLOPs
ARk	$4mnr + 4(m+n)r^2 + O((m+n)r)$
HALS	$4mnr + 4(m+n)r^2 + O((m+n)r)$
BPP	$4mnr + 4(m+n)r^2 + O(2^r)$
RTRI	$8mnr + 8(m+n)r^2 + O((m+n)r)$

Table 5.2
Data sets.

Data set	Size	Sparsity
TDT2	36,771 × 9,394	99.65%
Reuters	18,933 × 8,293	99.75%
20Newsgroups	26,214 × 18,846	99.66%
ORL	10,304 × 400	0.01%
Facescrub	9,216 × 22,631	0.39%
YaleB	10,000 × 2,432	2.70%
Caltech256	9,216 × 30,607	1.00%

term of $8mnr$. When running NMF on large scale data it is often the leading term, application of the input matrix A, which is the bottleneck computation.

5.2. Data sets. We use seven real world data sets in our experiments: three sparse text data sets[1] (TDT2, Reuters21578, and 20Newsgroups) and four dense image data sets (ORL, Facescrub, YaleB, and Caltech256). A summary of their characteristics can be found in Table 5.2. The text data is represented as a term-document matrix, and images are represented as a vector of pixels. In addition, we test the methods on various synthetic data sets as described later in subsection 5.5. Detailed descriptions of the real world data sets are as follows:

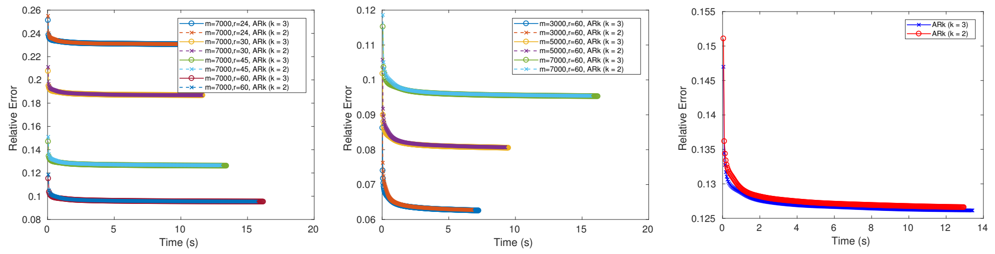
1. TDT2: The National Institute of Standards and Technology Topic Detection and Tracking corpus consists of news articles collected during 1998 and taken from various sources including television programs, radio programs, and news wires. The articles are classified into 96 categories. Documents appearing in multiple categories are pruned leaving us with 9,394 documents and 30 classes of documents in total.
2. Reuters: We use the ModApte version of the Reuters21578 corpus. It consists of articles appearing in the 1987 Reuters news wire. Retaining documents with only single labels leaves us with 8,293 documents in 65 categories.
3. 20Newsgroups: It is a collection of newsgroup documents partitioned across 20 different categories with 18,846 documents.
4. ORL[2]: AT&T Laboratories Cambridge collected 400 facial images of 40 different people with different expressions and postures. Each image has 92×112 pixels resulting in matrix of dimension $10,304 \times 400$.
5. Facescrub: This database contains 106,863 photos of 530 celebrities, 265 whom are male and 265 female [40]. The initial images that make up this data set were procured using Google Image Search. Subsequently, they were processed using the Haarcascade-based face detector from OpenCV 2.4.7 on the images to obtain a set of faces for each celebrity name, with the requirement that a face must be at least 96×96 pixels. In our experiment, we use photos from 256 male celebrities and scale the images to 96×96 pixels.

[1]The text data sets are available at http://www.cad.zju.edu.cn/home/dengcai/.
[2]https://github.com/fengbingchun/NN_Test.

6. YaleB[3]: 5,760 single light source images of 10 subjects were collected under 576 viewing conditions. The images have normal, sleepy, sad, and surprising expressions. A subset of 38 persons with 64 images per people, i.e., 2,432 images, is used in this paper. We scale the images to 100×100 pixels each.
7. Caltech256[4]: This corpus is a set of 256 object categories containing a total of 30,607 images. They were collected by choosing a set of object categories, downloading examples from Google Images, and then manually screening out all images that did not fit the category. Images are scaled to 96×96 pixels in our experiments.

We also use synthetic data for additional benchmarks. The details of these sets can be found in subsections 5.3 and 5.5.3.

5.3. ARkNLS with $k = 2$ versus $k = 3$. As stated earlier, although ARkNLS can be developed for any integer k, we focus on the choices of $k = 2$ and $k = 3$ for efficiency of closed-form solutions. We test the cases of ARkNLS for $k = 2$ and $k = 3$ on synthetic low-rank matrices. The synthetic matrices are created as $A = WH^T + N$ where $W \in \mathbb{R}_+^{m\times r}$ and $H \in \mathbb{R}_+^{n\times r}$ are random nonnegative matrices where the columns of W have unit norm and $N \in \mathbb{R}^{m\times n}$ is a random Gaussian matrix with 0 mean and 0.03 standard deviation. We ensure that A is nonnegative by replacing negative values with 0. We fix the number of columns n to 5,000 and vary the number of rows m. We run ARkNLS for 300 iterations of updating every column of U and V. The results of these experiments can be seen in Figure 5.1.



(a) Varying r for $m = 7,000$. (b) Varying m for $r = 60$. (c) $m = 7,000$ and $r = 45$.

FIG. 5.1. *Synthetic experiments for selecting k for ARk. Both $k = 2$ and $k = 3$ perform similarly with $k = 3$ obtaining marginally better approximations in slightly longer time.*

From Figure 5.1(a) and (b) we can see that the choices of k do not affect the running time and convergence characteristics of ARkNLS too much. Relative residual is measured as $\|A - UV^T\|_F/\|A\|_F$. We show a particular case of the runs in 5.1(c). Here we can see that $k = 3$ achieves slightly higher accuracy and runs at about the same time. This trend is true for all configurations of m and r. This makes sense intuitively since $k = 3$ updates one more column per block than $k = 2$. The major computational bottleneck in both variations comes from matrix multiplications involving A (see subsection 4.4) and does not vary with k. Hence we focus only on the $k = 3$ setting for the rest of our experiments.

5.4. Choice of NLS solvers (ARkNLS versus BPP). Our next experiment compares the choice of the NLS solver used in the $k = 3$ blocks generated in our algorithm. BPP is a general NLS solver and is used as a baseline to measure ARkNLS

[3]http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/Yale%20Face%20Database.htm.
[4]http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

performance. We repeat the experiments used in subsection 5.3 with using both methods as the NLS solver. The results can be seen in Figure 5.2. Since both ARkNLS and BPP are exact NLS solvers they both result in the same solution. Therefore the only difference is in execution time where ARkNLS is very effective. ARkNLS is consistently 3× faster than BPP. This is expected since BPP is an iterative method whereas ARkNLS finds a closed-form solution for each subproblem. With $k = 3$, BPP's advantage of moving multiple variables in and out of the active set is lost. This choice of k also harms BPP's convergence behavior since we are no longer solving a 2 block BCD problem but a multiblock one, making it more difficult to find a good solution in terms of residual as seen in other experiments (see subsection 5.5).

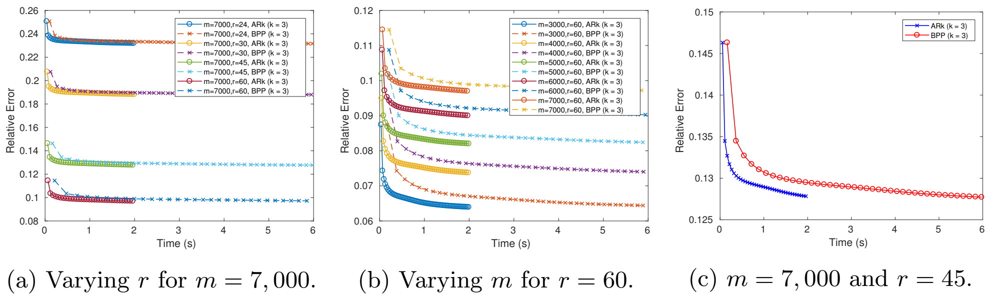

(a) Varying r for $m = 7,000$. (b) Varying m for $r = 60$. (c) $m = 7,000$ and $r = 45$.

FIG. 5.2. *Synthetic experiments for selecting the NLS solver used for each $k = 3$ block. ARk is consistently* 3× *faster than BPP for a variety of inputs.*

5.5. Experiments on synthetic data. The convergence behavior of the different NMF algorithms is compared on synthetic matrices in the following experiments.

5.5.1. Experiments on dense synthetic data. The dense synthetic matrices are created in the same manner as described in subsection 5.3. Defining a stopping criteria for iterative algorithms like NMF is often a tricky task. Many options exist [27], but for comparison purposes in this section we run all algorithms for 100 iterations and observe their convergence behavior. We ran our tests with $n = 15,000$ and vary m from 5,000 to 30,000 in increments of 5,000. r is fixed as one of 30,60, and 90. We split the input matrices into the Short-Fat ($m < n$), Square ($m = n$), and Tall-Skinny ($m > n$) cases. Only particular instances of each case are shown in Figures 5.3 to 5.5 with the results being similar in the other experiments.

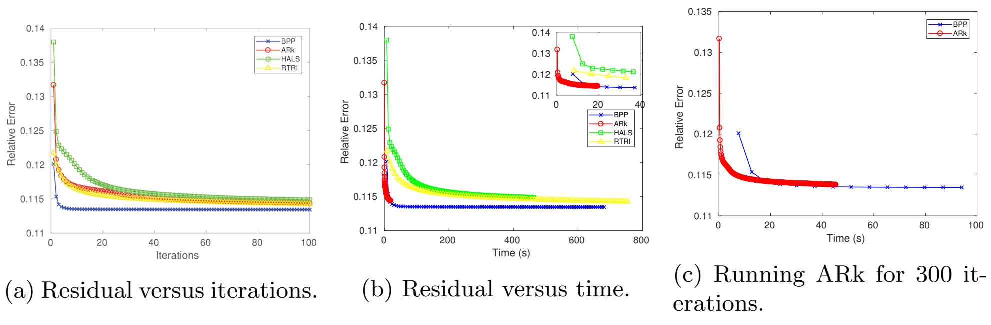
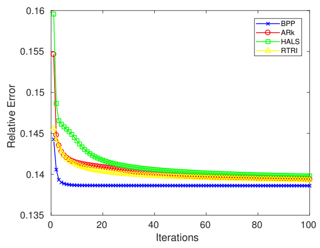
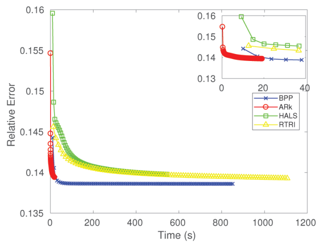

(a) Residual versus iterations. (b) Residual versus time. (c) Running ARk for 300 iterations.

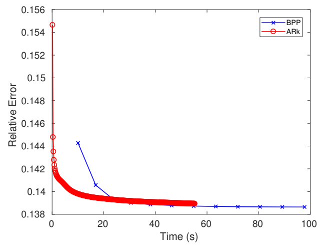
FIG. 5.3. *Short-Fat case ($m < n$): $A \in \mathbb{R}_+^{10,000\times 15,000}$ with $r = 60$. BPP performs the best in terms of residual with ARk being next best and reaching similar residuals much faster than the other methods.*



(a) Residual versus iterations.

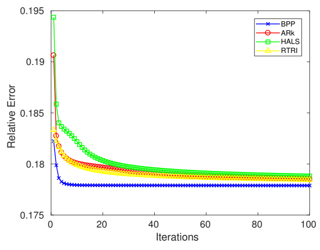


(b) Residual versus time.

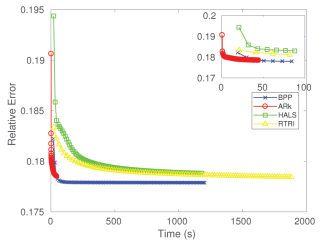


(c) Running ARk for 300 iterations.

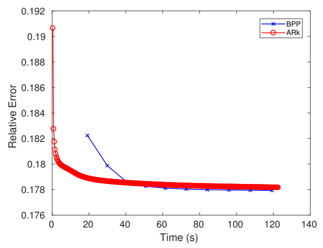
FIG. 5.4. *Square case* ($m = n$)*:* $A \in \mathbb{R}_+^{15{,}000 \times 15{,}000}$ *with* $r = 60$*. BPP performs the best in terms of residual with ARk being next best and reaching similar residuals much faster than the other methods.*



(a) Residual versus iterations.



(b) Residual versus time.



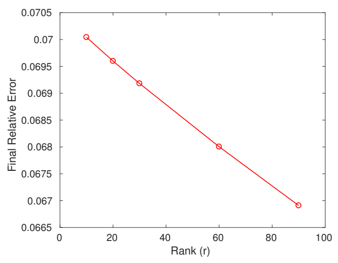
(c) Running ARk for 300 iterations.

FIG. 5.5. *Tall-Skinny case* ($m > n$)*:* $A \in \mathbb{R}_+^{25{,}000 \times 10{,}000}$ *with* $r = 60$*. BPP performs the best in terms of residual with ARk being next best and reaching similar residuals much faster than the other methods.*

Figure 5.3 shows the Short-Fat case with $m = 10{,}000$ and $r = 60$. BPP achieves the lowest residual while HALS, RTRI, and ARk perform slightly worse as seen in Figure 5.3(a). Figure 5.3(b) shows the convergence with respect to time. All algorithms converge very quickly, and BPP and ARk show the fastest drops in residual. It can be clearly seen that ARk is the most efficient of the algorithms, often completing over half of its iterations before the others complete their first iteration. From Figure 5.3(a) and (b) it looks like ARk might reach a lower residual if we allow it to run for a few more iterations and so allow it to run till 300 iterations in Figure 5.3(c) and compare it to BPP. BPP is still better, but the difference is marginal.

Figures 5.4 and 5.5 show the Square and Tall-Skinny cases, respectively. The observations from the Short-Fat case can be carried forward to these as well. Similar results were obtained for the other choices of r and m, and we omit them from this section for ease of presentation.

5.5.2. Experiments with various ranks. We test the effect of increasing the approximation rank (r) on the different algorithms. First we check the approximation quality of ARk when the r is increased in Figure 5.6(a) on a synthetic matrix $A \in \mathbb{R}_+^{20{,}000 \times 15{,}000}$ with a low rank of 150. We can see that ARk achieves a good approximation even at $r = 10$ and progressively gets better when r is increased. Next we see the effects on running time when r increases. Figure 5.6(b) shows the time per iteration, that is, the time taken to update all columns of U and V once, of the different algorithms as r increases on a matrix with $m = 20{,}000$ and $n = 15{,}000$. All


(a) ARk approximation.

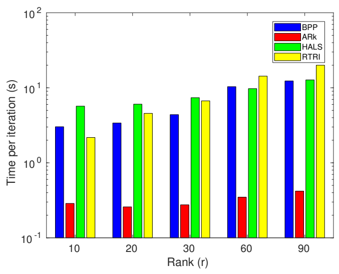
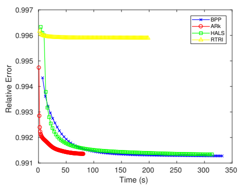
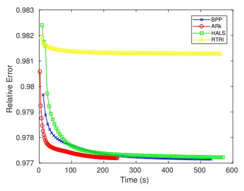
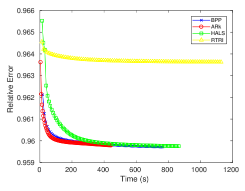

(b) Time per iteration.

FIG. 5.6. *Rank sweep experiments on* $A \in \mathbb{R}_+^{20{,}000\times 15{,}000}$*. ARk achieves better approximations with increased* r *as expected and maintains its computational efficiency over the other methods.*


(a) Sparsity 0.01.


(b) Sparsity 0.05.


(c) Sparsity 0.1.

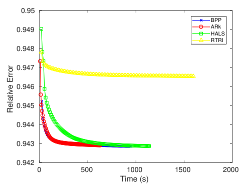

(d) Sparsity 0.15.

FIG. 5.7. *Sparse case:* $A \in \mathbb{R}_+^{20{,}000\times 15{,}000}$ *with varying sparsities. ARk achieves the lowest residual errors in the shortest computational time.*

algorithms show a moderate increase in time as r increases. ARk maintains about 10 times faster computational speed over the other algorithms for all r.

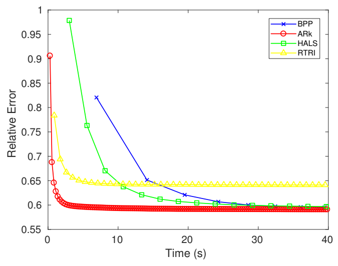
5.5.3. Experiments with sparse synthetic data. The sparse synthetic matrices are created in the following manner. We first generate a dense low-rank nonnegative matrix $L \in \mathbb{R}_+^{m\times n}$ as shown in subsection 5.3. Then we generate a uniform random sparse matrix $X \in \mathbb{R}_+^{m\times n}$ with the desired sparsity ρ and elementwise multiply it with L to obtain our synthetic matrix $A = X * L$. Here $*$ denotes the elementwise product of matrices. It must be noted that this matrix is not truly low-rank due to the elementwise product.

Figure 5.7 shows the performance of our four algorithms on a sparse $20,000 \times 15,000$ matrix with $r = 60$ and varying sparsity. We can see that BPP, ARk, and HALS achieve similar approximation errors while RTRI produces larger residual values. ARk is able to achieve the best relative error within the shortest time. Its relative speedup over the other algorithms is less than the dense case, but it still is faster by a factor of 2–3 compared to the other methods.

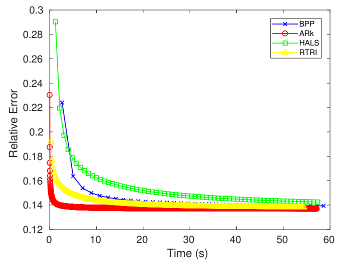
5.6. Experiments on real world data. We run the four algorithms on the real world data described in Table 5.2. We test a suite of approximation rank r, varying from 60 to 150 in increments of 30. In these experiments we want to measure how the algorithms converge over time and use an upper bound on time as the stopping criteria. The maximum time is selected as follows. From Figure 5.5 we can see that generally BPP achieves the lowest residual after 100 iterations. We run BPP for 100 iterations on all the data sets which then select the time needed to reach within 1%, 5%, 10%, or 15% of that final error. This approximation is chosen to keep the overall running time of each algorithm under 15 minutes. For most the data sets we are able to approximate up to 1% of error except for some instances of the larger Caltech and Facescrub runs. The results of these are displayed in Table 5.3. The table contains the

TABLE 5.3
Convergence results on real world data. All methods are run till the maximum time specified and the final relative error is captured.

Data set	Rank	Time (s)	AE (%)	Final relative error			
				BPP	ARk	HALS	RTRI
TDT2	60	31	1	0.7847 ± 0.0006	0.7788 ± 0.0005	0.7841 ± 0.0020	0.8585 ± 0.0012
	90	42	1	0.7548 ± 0.0004	0.7476 ± 0.0004	0.7567 ± 0.0022	0.8584 ± 0.0040
	120	60	1	0.7312 ± 0.0012	0.7247 ± 0.0003	0.7340 ± 0.0009	0.8603 ± 0.0039
	150	84	1	0.7123 ± 0.0007	0.7062 ± 0.0003	0.7161 ± 0.0014	0.8575 ± 0.0023
Reuters	60	24	1	0.6995 ± 0.0016	0.6935 ± 0.0006	0.7006 ± 0.0012	0.7907 ± 0.0025
	90	38	1	0.6693 ± 0.0002	0.6633 ± 0.0003	0.6715 ± 0.0008	0.7939 ± 0.0058
	120	56	1	0.6459 ± 0.0011	0.6402 ± 0.0006	0.6482 ± 0.0015	0.7951 ± 0.0018
	150	72	1	0.6258 ± 0.0015	0.6204 ± 0.0002	0.6293 ± 0.0012	0.7931 ± 0.0037
20Newsgroups	60	40	1	0.5943 ± 0.0017	0.5909 ± 0.0015	0.5965 ± 0.0017	0.6412 ± 0.0054
	90	70	1	0.5626 ± 0.0014	0.5582 ± 0.0010	0.5642 ± 0.0006	0.6385 ± 0.0025
	120	89	1	0.5394 ± 0.0006	0.5349 ± 0.0005	0.5417 ± 0.0011	0.6379 ± 0.0066
	150	108	1	0.5212 ± 0.0017	0.5172 ± 0.0010	0.5236 ± 0.0013	0.6354 ± 0.0049
ORL	60	58	1	0.1393 ± 0.0001	0.1369 ± 0.0001	0.1423 ± 0.0004	0.1381 ± 0.0001
	90	99	1	0.1241 ± 0.0002	0.1212 ± 0.0001	0.1263 ± 0.0003	0.1229 ± 0.0001
	120	162	1	0.1129 ± 0.0001	0.1092 ± 0.0001	0.1139 ± 0.0001	0.1129 ± 0.0004
	150	305	1	0.1038 ± 0.0001	0.0988 ± 0.0001	0.1030 ± 0.0002	0.1062 ± 0.0003
Facescrub	60	454	1	0.1593 ± 0.0001	0.1584 ± 0.0001	0.1630 ± 0.0003	0.2370 ± 0.0080
	90	701	1	0.1435 ± 0.0001	0.1420 ± 0.0000	0.1470 ± 0.0004	0.2319 ± 0.0028
	120	540	5	0.1351 ± 0.0004	0.1308 ± 0.0001	0.1400 ± 0.0003	0.2304 ± 0.0045
	150	746	15	0.1369 ± 0.0005	0.1223 ± 0.0000	0.1317 ± 0.0002	0.2278 ± 0.0036
YaleB	60	70	1	0.1679 ± 0.0003	0.1655 ± 0.0001	0.1751 ± 0.0005	0.1731 ± 0.0010
	90	126	1	0.1464 ± 0.0003	0.1437 ± 0.0000	0.1541 ± 0.0007	0.1515 ± 0.0008
	120	194	1	0.1309 ± 0.0002	0.1283 ± 0.0001	0.1376 ± 0.0004	0.1396 ± 0.0009
	150	364	1	0.1194 ± 0.0002	0.1160 ± 0.0001	0.1245 ± 0.0004	0.1316 ± 0.0008
Caltech256	60	495	1	0.2067 ± 0.0002	0.2054 ± 0.0000	0.2091 ± 0.0004	0.2812 ± 0.0039
	90	763	1	0.1910 ± 0.0001	0.1899 ± 0.0001	0.1939 ± 0.0000	0.2785 ± 0.0041
	120	567	5	0.1845 ± 0.0002	0.1793 ± 0.0001	0.1868 ± 0.0002	0.2756 ± 0.0058
	150	732	15	0.1944 ± 0.0003	0.1708 ± 0.0001	0.1787 ± 0.0002	0.2716 ± 0.0050



(a) 20Newsgroups with $r = 60$.



(b) ORL with $r = 60$.

FIG. 5.8. *Some typical convergence plots for world data. ARk is able to achieve good approximation error in a fraction of the time taken by the other methods.*

final residual of all the algorithms averaged over 5 runs with different initializations. AE is the approximation percentage of the error with respect to running BPP for 100 iterations.

We can observe that ARk is the best performing algorithm followed by BPP, HALS, and RTRI in that order. RTRI performs particularly poorly for the sparse inputs. The main reason for ARk performing well on these experiments is its computational efficiency. ARk is able to perform thousands of iterations in the same time as it takes BPP or HALS to perform tens. This can be clearly seen in Figure 5.8 where we can see that ARk has converged very quickly and before BPP can complete a single iteration. It is true that given enough time the other algorithms, especially BPP, might find a better solution, but this could be prohibitively expensive especially for larger values r. From these experiments we can conclude that ARk strikes a good balance between accuracy and computational efficiency and discovers good approximations much faster than the the other methods compared in this work.

6. Concluding remarks.

In this paper, we have established the recursive formula for the solutions of the rank-k NLS and developed the ARkNLS framework for NMF based on this recursive formula. We have further studied ARkNLS with $k = 3$ which builds upon the rank-3 residue iteration for NLS that updates two more columns than HALS per updating step. We have also introduced a new strategy that efficiently overcomes the potential singularity problem within the context of NMF computation. Extensive numerical comparisons using real data sets demonstrate that our new algorithm ARkNLS(k=3) provides state-of-the-art performance in terms of computational accuracy and CPU time.

Appendix A. Proof of Corollary 3.3.

Proof. Let

$$\widehat{\mathbf{b}} = \mathbf{b} - \frac{\mathbf{g}_3\mathbf{g}_3^T}{\|\mathbf{g}_3\|^2}\mathbf{b}, \quad \widehat{\mathbf{g}}_1 = \mathbf{g}_1 - \frac{\mathbf{g}_3\mathbf{g}_3^T}{\|\mathbf{g}_3\|^2}\mathbf{g}_1, \quad \widehat{\mathbf{g}}_2 = \mathbf{g}_2 - \frac{\mathbf{g}_3\mathbf{g}_3^T}{\|\mathbf{g}_3\|^2}\mathbf{g}_2.$$

A simple calculation yields that

$$\text{(A.1)} \quad \widehat{\mathbf{g}}_2^T\widehat{\mathbf{g}}_1 = \left(\mathbf{g}_2 - \frac{\mathbf{g}_3^T\mathbf{g}_2}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right)^T\left(\mathbf{g}_1 - \frac{\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right) = \frac{(\mathbf{g}_2^T\mathbf{g}_1)\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_2)(\mathbf{g}_3^T\mathbf{g}_1)}{\|\mathbf{g}_3\|^2},$$

$$\text{(A.2)} \quad \|\widehat{\mathbf{g}}_1\|^2 = \left(\mathbf{g}_1 - \frac{\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right)^T\left(\mathbf{g}_1 - \frac{\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right) = \frac{\|\mathbf{g}_1\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_1)^2}{\|\mathbf{g}_3\|^2},$$

$$\text{(A.3)}\quad \|\widehat{\mathbf{g}}_2\|^2 = \left(\mathbf{g}_2 - \frac{\mathbf{g}_3^T\mathbf{g}_2}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right)^T\left(\mathbf{g}_2 - \frac{\mathbf{g}_3^T\mathbf{g}_2}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right) = \frac{\|\mathbf{g}_2\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_2)^2}{\|\mathbf{g}_3\|^2},$$

$$\text{(A.4)}\quad \widehat{\mathbf{b}}^T\widehat{\mathbf{g}}_2 = \left(\mathbf{b} - \frac{\mathbf{g}_3^T\mathbf{b}}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right)^T\left(\mathbf{g}_2 - \frac{\mathbf{g}_3^T\mathbf{g}_2}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right) = \frac{(\mathbf{b}^T\mathbf{g}_2)\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_2)(\mathbf{g}_3^T\mathbf{b})}{\|\mathbf{g}_3\|^2},$$

$$\text{(A.5)}\quad \widehat{\mathbf{b}}^T\widehat{\mathbf{g}}_1 = \left(\mathbf{b} - \frac{\mathbf{g}_3^T\mathbf{b}}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right)^T\left(\mathbf{g}_1 - \frac{\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_3\|^2}\mathbf{g}_3\right) = \frac{(\mathbf{b}^T\mathbf{g}_1)\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_1)(\mathbf{g}_3^T\mathbf{b})}{\|\mathbf{g}_3\|^2}.$$

Let

$$s(\begin{bmatrix}\widehat{\mathbf{g}}_1 & \widehat{\mathbf{g}}_2\end{bmatrix}, \widehat{\mathbf{b}}) = \begin{bmatrix} s_1(\begin{bmatrix}\widehat{\mathbf{g}}_1 & \widehat{\mathbf{g}}_2\end{bmatrix}, \widehat{\mathbf{b}}) \\ s_2(\begin{bmatrix}\widehat{\mathbf{g}}_1 & \widehat{\mathbf{g}}_2\end{bmatrix}, \widehat{\mathbf{b}}) \end{bmatrix}$$

be the solution to the optimization problem

$$\min_{y_1, y_2 \geq 0} \|y_1\widehat{\mathbf{g}}_1 + y_2\widehat{\mathbf{g}}_2 - \widehat{\mathbf{b}}\|.$$

Then from (A.1)–(A.5) and Corollary 3.2, we have

$$\begin{aligned}
&s_2\left(\begin{bmatrix}\widehat{\mathbf{g}}_1 & \widehat{\mathbf{g}}_2\end{bmatrix}, \widehat{\mathbf{b}}\right) \\
&\quad = \frac{1}{\|\widehat{\mathbf{g}}_2\|^2}\left[\widehat{\mathbf{b}}^T\widehat{\mathbf{g}}_2 - \widehat{\mathbf{g}}_2^T\widehat{\mathbf{g}}_1\left[\frac{\|\widehat{\mathbf{g}}_2\|^2\widehat{\mathbf{b}}^T\widehat{\mathbf{g}}_1 - \widehat{\mathbf{b}}^T\widehat{\mathbf{g}}_2\cdot\widehat{\mathbf{g}}_2^T\widehat{\mathbf{g}}_1}{\|\widehat{\mathbf{g}}_1\|^2\|\widehat{\mathbf{g}}_2\|^2 - (\widehat{\mathbf{g}}_1^T\widehat{\mathbf{g}}_2)^2}\right]_+\right]_+ \\
&\quad = \left[\frac{\mathbf{b}^T\mathbf{g}_2\cdot\|\mathbf{g}_3\|^2 - \mathbf{b}^T\mathbf{g}_3\cdot\mathbf{g}_3^T\mathbf{g}_2}{\|\mathbf{g}_2\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_2)^2} - \frac{\mathbf{g}_2^T\mathbf{g}_1\cdot\|\mathbf{g}_3\|^2 - \mathbf{g}_3^T\mathbf{g}_2\cdot\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_2\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_2)^2}\left[\frac{\det([\mathbf{b}, \mathbf{g}_2, \mathbf{g}_3]^T G)}{\det(G^TG)}\right]_+\right]_+ \\
&\quad = p.
\end{aligned}$$

Moreover,

$$\begin{aligned}
&s_1\left(\begin{bmatrix}\widehat{\mathbf{g}}_1 & \widehat{\mathbf{g}}_2\end{bmatrix}, \widehat{\mathbf{b}}\right) \\
&\quad = \left[\frac{\mathbf{b}^T\mathbf{g}_1\cdot\|\mathbf{g}_3\|^2 - \mathbf{b}^T\mathbf{g}_3\cdot\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_1\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_1)^2} - \frac{\mathbf{g}_2^T\mathbf{g}_1\cdot\|\mathbf{g}_3\|^2 - \mathbf{g}_3^T\mathbf{g}_2\cdot\mathbf{g}_3^T\mathbf{g}_1}{\|\mathbf{g}_1\|^2\|\mathbf{g}_3\|^2 - (\mathbf{g}_3^T\mathbf{g}_1)^2}s_2(\begin{bmatrix}\widehat{\mathbf{g}}_1 & \widehat{\mathbf{g}}_2\end{bmatrix}, \widehat{\mathbf{b}})\right]_+ \\
&\quad = \widetilde{p}.
\end{aligned}$$

Therefore, according to Theorem 3.1 and Corollary 3.2, the solution to the optimization problem

$$\min_{\{y_1, y_2, y_3\}\geq 0} \|y_1\mathbf{g}_1 + y_2\mathbf{g}_2 + y_3\mathbf{g}_3 - \mathbf{b}\|$$

is

$$\begin{aligned}
y_3^* &= \frac{1}{\|\mathbf{g}_3\|^2}\left[\mathbf{g}_3^T\left(\mathbf{b} - [\mathbf{g}_1, \mathbf{g}_2]\cdot s(\begin{bmatrix}\widehat{\mathbf{g}}_1 & \widehat{\mathbf{g}}_2\end{bmatrix}, \widehat{\mathbf{b}})\right)\right]_+ \\
&= \frac{1}{\|\mathbf{g}_3\|^2}\left[\mathbf{g}_3^T\mathbf{b} - \mathbf{g}_3^T\mathbf{g}_1\cdot\widetilde{p} - \mathbf{g}_3^T\mathbf{g}_2\cdot p\right]_+
\end{aligned}$$

with $s_1([\ \widehat{\mathbf{g}}_1 \quad \widehat{\mathbf{g}}_2\ ], \widehat{\mathbf{b}}) = \widetilde{p}$, $s_2([\ \widehat{\mathbf{g}}_1 \quad \widehat{\mathbf{g}}_2\ ], \widehat{\mathbf{b}}) = p$, and

$$\begin{cases}
y_2^* &= s_2\left(\begin{bmatrix}\mathbf{g}_1 & \mathbf{g}_2\end{bmatrix}, \mathbf{b} - \mathbf{g}_3y_3^*\right) \\
&= \frac{1}{\|\mathbf{g}_2\|^2}\left[(\mathbf{b} - \mathbf{g}_3y_3^*)^T\mathbf{g}_2 - \mathbf{g}_2^T\mathbf{g}_1\left[\frac{\|\mathbf{g}_2\|^2(\mathbf{b}-\mathbf{g}_3y_3^*)^T\mathbf{g}_1 - (\mathbf{b}-\mathbf{g}_3y_3^*)^T\mathbf{g}_2\cdot\mathbf{g}_2^T\mathbf{g}_1}{\|\mathbf{g}_1\|^2\|\mathbf{g}_2\|^2 - (\mathbf{g}_2^T\mathbf{g}_1)^2}\right]_+\right]_+ \\
&= \frac{1}{\|\mathbf{g}_2\|^2}\left[(\mathbf{b} - \mathbf{g}_3y_3^*)^T\mathbf{g}_2 - \mathbf{g}_2^T\mathbf{g}_1\left[\frac{(\mathbf{b}^T\mathbf{g}_1\|\mathbf{g}_2\|^2 - \mathbf{b}^T\mathbf{g}_2\cdot\mathbf{g}_2^T\mathbf{g}_1) - (\mathbf{g}_3^T\mathbf{g}_1\|\mathbf{g}_2\|^2 - \mathbf{g}_3^T\mathbf{g}_2\cdot\mathbf{g}_2^T\mathbf{g}_1)y_3^*}{\|\mathbf{g}_1\|^2\|\mathbf{g}_2\|^2 - (\mathbf{g}_2^T\mathbf{g}_1)^2}\right]_+\right]_+ \\
y_1^* &= s_1\left(\begin{bmatrix}\mathbf{g}_1 & \mathbf{g}_2\end{bmatrix}, \mathbf{b} - \mathbf{g}_3y_3^*\right) = \frac{1}{\|\mathbf{g}_1\|^2}[(\mathbf{b} - \mathbf{g}_3y_3^*)^T\mathbf{g}_1 - (\mathbf{g}_2^T\mathbf{g}_1)y_2^*]_+.
\end{cases}$$

Hence, Corollary 3.3 is proved. □

REFERENCES

[1] A. M. S. Ang and N. Gillis, *Accelerating nonnegative matrix factorization algorithms using extrapolation*, Neural Comput., 31 (2019), pp. 417–439, https://doi.org/10.1162/neco_a_01157.

[2] T. Bouwmans, A. Sobral, S. Javed, S. K. Jung, and E.-H. Zahzah, *Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset*, Comput. Sci. Rev., 23 (2017), pp. 1–71, https://doi.org/10.1016/j.cosrev.2016.11.001.

[3] R. Bro and S. De Jong, *A fast non-negativity-constrained least squares algorithm*, J. Chemometrics, 11 (1997), pp. 393–401, https://doi.org/10.1002/(SICI)1099-128X(199709/10)11:5⟨393::AID-CEM483⟩3.0.CO;2-L.

[4] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, *Metagenes and molecular pattern discovery using matrix factorization*, Proc. Natl. Acad. Sci. USA, 101 (2004), pp. 4164–4169, https://doi.org/10.1073/pnas.0308531101.

[5] I. Buciu, *Non-negative matrix factorization, a new tool for feature extraction: Theory and applications*, Internat. J. Comput. Commun. Control, 3 (2008), pp. 67–74.

[6] Y. T. Chow, T. Wu, and W. Yin, *Cyclic coordinate-update algorithms for fixed-point problems: Analysis and applications*, SIAM J. Sci. Comput., 39 (2017), pp. A1280–A1300, https://doi.org/10.1137/16M1102653.

[7] A. Cichocki and A.-H. Phan, *Fast local algorithms for large scale nonnegative matrix and tensor factorizations*, IEICE Trans. Fund. Electron. Commun. Comput. Sci., 92 (2009), pp. 708–721, https://doi.org/10.1587/transfun.E92.A.708.

[8] A. Cichocki, R. Zdunek, and S.-I. Amari, *Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization*, in Proceedings of the International Conference on Independent Component Analysis and Signal Separation, Springer, 2007, pp. 169–176, https://doi.org/10.1007/978-3-540-74494-8_22.

[9] A. Cichocki, R. Zdunek, A. H. Phan, and S.-I. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*, John Wiley & Sons, New York, 2009, https://doi.org/10.1002/9780470747278.

[10] G. Cui, X. Li, and Y. Dong, *Subspace clustering guided convex nonnegative matrix factorization*, Neurocomputing, 292 (2018), pp. 38–48, https://doi.org/10.1016/j.neucom.2018.02.067.

[11] K. Devarajan, *Nonnegative matrix factorization: An analytical and interpretive tool in computational biology*, PLoS Comput. Biol., 4 (2008), e1000029, https://doi.org/10.1371/journal.pcbi.1000029.

[12] B. Dong, M. M. Lin, and H. Park, *Integer matrix approximation and data mining*, J. Sci. Comput., 75 (2018), pp. 198–224, https://doi.org/10.1007/s10915-017-0531-7.

[13] B. Drake, T. Huang, A. Beavers, R. Du, and H. Park, *Event detection based on nonnegative matrix factorization: Ceasefire violation, environmental, and malware events*, in Proceedings of the International Conference on Applied Human Factors and Ergonomics, Springer, 2017, pp. 158–169, https://doi.org/10.1007/978-3-319-60585-2_16.

[14] R. Du, B. Drake, and H. Park, *Hybrid clustering based on content and connection structure using joint nonnegative matrix factorization*, J. Global Optim., 74 (2019), pp. 861–877, https://doi.org/10.1007/s10898-017-0578-x.

[15] R. Du, D. Kuang, B. Drake, and H. Park, *DC-NMF: Nonnegative matrix factorization based on divide-and-conquer for fast clustering and topic modeling*, J. Global Optim., 68 (2017), pp. 777–798, https://doi.org/10.1007/s10898-017-0515-z.

[16] R. Du, D. Kuang, B. Drake, and H. Park, *Hierarchical community detection via rank-2 symmetric nonnegative matrix factorization*, Comput. Soc. Netw., 4 (2017), p. 7, https://doi.org/10.1186/s40649-017-0043-5.

[17] N. B. Erichson, A. Mendible, S. Wihlborn, and J. N. Kutz, *Randomized nonnegative matrix factorization*, Pattern Recogn. Lett., 104 (2018), pp. 1–7, https://doi.org/10.1016/j.patrec.2018.01.007.

[18] R. Fujimoto, A. Guin, M. Hunter, H. Park, G. Kanitkar, R. Kannan, M. Milholen, S. Neal, and P. Pecher, *A dynamic data driven application system for vehicle tracking*, Proc. Comput. Sci., 29 (2014), pp. 1203–1215, https://doi.org/10.1016/j.procs.2014.05.108.

[19] N. Gillis, D. Kuang, and H. Park, *Hierarchical clustering of hyperspectral images using rank-two nonnegative matrix factorization*, IEEE Trans. Geosci. Remote Sensing, 53 (2014), pp. 2066–2078, https://doi.org/10.1109/TGRS.2014.2352857.

[20] Z.-Q. He and X. Yuan, *Block iteratively reweighted algorithms for robust symmetric nonnegative matrix factorization*, IEEE Signal Process. Lett., 25 (2018), pp. 1510–1514, https://doi.org/10.1109/LSP.2018.2865857.

[21] N. Ho, *Nonnegative Matrix Factorization Algorithms and Applications*, Ph.D. thesis, Univ. Catholique de Louvain, 2008.

[22] P. O. Hoyer, *Non-negative matrix factorization with sparseness constraints*, J. Mach. Learn. Res., 5 (2004), pp. 1457–1469.

[23] D. Kim, S. Sra, and I. S. Dhillon, *Fast Newton-type methods for the least squares nonnegative matrix approximation problem*, in Proceedings of the 2007 SIAM International Conference on Data Mining, SIAM, Philadelphia, PA, 2007, pp. 343–354, https://doi.org/10.1137/1.9781611972771.31.

[24] H. Kim and H. Park, *Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis*, Bioinformatics, 23 (2007), pp 1495–1502, https://doi.org/10.1093/bioinformatics/btm134.

[25] H. Kim and H. Park, *Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 713–730, https://doi.org/10.1137/07069239X.

[26] J. Kim, *Nonnegative Matrix and Tensor Factorizations, Least Squares Problems, and Applications*, Ph.D. thesis, Georgia Institute of Technology, 2011.

[27] J. Kim, Y. He, and H. Park, *Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework*, J. Global Optim., 58 (2014), pp. 285–319, https://doi.org/10.1007/s10898-013-0035-4.

[28] J. Kim and H. Park, *Toward faster nonnegative matrix factorization: A new algorithm and comparisons*, in Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 353–362, https://doi.org/10.1109/ICDM.2008.149.

[29] J. Kim and H. Park, *Fast nonnegative matrix factorization: An active-set-like method and comparisons*, SIAM J. Sci. Comput., 33 (2011), pp. 3261–3281, https://doi.org/10.1137/110821172.

[30] D. Kuang, C. Ding, and H. Park, *Symmetric nonnegative matrix factorization for graph clustering*, in Proceedings of the 2012 SIAM International Conference on Data Mining, SIAM, Philadelphia, PA, 2012, pp. 106–117, https://doi.org/10.1137/1.9781611972825.10.

[31] D. Kuang and H. Park, *Fast rank-*2 *nonnegative matrix factorization for hierarchical document clustering*, in Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2013, pp. 739–747, https://doi.org/10.1145/2487575.2487606.

[32] D. Kuang, S. Yun, and H. Park, *SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering*, J. Global Optim., 62 (2015), pp. 545–574, https://doi.org/10.1007/s10898-014-0247-2.

[33] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, SIAM, Philadelphia, PA, 1995, https://doi.org/10.1137/1.9781611971217.

[34] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791, https://doi.org/10.1038/44565.

[35] Z. Li, X. Wu, and H. Peng, *Nonnegative matrix factorization on orthogonal subspace*, Pattern Recogn. Lett., 31 (2010), pp. 905–911, https://doi.org/10.1016/j.patrec.2009.12.023.

[36] C.-J. Lin, *Projected gradient methods for nonnegative matrix factorization*, Neural Comput., 19 (2007), pp. 2756–2779, https://doi.org/10.1162/neco.2007.19.10.2756.

[37] H. Liu and Y. Zhou, *Rank-two residue iteration method for nonnegative matrix factorization*, Neurocomputing, 74 (2011), pp. 3305–3312, https://doi.org/10.1016/j.neucom.2011.05.011.

[38] P. Luo, J. Peng, Z. Guan, and J. Fan, *Dual regularized multi-view non-negative matrix factorization for clustering*, Neurocomputing, 294 (2018), pp. 1–11, https://doi.org/10.1016/j.neucom.2017.10.023.

[39] M. Merritt and Y. Zhang, *Interior-point gradient method for large-scale totally nonnegative least squares problems*, J. Optim. Theory Appl., 126 (2005), pp. 191–202, https://doi.org/10.1007/s10957-005-2668-z.

[40] H.-W. Ng and S. Winkler, *A data-driven approach to cleaning large face datasets*, in Proceedings of the 2014 IEEE International Conference on Image Processing (ICIP), IEEE, 2014, pp. 343–347, https://doi.org/10.1109/ICIP.2014.7025068.

[41] P. Paatero and U. Tapper, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, Environmetrics, 5 (1994), pp. 111–126, https://doi.org/10.1002/env.3170050203.

[42] V. P. Pauca, J. Piper, and R. J. Plemmons, *Nonnegative matrix factorization for spectral data analysis*, Linear Algebra Appl., 416 (2006), pp. 29–47, https://doi.org/10.1016/j.laa.2005.06.025.

[43] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons, *Text mining using non-negative matrix factorizations*, in Proceedings of the 2004 SIAM International Conference

on Data Mining, SIAM, Philadelphia, PA, 2004, pp. 452–456, https://doi.org/10.1137/1.9781611972740.45.

[44] X. Peng, D. Chen, and D. Xu, *Semi-supervised least squares nonnegative matrix factorization and graph-based extension*, Neurocomputing, 320 (2018), pp. 98–111, https://doi.org/10.1016/j.neucom.2018.09.026.

[45] A. Sapienza, A. Bessi, and E. Ferrara, *Non-negative tensor factorization for human behavioral pattern mining in online games*, Information, 9 (2018), p. 66, https://doi.org/10.3390/info9030066.

[46] J. Sun, Z. Wang, F. Sun, and H. Li, *Sparse dual graph-regularized nmf for image co-clustering*, Neurocomputing, 316 (2018), pp. 156–165, https://doi.org/10.1016/j.neucom.2018.07.062.

[47] A. Tosyali, J. Kim, J. Choi, and M. K. Jeong, *Regularized asymmetric nonnegative matrix factorization for clustering in directed networks*, Pattern Recogn. Lett., 125 (2019), pp. 750–757, https://doi.org/10.1016/j.patrec.2019.07.005.

[48] M. H. Van Benthem and M. R. Keenan, *Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems*, J. Chemometrics, 18 (2004), pp. 441–450, https://doi.org/10.1002/cem.889.

[49] C. Wang, X. Song, and J. Zhang, *Graph regularized nonnegative matrix factorization with sample diversity for image representation*, Engrg. Appl. Artif. Intell., 68 (2018), pp. 32–39, https://doi.org/10.1016/j.engappai.2017.10.018.

[50] Y.-X. Wang and Y.-J. Zhang, *Nonnegative matrix factorization: A comprehensive review*, IEEE Trans. Knowledge Data Engrg., 25 (2012), pp. 1336–1353, https://doi.org/10.1109/TKDE.2012.51.

[51] B. Wu, E. Wang, Z. Zhu, W. Chen, and P. Xiao, *Manifold NMF with l_{21} norm for clustering*, Neurocomputing, 273 (2018), pp. 78–88, https://doi.org/10.1016/j.neucom.2017.08.025.

[52] W. Xu, X. Liu, and Y. Gong, *Document clustering based on non-negative matrix factorization*, in Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2003, pp. 267–273, https://doi.org/10.1145/860435.860485.

[53] R. Zdunek and A. Cichocki, *Non-negative matrix factorization with quasi-Newton optimization*, in Proceedings of the International Conference on Artificial Intelligence and Soft Computing, Springer, 2006, pp. 870–879, https://doi.org/10.1007/11785231_91.

[54] R. Zdunek and A. Cichocki, *Fast nonnegative matrix factorization algorithms using projected gradient approaches for large-scale problems*, Comput. Intell. Neurosci., 2008 (2008), https://doi.org/10.1155/2008/939567.

[55] F. Zhu and P. Honeine, *Online kernel nonnegative matrix factorization*, Signal Process., 131 (2017), pp. 143–153, https://doi.org/10.1016/j.sigpro.2016.08.011.