symmetry

MDPI

Article

Approximation Conjugate Gradient Method for Low-Rank Matrix Recovery

Zhilong Chen [1], Peng Wang [1,2,3,*] and Detong Zhu [4]

[1] Mathematics and Statistics College, Hainan Normal University, Haikou 570203, China; 202312070100004@hainnu.edu.cn
[2] Key Laboratory of Data Science and Intelligence Education of Ministry of Education, Hainan Normal University, Haikou 570203, China
[3] Key Laboratory of Computational Science and Application of Hainan Province, Haikou 570203, China
[4] Mathematics and Science College, Shanghai Normal University, Shanghai 200234, China; dtzhu@shnu.edu.cn
* Correspondence: pengwang621@163.com or 050115@hainnu.edu.cn

Abstract: Large-scale symmetric and asymmetric matrices have emerged in predicting the relationship between genes and diseases. The emergence of large-scale matrices increases the computational complexity of the problem. Therefore, using low-rank matrices instead of original symmetric and asymmetric matrices can greatly reduce computational complexity. In this paper, we propose an approximation conjugate gradient method for solving the low-rank matrix recovery problem, i.e., the low-rank matrix is obtained to replace the original symmetric and asymmetric matrices such that the approximation error is the smallest. The conjugate gradient search direction is given through matrix addition and matrix multiplication. The new conjugate gradient update parameter is given by the F-norm of matrix and the trace inner product of matrices. The conjugate gradient generated by the algorithm avoids SVD decomposition. The backtracking linear search is used so that the approximation conjugate gradient direction is computed only once, which ensures that the objective function decreases monotonically. The global convergence and local superlinear convergence of the algorithm are given. The numerical results are reported and show the effectiveness of the algorithm.

Keywords: approximation conjugate gradient method; low-rank matrix recovery; backtracking linear search technique; global convergence; superlinear convergence

MSC: 49M37; 65K05; 90C30; 90C56

Citation: Chen, Z.; Wang, P.; Zhu, D. Approximation Conjugate Gradient Method for Low-Rank Matrix Recovery. *Symmetry* **2024**, *16*, 547. https://doi.org/10.3390/sym16050547

Academic Editors: Boris Malomed, Hidetsugu Sakaguchi, David Laroze, Qing-Wen Wang and Calogero Vetro

Received: 22 March 2024
Revised: 21 April 2024
Accepted: 24 April 2024
Published: 2 May 2024

1. Introduction

Problem Description Motivation

The low-rank matrix plays an important role in a broad range of applications and multiple scientific fields. For example, many bioinformatics experts use the matrix restoration technique to predict the relationship between diseases and genes [1]. In many bioinformatics problems, the information between genes and diseases is presented in the form of a symmetric or asymmetric matrix, but the dimensions of symmetric and asymmetric matrices are extremely large, which makes it difficult to calculate effectively in practical situations and even leads to the inability to perform calculations. Hence, it is often possible to recover it due to the low-rank structure. In many cases, the rank r of the low-rank matrix is fixed and the matrix decomposition $M = UV^T$ is completed, where $M \in \mathbb{R}^{m\times n}$ is a known matrix and $U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}$ is the variable matrices of the following problem:

$$\min_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{n\times r}} F(U,V) = \frac{1}{2}\|UV^T - M\|_F^2. \tag{1}$$

In many cases, matrix M is a symmetric matrix, i.e., $M \in \mathbb{R}^{n\times n}$ and the matrix decomposition $M = UU^T$, $U \in \mathbb{R}^{n\times r}$, problem (1) is equivalent to solving the following low-rank symmetric matrix recovery problem:

$$\min_{U\in\mathbb{R}^{n\times r}} F(U,U) = \frac{1}{2}\|UU^T - M\|_F^2. \tag{2}$$

As the factorized problem (1) or (2) involves only $(m+n)r$ or $2nr$ variables, these methods are, in general, more scalable and can cope with larger matrices. The solution to the matrix recovery problem has also attracted the interest of many scholars.

On the Euclidean manifold, Sun and Luo [2] established a theoretical guarantee for the factorization-based formulation to correctly recover the underlying low-rank matrix. Tu et al. [3] proposed a Procrustes Flow algorithm for solving the problem of recovering a low-rank matrix from linear measurements.

On other Riemannian manifolds, Keshavan et al. [4] introduced an efficient algorithm such that the method could reconstruct matrix $M \in \mathbb{R}^{m\alpha\times n}$ from $O(rn)$. Ngo and Saad [5] described gradient methods based on a scaled metric for low-rank matrix completion. Vandereycken [6] proposed a new algorithm for matrix completion. Mishra and Sepulchre [7] introduced a nonlinear conjugate gradient method for solving low-rank matrix completion. Mishra et al. [8] proposed a gradient descent and trust-region algorithm for low-rank matrix completion. Boumal and Absil [9] exploited the geometry of the low-rank constraint to recast the problem as an unconstrained optimization problem on a single Grassmann manifold. Wei et al. [10] introduced a family of Riemannian optimization algorithms for low-rank matrix recovery problems. In the past year, Najafi and Hajarian [11] proposed an improved Riemannian conjugate gradient method to solve robust matrix completion problems. Duan et al. [12] proposed the Riemannian conjugate gradient method for solving the low-rank tensor completion problem.

In addition to the aforementioned work, there are also some algorithms for solving matrix completion problems (see [13–18]). For solving matrix recovery problems, some algorithms and related research have also been widely studied (see [19–24]).

Inspired by the algorithm ideas in [25,26], this paper proposes an approximation conjugate gradient method to solve the low-rank matrix recovery problem. The approximation conjugate gradient direction is given such that the search direction is the descent direction of the objective $F(U,V)$. The algorithm avoids the singular value decomposition of the matrix and only obtains the search direction of the problem through the multiplication of the matrix. The backtracking line search technique ensures that the objective function of problem (1) is monotonic descent.

In this paper, the approximation conjugate gradient direction is introduced in Section 2. In Section 3, the detailed steps of the approximation conjugate gradient algorithm are proposed. The global convergence of the algorithm is given in Section 4. In Section 5, the local superlinear convergence of the algorithm is given. Numerical results are presented in Section 6, and in Section 7, some conclusions are given.

Notation

In this paper, the low-rank matrix recovery problem is solved by the approximation Newton algorithm. In problem (1), let $Z \in \mathbb{R}^{m\times n}$ denote the variables, and Z^* be the optimal solution of objective function $f(Z)$. The F-norm of matrix Z is defined as $\|Z\|_F^2 = \sum_{i=1,j=1}^{m,n} Z_{ij}^2$. A symmetric and positive semidefinite matrix $B - A$ is represented as $A \preceq B$. Given any matrix $A, B \in \mathbb{R}^{m\times n}$, $A \bullet B$ denotes the usual trace inner product of A and B.

2. The Approximation Conjugate Gradient Step

In this paper, we propose a new approximation conjugate gradient algorithm for solving problem (1). Next, we first introduce the traditional conjugate gradient algorithm for solving the unconstrained nonlinear programming problem.

In the Euclidean case, the conjugate gradient methods are line search algorithms to solve the unconstrained nonlinear programming problem, where the objective function is $f : \mathbb{R}^n \to \mathbb{R}$. The sequence $\{x_k\}$ in $\mathbb{R}^n$ is generated by

$$x_{k+1} = x_k + \alpha_k \hat{\eta}_k,$$

with $\alpha_k > 0$ for all $k \geq 0$ from an initial point $x_0 \in \mathbb{R}^n$, and the search directions $\hat{\eta}_k \in \mathbb{R}^n$ are computed using the gradient $\hat{g}_k = \nabla f(x_k)$ as $\hat{\eta}_0 = -\hat{g}_0$ and

$$\hat{\eta}_{k+1} = -\hat{g}_{k+1} + \beta_{k+1}\hat{\eta}_k = -\nabla f(x_{k+1}) + \beta_{k+1}\hat{\eta}_k$$

for all $k \geq 0$. The computation of real values β_{k+1} is crucial for the performance of conjugate gradient methods. A famous real value of β_{k+1} is

$$\text{Conjugatedescent} : \beta_{k+1}^{CD} = \frac{\|\hat{g}_{k+1}\|_2^2}{-\hat{g}_k^T \hat{\eta}_k}.$$

It was proposed by Fletcher [27].

To solve problem (1), we let

$$\begin{aligned} \nabla F_U(Z) =& \nabla F_V(U, V) = (UV^T - M)V, \\ \nabla F_V(Z) =& \nabla F_U(U, V) = (UV^T - M)^T U. \end{aligned}$$

Hence,

$$\nabla F(Z) = \nabla F(U, V) = \begin{bmatrix} \nabla F_U(U, V) \\ \nabla F_V(U, V) \end{bmatrix} = \begin{bmatrix} (UV^T - M)V \\ (UV^T - M)^T U \end{bmatrix},$$

and

$$g_k = \nabla F(Z_k) = \nabla F(U_k, V_k) = \begin{bmatrix} \nabla F_U(U_k, V_k) \\ \nabla F_V(U_k, V_k) \end{bmatrix} = \begin{bmatrix} (U_k V_k^T - M)V_k \\ (U_k V_k^T - M)^T U_k \end{bmatrix},$$

on iteration point $Z_k = \begin{bmatrix} U_k \\ V_k \end{bmatrix}$.

The sequence $\{Z_k\}$ is generated by

$$Z_{k+1} = Z_k + \alpha_k \eta_k,$$

from an initial point $Z_0 \in \mathbb{R}^{(m+n)\times r}$ with $\alpha_k > 0$ for all $k \geq 0$. The search direction $\eta_k \in \mathbb{R}^{(m+n)\times r}$ are computed using the gradient $g_k = \nabla F(Z_k)$ as $\eta_0 = -g_0 = -\nabla F(Z_0)$ and

$$\begin{aligned} \eta_{k+1} =& -g_{k+1} + \beta_{k+1}\eta_k \\ =& -\nabla F(Z_{k+1}) + \beta_{k+1}\eta_k, \end{aligned} \tag{3}$$

for all $k \geq 0$. To obtain better search directions, we constructed the following real value β_{k+1} of Algorithm 1 based on the structure of the problem itself,

$$\beta_{k+1} = \begin{cases} \frac{\|\nabla F(Z_{k+1})\|_F^2}{-\nabla F(Z_{k+1}) \bullet \eta_k}, & \text{if} |\nabla F(Z_{k+1}) \bullet \eta_k| > \frac{\epsilon^2}{2}, \\ \frac{\|\nabla F(Z_{k+1})\|_F^2}{2\epsilon^2}, & \text{otherwise}, \end{cases} \tag{4}$$

where $\nabla F(Z_{k+1}) \bullet \eta_k$ denotes the usual trace inner product of $\nabla F(Z_{k+1}), \eta_k \in \mathbb{R}^{(m+n)\times r}$ and $\epsilon \in (0,1)$.

Algorithm 1 (ACGA)

Input: Choose an initial $Z_0 = \begin{bmatrix} U_0 \\ V_0 \end{bmatrix}$, where $U_0 \in \mathbb{R}^{m\times r}$, $V_0 \in \mathbb{R}^{n\times r}$. Let $0 < \beta < 1, \epsilon > 0$ and set $k = 0$.

Main step:

1. If $k = 0$, then compute $\nabla F(Z_k)$ and let $\eta_k = -\nabla F(Z_k)$.
2. If $k = 0$ and $\|\nabla F(Z_k)\|_F \leq \epsilon$, then stop, Z_k is the optimal solution of problem (1).
3. Compute $\alpha_k \in (0,1]$ such that

$$F(Z_k + \alpha_k \eta_k) \leq F(Z_k) + \alpha_k \beta \nabla F(Z_k) \bullet \eta_k. \tag{5}$$

4. Let $Z_{k+1} = Z_k + \alpha_k \eta_k$.
5. Calculate $\nabla F(Z_{k+1})$, if $\|\nabla F(Z_{k+1})\|_F \leq \epsilon$, then stop, Z_{k+1} is the optimal solution of problem (1).
6. Compute β_{k+1} as (2) and

$$\eta_{k+1} = -\nabla F(Z_{k+1}) + \beta_{k+1}\eta_k,$$

let $k = k + 1$, go to 3.

3. The Approximation Conjugate Gradient Algorithm (ACGA)

In Section 2, we introduce the approximation conjugate gradient step with updated real value β_k. .

Remark 1. *We obtain the step size α_k by the backtracking line search based on objective function $F(Z)$, i.e., given $\omega \in (0,1)$, let $\alpha_k = 1, \omega, \omega^2, \cdots$ until (5) holds.*

Remark 2. *We can randomly generate initial $m \times r$ matrix U_0 and initial $n \times r$ matrix V_0 with rank r. We can generate the matrices $U_0 \in \mathbb{R}^{m\times r}$ and $V_0 \in \mathbb{R}^{n\times r}$ as follow,*

$$U_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}_{m\times r} \text{ and } V_0 = \begin{bmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \end{bmatrix}_{n\times r}.$$

4. Convergence Analysis

In this section, to prove the convergence of the algorithm (ACGA), we require the model to satisfy the following assumption:

Assumption 1. *Assume that the objective functions of problem (1) are twice continuous differentiable, i.e., for any $F(Z)$, it is twice continuous differentiable. For any point $Z \in \mathbb{R}^{(m+n)\times r}$ in our method, we define the level set*

$$L(Z_0) = \{Z \in \mathbb{R}^{(m+n)\times r}, F(Z) \leq F(Z_0)\}.$$

Assume the level set $L(Z_0)$ is bounded.

Assumption 2. *Assume the gradient of $F(Z)$ is Lipschitz-continuous, i.e., there exists $\kappa_H > 0$ such that*

$$\|\nabla F(X) - \nabla F(Y)\|_F \leq \kappa_H \|X - Y\|_F,$$

for any $X, Y \in \mathbb{R}^{m\times n}$.

The bounded function $F(Z)$ is important for the proof of convergence of Algorithm 1 and the following lemma presents the boundedness of the objective function $F(Z)$.

Lemma 1. *Under Assumption 1, there exists constant $\kappa_f > 0$ and $\kappa_g > 0$ such that*

$$\begin{aligned} \|F(Z_k)\|_F \leq & \kappa_f, \\ \|\nabla F(Z_k)\|_F \leq & \kappa_g. \end{aligned}$$

Proof. The proof is similar to Lemma 3.2 in [28]. □

The purpose of step 3 of Algorithm 1 is to monotonically decrease the objective function of problem (1). Hence, we need to prove that $\nabla F(Z_k) \bullet \eta_k < 0$ when Algorithm 1 does not terminate.

Lemma 2. *If Algorithm 1 dose not terminate, i.e., for constant $\epsilon > 0$, we have $\|\nabla F(Z_k)\|_F > \epsilon$, then*

$$\nabla F(Z_k) \bullet \eta_k < 0.$$

Proof. If Algorithm 1 dose not terminate, i.e., $\|\nabla F(Z_k)\| > \epsilon$, then according to the definition of η_k, we have that

$$\begin{aligned} \nabla F(Z_k) \bullet \eta_k = & \nabla F(Z_k) \bullet (-\nabla F(Z_k) + \beta_k \eta_{k-1}) \\ = & -\nabla F(Z_k) \bullet \nabla F(Z_k) + \beta_k \nabla F(Z_k) \bullet \eta_{k-1} \\ = & -\|\nabla F(Z_k)\|_F^2 + \beta_k \nabla F(Z_k) \bullet \eta_{k-1}. \end{aligned} \tag{6}$$

If $|\nabla F(Z_k) \bullet \eta_{k-1}| > \frac{\epsilon^2}{2}$, by (4) and (6), we have that

$$\begin{aligned} \nabla F(Z_k) \bullet \eta_k = & -\|\nabla F(Z_k)\|_F^2 + \beta_k \nabla F(Z_k) \bullet \eta_{k-1} \\ = & -\|\nabla F(Z_k)\|_F^2 + \frac{\|\nabla F(Z_k)\|_F^2}{-\nabla F(Z_k) \bullet \eta_{k-1}} \nabla F(Z_k) \bullet \eta_{k-1} \\ = & -2\|\nabla F(Z_k)\|_F^2 \\ \leq & -2\epsilon^2 < 0. \end{aligned}$$

If $|\nabla F(Z_k) \bullet \eta_{k-1}| \leq \frac{\epsilon^2}{2}$, then we have that

$$\begin{aligned} \nabla F(Z_k) \bullet \eta_k = & -\|\nabla F(Z_k)\|_F^2 + \beta_k \nabla F(Z_k) \bullet \eta_{k-1} \\ \leq & -\|\nabla F(Z_k)\|_F^2 + \beta_k |\nabla F(Z_k) \bullet \eta_{k-1}| \\ \leq & -\|\nabla F(Z_k)\|_F^2 + \beta_k \frac{\epsilon^2}{2} \\ \leq & -\|\nabla F(Z_k)\|_F^2 + \frac{\|\nabla F(Z_k)\|_F^2}{2} \\ = & -\frac{\epsilon^2}{2} < 0. \end{aligned}$$

□

Step 3 is important for Algorithm 1 because it ensures that the objective function is monotonically decreasing. But we must prove that step 3 is to terminate in a finite step, i.e., there exists $\alpha_k > 0$ such that (5) holds.

Lemma 3. *Under Assumptions 1 and 2, let $\{Z_k\}$ be the sequence generated by Algorithm 1, then step 3 is to terminate in a finite step, i.e., there exists $\alpha_k > 0$ such that (5) holds.*

Proof. Suppose Algorithm 1 dose not terminate on iteration k, i.e., there exists positive constant $\epsilon > 0$ such that $\|\nabla F(Z_k)\|_F \geq \epsilon$. First, we prove that $\|\eta_k\|_F$ is bounded for all $k \geq 0$. We use mathematical induction to prove this conclusion. When $k = 0$, we have $\eta_0 = -\nabla F(Z_0)$, according to Lemma 1, we have $\|\eta_0\|_F = \|\nabla F(Z_0)\|_F \leq \kappa_g$, suppose η_k is bounded, then by (3), we have that

$$\begin{aligned}\|\eta_{k+1}\|_F =& \| - \nabla F(Z_{k+1}) + \beta_{k+1}\eta_k\|_F \\ \leq& \|\nabla F(Z_{k+1})\|_F + |\beta_{k+1}|\|\eta_k\|_F. \end{aligned} \tag{7}$$

By (4), if $|\nabla F(Z_{k+1}) \bullet \eta_k| > \frac{\epsilon^2}{2}$, then

$$\beta_{k+1} = \frac{\|\nabla F(Z_{k+1})\|_F^2}{-\nabla F(Z_{k+1}) \bullet \eta_k}.$$

By the bounded of $\nabla F(Z_{k+1})$ and η_k, we know that

$$|\beta_{k+1}| = \frac{\|\nabla F(Z_{k+1})\|_F^2}{|\nabla F(Z_{k+1}) \bullet \eta_k|} \leq \frac{2\kappa_g^2}{\epsilon^2}.$$

If $|\nabla F(Z_{k+1}) \bullet \eta_k| \leq \frac{\epsilon^2}{2}$, then by (4), we know that

$$|\beta_{k+1}| = \frac{\|\nabla F(Z_{k+1})\|_F^2}{2\epsilon} \leq \frac{\kappa_g^2}{2\epsilon^2}.$$

Hence, let $\kappa_\beta = \frac{\kappa_g^2}{2\epsilon^2}$, we have that $|\beta_{k+1}| \leq \kappa_\beta$. Hence, by (7), we have that

$$\|\eta_{k+1}\|_F \leq \kappa_g + \kappa_\beta\|\eta_k\|_F.$$

According to the assumption of the bounded of η_k, we obtain that there exists a positive constant κ_η such that

$$\|\eta_{k+1}\| \leq \kappa_\eta.$$

According to Assumption 2, the gradient of $F(Z)$ is Lipschitz-continuous, then

$$F(Z_{k+1}) \leq F(Z_k) + \alpha_k \nabla F(Z_k) \bullet \eta_k + \frac{\kappa_H \alpha_k^2}{2}\|\eta_k\|_F^2. \tag{8}$$

If $|\nabla F(Z_k) \bullet \eta_{k-1}| > \frac{\epsilon^2}{2}$, then by (5) and (8), we have that

$$\begin{aligned} & F(Z_{k+1}) - F(Z_k) - \alpha_k\beta\nabla F(Z_k) \bullet \eta_k \\ \leq& \alpha_k \nabla F(Z_k) \bullet \eta_k + \frac{\alpha_k^2}{2}\kappa_H\|\eta_k\|_F^2 - \alpha_k\beta\nabla F(Z_k) \bullet \eta_k \\ =& (1-\beta)\alpha_k \nabla F(Z_k) \bullet \eta_k + \frac{\alpha_k^2\kappa_H}{2}\|\eta_k\|_F^2 \\ \leq& (1-\beta)\alpha_k \nabla F(Z_k) \bullet (-\nabla F(Z_k) + \beta_k\eta_{k-1}) + \frac{\alpha_k^2\kappa_H^2}{2}\kappa_\eta^2 \\ \leq& (1-\beta)\alpha_k \left(-\|\nabla F(Z_k)\|_F^2 + \frac{\|\nabla F(Z_k)\|_F^2}{-\nabla F(Z_k) \bullet \eta_{k-1}} \nabla F(Z_k) \bullet \eta_{k-1} \right) + \frac{\alpha_k^2\kappa_H^2}{2}\kappa_\eta^2 \\ \leq& -2(1-\beta)\alpha_k\epsilon^2 + \frac{\alpha_k^2\kappa_H^2}{2}\kappa_\eta^2. \end{aligned} \tag{9}$$

By (9), we want $F(Z_{k+1}) - F(Z_k) - \alpha_k \beta \nabla F(Z_k) \bullet \eta_k \leq 0$, then

$$-2(1-\beta)\alpha_k \epsilon^2 + \frac{\alpha_k^2 \kappa_H^2}{2} \kappa_\eta^2 \leq 0. \tag{10}$$

By (10), we have that

$$\alpha_k \leq \frac{4(1-\beta)\epsilon^2}{\kappa_H^2 \kappa_\eta^2}.$$

Hence, we let $\alpha_k = \frac{2(1-\beta)\epsilon^2}{\kappa_H^2 \kappa_\eta^2} > 0$, then the conclusion holds.

If $|\nabla F(Z_k) \bullet \eta_{k-1}| \leq \frac{\epsilon^2}{2}$, similar to the proof of (9), we have that

$$\begin{aligned}
&F(Z_{k+1}) - F(Z_k) - \alpha_k \beta \nabla F(Z_k) \bullet \eta_k \\
\leq &(1-\beta)\alpha_k \left(-\|\nabla F(Z_k)\|_F^2 + \frac{\|\nabla F(Z_k)\|_F^2}{2\epsilon^2} \nabla F(Z_k) \bullet \eta_{k-1} \right) + \frac{\alpha_k^2 \kappa_H^2}{2} \kappa_\eta^2 \\
\leq &(1-\beta)\alpha_k \left(-\|\nabla F(Z_k)\|_F^2 + \frac{\|\nabla F(Z_k)\|_F^2}{2} \right) + \frac{\alpha_k^2 \kappa_H^2}{2} \kappa_\eta^2 \\
\leq &-\frac{(1-\beta)\alpha_k \epsilon^2}{2} + \frac{\alpha_k^2 \kappa_H^2}{2} \kappa_\eta^2.
\end{aligned}$$

Hence, to ensure $F(Z_{k+1}) - F(Z_k) - \alpha_k \beta \nabla F(Z_k) \bullet \eta_k \leq 0$, we want $-\frac{(1-\beta)\alpha_k \epsilon^2}{2} + \frac{\alpha_k^2 \kappa_H^2}{2} \kappa_\eta^2 \leq 0$, then

$$\frac{\alpha_k^2 \kappa_H^2}{2} \kappa_\eta^2 \leq \frac{(1-\beta)\alpha_k \epsilon^2}{2},$$

which means that

$$\alpha_k \leq \frac{(1-\beta)\epsilon^2}{4\kappa_H^2 \kappa_\eta^2}.$$

Let $\alpha_k = \frac{(1-\beta)\epsilon^2}{8\kappa_H^2 \kappa_\eta^2} > 0$, then the conclusion holds. □

The following theorem shows that the sequence $\{Z_k\}$ generated by Algorithm 1 converges to the critical point of problem (1).

Theorem 1. *Under Assumptions 1 and 2, the sequence $\{Z_k\}$ generated by Algorithm 1 satisfy*

$$\lim_{k \to \infty} \|\nabla f(Z_k)\|_F = 0.$$

It means that the iteration sequence $\{Z_k\}$ converges to the critical point of problem (1).

Proof. By (6), we have that

$$\nabla F(Z_k) \bullet \eta_k = -\|\nabla F(Z_k)\|_F^2 + \beta_k \nabla F(Z_k) \bullet \eta_{k-1}.$$

By (4), if $|\nabla F(Z_k) \bullet \eta_{k-1}| > \frac{\epsilon^2}{2}$, we have that

$$\begin{aligned} -\nabla F(Z_k) \bullet \eta_k =& \|\nabla F(Z_k)\|_F^2 - \beta_k \nabla F(Z_k) \bullet \eta_{k-1} \\ =& \|\nabla F(Z_k)\|_F^2 + \frac{\|\nabla F(Z_k)\|_F^2}{\nabla F(Z_k) \bullet \eta_{k-1}} \nabla F(Z_k) \bullet \eta_{k-1} \\ =& 2\|\nabla F(Z_k)\|_F^2. \end{aligned} \tag{11}$$

If $|\nabla F(Z_k) \bullet \eta_{k-1}| \leq \frac{\epsilon^2}{2}$, then

$$\begin{aligned} -\nabla F(Z_k) \bullet \eta_k =& \|\nabla F(Z_k)\|_F^2 - \beta_k \nabla F(Z_k) \bullet \eta_{k-1} \\ =& \|\nabla F(Z_k)\|_F^2 - \frac{\|\nabla F(Z_k)\|_F^2}{2\epsilon^2} \nabla F(Z_k) \bullet \eta_{k-1} \\ \geq& \frac{\|\nabla F(Z_k)\|_F^2}{2}. \end{aligned} \tag{12}$$

Combining (5), (11) and (12), we can see that

$$\begin{aligned} F(Z_k) - F(Z_{k+1}) \geq& -\alpha_k \beta \nabla f(Z_k) \bullet \eta_k \\ \geq& \frac{\alpha_k \beta}{2} \|\nabla f(Z_k)\|_F^2, \end{aligned}$$

hence, for any large enough $N > 0$, we have that

$$\begin{aligned} F(Z_0) - F(Z_N) =& \sum_{k=0}^{N} F(Z_k) - F(Z_{k+1}) \\ \geq& \sum_{k=0}^{N} \frac{1}{2} \alpha_k \beta \|\nabla F(Z_k)\|_F^2 \\ =& \frac{1}{2} \beta \sum_{k=0}^{N} \alpha_k \|\nabla F(Z_k)\|_F^2. \end{aligned} \tag{13}$$

According to Assumption 1, we know that $F(Z)$ is blow bounded, hence, by (13), we have that

$$\lim_{N \to \infty} \alpha_N \|\nabla F(Z_N)\|_F = 0. \tag{14}$$

According to Lemma 3, we know that $\alpha_N \geq \frac{(1-\beta)\epsilon^2}{8\kappa_H^2 \kappa_\eta^2} > 0$ for any index N if $\|\nabla F(Z_N)\|_F > \epsilon$, and

$$\alpha_N \|\nabla F(Z_N)\|_F \geq \frac{(1-\beta)\epsilon^2}{8\kappa_H^2 \kappa_\eta^2} \|\nabla F(Z_N)\|_F. \tag{15}$$

Combining (14) and (15), we have that

$$\lim_{k \to \infty} \|\nabla F(Z_k)\|_F = 0.$$

□

5. Local Convergence

In Section 4, we have shown that the sequence $\{\|\nabla F(Z_k)\|_F\}$ generated by the conjugate gradient algorithm converges globally to zero. We now discuss the superlinear convergence in a neighborhood of the solution Z^*.

First, we show that if $\frac{\|\eta_k\|_F^2}{\|\nabla F(Z_k)\|_F^2} \leq \frac{1-\beta}{\kappa_H}$, then $\alpha_k = 1$ for any iteration index k.

Lemma 4. *Under Assumptions 1 and 2, let* $\{Z_k\}$ *be the sequence generated by Algorithm 1, if* $\frac{\|\eta_k\|_F^2}{\|\nabla F(Z_k)\|_F^2} \leq \frac{(1-\beta)}{\kappa_H}$, *then* $\alpha_k = 1$.

Proof. First, we consider the follow inequality

$$\begin{aligned} &F(Z_{k+1}) - F(Z_k) - \beta \nabla F(Z_k) \bullet \eta_k \\ \leq &(1-\beta)\nabla f(Z_k) \bullet \eta_k + \frac{\kappa_H}{2}\|\eta_k\|_F^2 \\ \leq &-\frac{1-\beta}{2}\|\nabla F(Z_k)\|_F^2 + \frac{\kappa_H}{2}\|\eta_k\|_F^2. \end{aligned}$$

We want $f(Z_{k+1}) - f(Z_k) - \beta \nabla f(Z_k) \bullet \eta_k \leq 0$, then

$$\frac{\kappa_H}{2}\|\eta_k\|_F^2 \leq \frac{1-\beta}{2}\|\nabla F(Z_k)\|_F^2. \tag{16}$$

By (16) and $\frac{\|\eta_k\|_F^2}{\|\nabla F(Z_k)\|_F^2} \leq \frac{1-\beta}{\kappa_H}$, we have that $f(Z_{k+1}) - f(Z_k) - \beta \nabla f(Z_k) \bullet \eta_k \leq 0$ holds, which means that $\alpha_k = 1$. □

Finally, we show that the sequence $\{Z_k\}$ generated by Algorithm 1 converges to critical point Z^* superlinearly.

Theorem 2. *Under Assumptions 1 and 2, if* $\frac{\|\eta_k\|_F^2}{\|\nabla F(Z_k)\|_F^2} \leq \frac{1-\beta}{\kappa_H}$ *and the objective function* $F(Z)$ *satisfies the local error bound condition, i.e., there exists* $\kappa_1 > 0$ *such that* $\|F(X) - F(Y)\|_F \geq \kappa_1\|X - Y\|_F$ *for all* $X, Y \in R^{m\times n}$, *then the sequence* $\{Z_k\}$ *generated by Algorithm 1 converges to critical point* Z^* *superlinearly.*

Proof. Because $\frac{\|\eta_k\|_F^2}{\|\nabla F(Z_k)\|_F^2} \leq \frac{1-\beta}{\kappa_H}$, according to Lemma 4, we have $\alpha_k = 1$. Then,

$$\begin{aligned} \kappa_1\|Z_{k+1} - Z^*\|_F \leq &\|F(Z_{k+1}) - F(Z^*)\|_F \\ = &\|\nabla F(Z_{k+1}) \bullet (Z_{k+1} - Z^*) + o(\|Z_{k+1} - Z^*\|_F)\|_F \\ = &\|(\nabla F(Z_{k+1}) - \nabla F(Z_k)) \bullet (Z_{k+1} - Z^*) \\ &+ \nabla F(Z_k) \bullet (Z_{k+1} - Z^*) + o(\|Z_{k+1} - Z^*\|_F)\|_F \\ \leq &\|\nabla F(Z_{k+1}) - \nabla F(Z_k)\|_F\|Z_{k+1} - Z^*\|_F \\ &+ \|\nabla F(Z_k) \bullet (Z_k + \eta_k - Z^*)\|_F + o(\|Z_{k+1} - Z^*\|_F) \\ = &\|\nabla F(Z_{k+1}) - \nabla F(Z^*) + \nabla F(Z^*) - \nabla F(Z_k)\|_F\|Z_{k+1} - Z^*\|_F \\ &+ \|\nabla F(Z_k) \bullet (Z_k + \eta_k - Z^*)\|_F + o(\|Z_{k+1} - Z^*\|_F) \\ \leq &\|\nabla F(Z_{k+1}) - \nabla F(Z^*)\|_F\|Z_{k+1} - Z^*\|_F + \|\nabla F(Z_k) - \nabla F(Z^*)\|_F\|Z_{k+1} - Z^*\|_F \\ &+ \|\nabla F(Z_k) \bullet (Z_k + \eta_k - Z^*)\|_F + o(\|Z_{k+1} - Z^*\|_F) \\ \leq &\|\nabla F(Z_{k+1}) - \nabla F(Z^*)\|_F\|Z_{k+1} - Z^*\|_F + \|\nabla F(Z_k) - \nabla F(Z^*)\|_F\|Z_{k+1} - Z^*\|_F \\ &+ \|\nabla F(Z_k) - \nabla F(Z^*)\|_F\|Z_k - Z^*\|_F + \|\nabla F(Z_k) \bullet \eta_k\|_F + o(\|Z_{k+1} - Z^*\|_F). \end{aligned} \tag{17}$$

According to Assumption 2,

$$\begin{aligned} \|\nabla F(Z_{k+1}) - \nabla F(Z^*)\|_F\|Z_{k+1} - Z^*\|_F \leq &\kappa_H\|Z_{k+1} - Z^*\|_F^2 \\ \|\nabla F(Z_k) - \nabla F(Z^*)\|_F\|Z_{k+1} - Z^*\|_F \leq &\kappa_H\|Z_{k+1} - Z^*\|_F\|Z_k - Z^*\|_F \\ \|\nabla F(Z_k) - \nabla F(Z^*)\|_F\|Z_k - Z^*\|_F \leq &\kappa_H\|Z_k - Z^*\|_F^2. \end{aligned} \tag{18}$$

By (11) and (12), we have that

$$\begin{aligned}\|\nabla F(Z_k) \bullet \eta_k\|_F \leq & 3\|\nabla F(Z_k)\|_F^2 \\ = & 3\|\nabla F(Z_k) - \nabla F(Z^*)\|_F^2 \\ \leq & 3\kappa_H^2\|Z_k - Z^*\|^2. \end{aligned} \tag{19}$$

Combining (17), (18) and (19), we have that

$$\begin{aligned}\kappa_1\|Z_{k+1} - Z^*\|_F \leq & \kappa_H\|Z_{k+1} - Z^*\|_F^2 + \kappa_H\|Z_{k+1} - Z^*\|_F\|Z_k - Z^*\|_F \\ & + \kappa_H\|Z_k - Z^*\|_F^2 + \frac{1}{2}\|\nabla F(Z_k) - \nabla F(Z^*)\|_F^2 + o(\|Z_{k+1} - Z^*\|_F) \\ \leq & \kappa_H\|Z_{k+1} - Z^*\|_F^2 + \kappa_H\|Z_{k+1} - Z^*\|_F\|Z_k - Z^*\|_F \\ & + 3\kappa_H^2\|Z_k - Z^*\|_F^2 + o(\|Z_{k+1} - Z^*\|_F). \end{aligned} \tag{20}$$

Since $Z_k \to Z^*$, we know that

$$\max\{o(\|Z_{k+1} - Z^*\|_F), \kappa_H\|Z_{k+1} - Z^*\|_F^2\} \leq \frac{\kappa_1}{4}\|Z_{k+1} - Z^*\|_F,$$

for large enough k.

Hence, by (20), we obtain that

$$\kappa_1\|Z_{k+1} - Z^*\|_F \leq \kappa_H\|Z_{k+1} - Z^*\|_F\|Z_k - Z^*\|_F + 3\kappa_H^2\|Z_k - Z^*\|_F^2 + \frac{\kappa_1}{2}\|Z_{k+1} - Z^*\|_F,$$

which means that

$$\frac{\kappa_1}{2}\|Z_{k+1} - Z^*\|_F \leq \kappa_H\|Z_{k+1} - Z^*\|_F\|Z_k - Z^*\|_F + 3\kappa_H^2\|Z_k - Z^*\|_F^2. \tag{21}$$

Divide by $\|Z_k - Z^*\|_F$ on both sides of the inequality (21), we have that

$$\frac{\|Z_{k+1} - Z^*\|_F}{\|Z_k - Z^*\|_F} \leq \frac{2\kappa_H}{\kappa_1}\|Z_{k+1} - Z^*\|_F + \frac{6\kappa_H^2}{\kappa_1}\|Z_k - Z^*\|_F.$$

As $Z_{k+1} \to Z^*$ and $Z_k \to Z^*$ when $k \to +\infty$, we can obtain that

$$\frac{\|Z_{k+1} - Z^*\|_F}{\|Z_k - Z^*\|_F} \to 0, \tag{22}$$

as $k \to +\infty$, i.e.,

$$\|Z_{k+1} - Z^*\|_F = o(\|Z_k - Z^*\|_F),$$

which means that the sequence $\{Z_k\}$ generated by Algorithm 1 converges to critical point Z^* superlinearly. □

6. Numerical Results

In Section 3, the approximation conjugate gradient method is given, and the global and local superlinear convergence of this algorithm are given in Sections 4 and 5, respectively. Next, we give the numerical experiments to illustrate the performance of the method proposed in this paper. We apply ACGA to synthetic problems. The synthetic data of the experiments are increased as they are in [29]. First, the matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ with independent and identically distributed Gaussian entries such that $M = UV^T$ is filled with zero-mean and unite-variance nonindependent Gaussian entries are given. Then, the sample $K = \rho r(m + n - r)$ is given at random, where ρ is the oversampling factor. To test the effectiveness of the algorithm, we compare ACGA to RMC [29], AOPMC [30], GRASTA [31], and l_2 method RTRMC [9].

RMC was the method for solving the low-rank matrix completion, in which the fraction of the entries was corrupted by non-Gaussian noise and typically outliers. This method smoothed the $l\infty$ norm, and the low-rank constraint was dealt with by Riemannian optimization.

AOPMC was a robust method for recovering the low-rank matrix using the adaptive outlier pursuit technique and Riemannian trust-region method when part of the measurements was damaged by outliers.

GRASTA was a Grassmannian robust adaptive subspace tracking algorithm. This method needed the rank-r SVD decomposition for the matrix and could improve computation speed while still allowing for accurate subspace learning.

RTRMC exploited the geometry of the low-rank constraint to recast the problem as an unconstrained optimization problem on a single Grassmann manifold. The second-order Riemannian trust-region methods and Riemannian conjugate gradient methods are used to solve the unconstrained optimization problem.

Before solving the low-rank recovery problem, we will introduce the parameters selected in the actual calculation of the algorithm proposed in this paper. The important parameter of this paper is $\beta \in (0, 1)$. To obtain a larger search step size and the objective function of problem (P) descent more quickly, the backtracking line search parameter β in step 3 of the algorithm selects a smaller value close to 0, so $\beta = 0.15$ is selected in the algorithm proposed in this paper.

To solve the low-rank recovery problem using the algorithm in this paper, we use MATLAB (2014a) to write a computer program, and the computer is a ThinkPad T480 (CPU is i7-8550U, main frequency is 1.99 Hz, memory is 8 G). The termination accuracy of the algorithm is chosen by $\epsilon = 10^{-2}$, $\epsilon = 10^{-3}$ and $\epsilon = 10^{-5}$, respectively.

In this paper, we selected 12 problems to test the effectiveness of the algorithm. The number of rows and columns of the selected questions, as well as the rank of matrices U and V, are shown in the following Table 1. In Table 1, m and n denote the number of rows and columns of matrix M, respectively. r denotes the rank of matrices U and V. To test our algorithm, we randomly generated 50 different scenarios for each problem in Table 1 and obtained the final number of iterations by a weighted average of all random problem results. We have set three different termination accuracies for our algorithm ($\epsilon = 10^{-2}, 10^{-3}, 10^{-5}$) and used our algorithm to calculate the test problems in Table 1 for each accuracy. We also used the four classic methods mentioned above to solve every test problem in Table 1 and recorded the number of iterations of each method to solve every test problem in Table 1. The comparison results of our algorithm with the other four algorithms under different termination accuracies are shown in Figures 1–3.

Table 1. The information of test problems.

Problem	m	n	r	Problem	m	n	r
1	100	100	20	7	100	300	20
2	300	300	20	8	100	500	20
3	500	500	20	9	100	800	20
4	800	800	20	10	100	1000	20
5	1000	1000	20	11	300	1000	20
6	50	100	20	12	500	1000	20

To draw a comparison diagram of the results of five algorithms, we use the performance comparison formula of the algorithm proposed by Dolan and More [32] to calculate the computational efficiency between different algorithms. Here are the specific formulas:

$$r_{p,s} = \frac{\tau_{p,s}}{\min\{\tau_{p,u} : u \in S\}}, \tag{23}$$

where S denotes the set of algorithms. Let P denote problems set, $n_s = |S|$ and $n_p = |P|$. For $\forall t \geq 1$, let

$$\rho_s(t) = \frac{1}{n_p} size\{p \in P : r_{p,s} \leq t\}, \tag{24}$$

where $\rho_s(t)$ represent the efficiency of each solver.

We solve the test problems in Table 1 using ACGA, RMC, AOPMC, GRASTA and l_2 method RTRMC, respectively. First, the terminate parameter is chosen by $\epsilon = 10^{-2}$. The test results are reported in the following Figure 1.

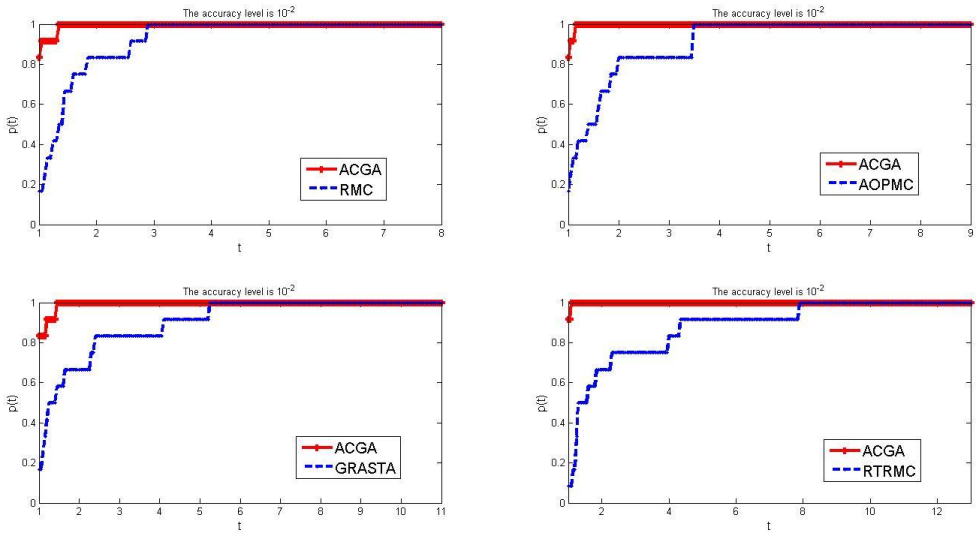

Figure 1. Comparison results of the number of iterations between ACGC and the other four algo−rithms when $\epsilon = 10^{-2}$ and $r = 20$.

From Figure 1, we can see that our algorithm has significant effectiveness compared to the other four algorithms for calculating low-rank matrices. Especially when $t = 1$, our algorithm has reached 0.83 and even reached 0.9 in comparison with RTRMC, but other algorithms only reach between 0.1 and 0.2. This indicates that at an accuracy level of $\epsilon = 10^{-2}$, our algorithm can solve 83–90% of test problems faster than the other four algorithms. By (23) and (24), the red curve representing our algorithm tends to 1 faster than the blue curve representing other algorithms, indicating that our algorithm is more effective in solving the test problems in Table 1 using all five algorithms. From the abscissa of the four subgraphs mentioned above, it can be seen that our algorithm spends much fewer iterations solving most testing problems than the other four algorithms. In summary, our algorithm is more effective in solving the test problems in Table 1.

Next, we use five algorithms to solve the test problems when the terminate parameter is chosen by $\epsilon = 10^{-3}$. The test results are reported in Figure 2.

From Figure 2, we can see that when the termination parameter is set to $\epsilon = 10^{-3}$, it indicates that our algorithm's red curve tends to 1 much faster than the other four algorithms. When $t = 1$, the red curve has already reached 0.8, or even 0.9, while other algorithms only reach around 0.2, with a worst-case value of 0.1. This indicates that even with improved termination accuracy, our algorithm still has high computational efficiency, i.e., our algorithm spends fewer total iterations when solving more than 80% of problems. According to the abscissa of each subgraph, it can be seen that our algorithm spends much fewer iterations in solving many testing problems in Table 1 than the other four algorithms, indicating that our algorithm has higher efficiency in solving testing problems.

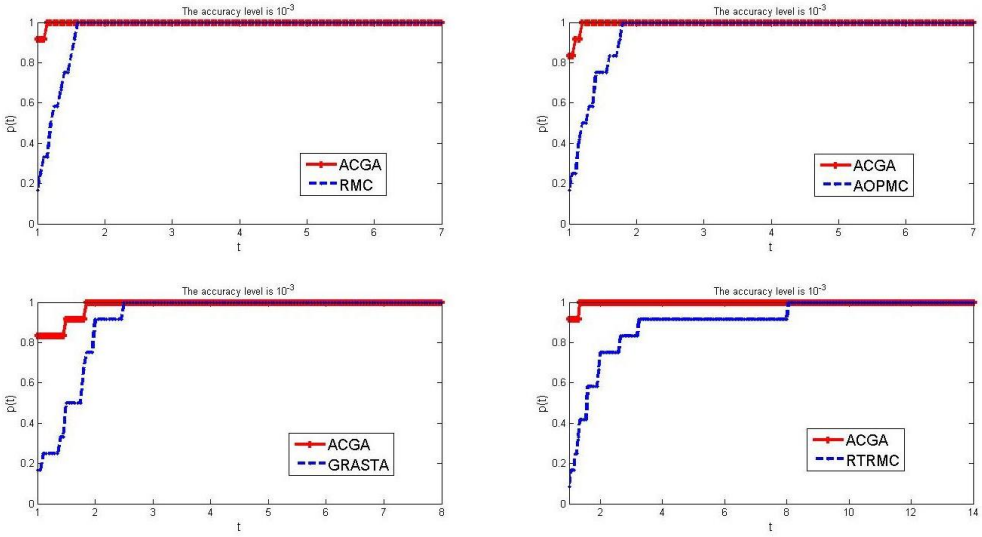

Figure 2. Comparison results of the number of iterations between ACGC and other four algorithms when $\epsilon = 10^{-3}$ and $r = 20$.

From Figure 3, we clearly see that with the continuous improvement of termination accuracy, the five algorithms listed in this article have shown a decrease in efficiency when solving test problems compared to those with accuracies of $\epsilon = 10^{-2}$ and $\epsilon = 10^{-3}$. In other words, the number of iterations required to solve all test problems has greatly increased. However, when the termination accuracy is $\epsilon = 10^{-5}$, our algorithm still has higher efficiency compared to the other four algorithms. The main manifestation is that the red curve representing our algorithm has reached a range of 0.8 to 0.9 when $t = 1$, but other algorithms only reach around 0.4 or 0.5. This indicates that our algorithm spends less iteration time solving most test problems than the other four algorithms. In addition, the speed at which the red curve tends to 1 is also faster than the blue curve representing other algorithms, and it can be clearly seen from the abscissa that although the overall solving efficiency has decreased, our algorithm still spends far fewer iterations on solving some problems than the other four algorithms.

Finally, we changed the termination parameter to $\epsilon = 10^{-5}$ and still used five different algorithms to solve the test problem. The results are reported in Figure 3.

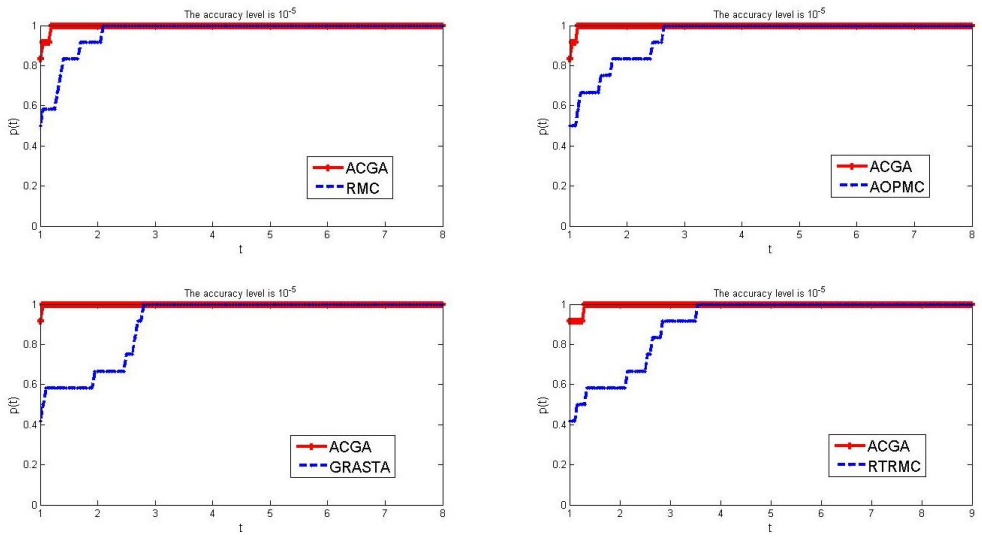

Figure 3. Comparison results of the number of iterations between ACGC and other four algorithms when $\epsilon = 10^{-5}$ and $r = 20$.

We choose the rank $r = 15, r = 25, r = 30$ and $r = 35$ for solving the test problems in Table 1 using Algorithm 3, RMC, AOPMC , GRASTA and RTRMC, respectively. The number of iterations of every method for solving every problem of Table 1 is reported. Using (23) and (24), we obtain a comparison graph of the number of iterations when we solve every problem in Table 1 using four algorithms. The comparison results are reported in Figure 4.

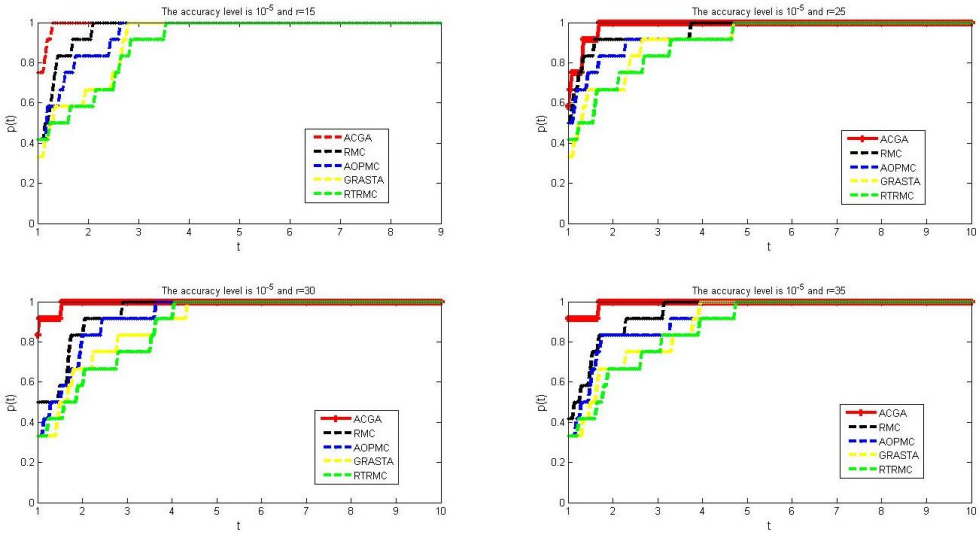

Figure 4. Comparison results of the number of iterations for five algorithms when we choose $r = 15$, $r = 25$, $r = 30$, and $r = 35$, respectively.

By (23) and (24), we can see that the red curve tends to 1 and is also faster than other curves representing other algorithms. This means that our algorithm costs fewer iterations than the other four algorithms when we solve all test problems in the table, with $r = 15, r = 25, r = 30$, and $r = 35$, respectively.

Next, we will test our algorithm on the Netflix dataset, which is a real-world dataset in the Netflix Prize [33]. Netflix is a movie rental company that recommends movies to its users. Hence, they have much information from users. This information is translated into a 480,189 $\times$ 17,770 matrix (this matrix is denoted as M). To test our algorithm, we randomly selected 10 matrices of 500 $\times$ 1000 in M, and the rank of U and V are chosen by $r = 20$. We will choose $\epsilon = 10^{-2}, \epsilon = 10^{-3}, \epsilon = 10^{-4}$ and $\epsilon = 10^{-5}$ to test five algorithms. The number of iterations for each algorithm at different accuracies and ranks is recorded in Figure 5.

From Figure 5, we can see that the speed at which the red curve tends towards 1 is the fastest at different accuracies. This means that for different termination accuracies, the number of iterations of our algorithm is less than the other four algorithms in most cases when solving all real-test problems, indicating that our algorithm is effective in solving practical problems.

We determine the complexity of the algorithm by recording the *CPU* time when the algorithm solves the problems in Table 1. In other words, the more *CPU* time is spent, the higher its complexity; conversely, the lower its computational complexity. Next, we calculate the test problems in Table 1 using five different algorithms and record the *CPU* time by each algorithm in Figure 6.

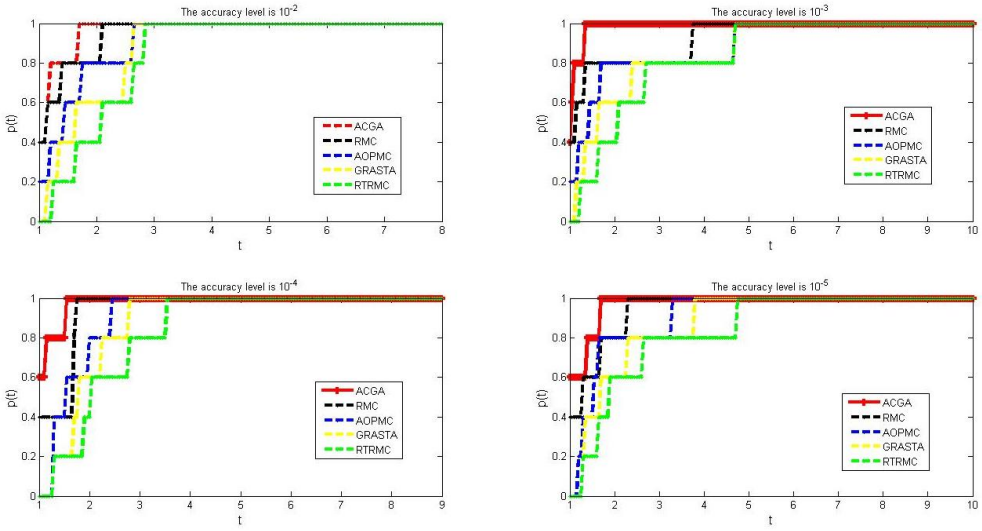

Figure 5. Comparison results of number of iterations for five algorithms when we choose $\epsilon = 10^{-2}$, $\epsilon = 10^{-3}, \epsilon = 10^{-4}$ and $\epsilon = 10^{-5}$, respectively.

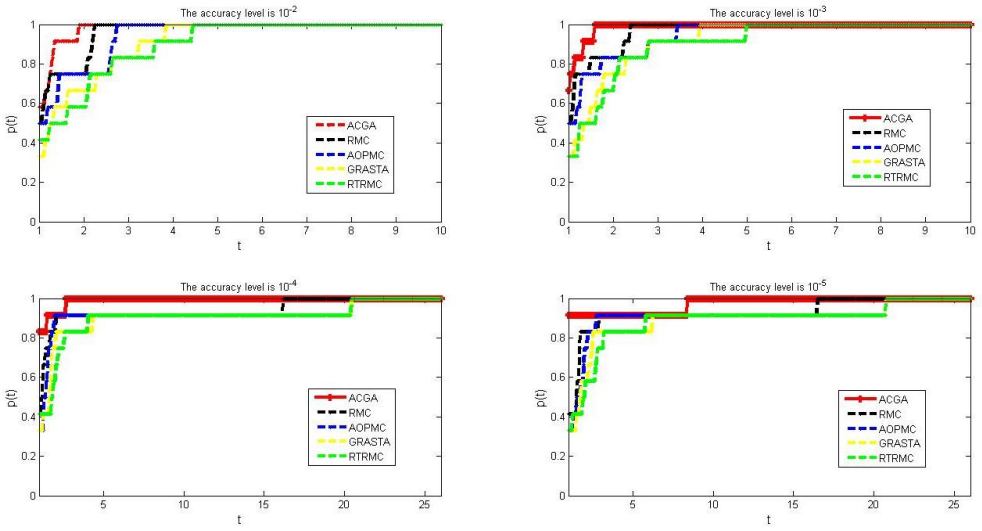

Figure 6. Comparison results of *CPU* time for five algorithms when we choose $\epsilon = 10^{-2}$, $\epsilon = 10^{-3}, \epsilon = 10^{-4}$ and $\epsilon = 10^{-5}$, respectively.

From Figure 6, we can see that the speed at which the red curve tends towards 1 is the fastest, indicating that our algorithm spends less *CPU* time than other algorithms when solving most of the test problems in Table 1. In particular, from Figure 6, we can see that the yellow curve representing the SVD decomposition algorithm is much lower than the red curve representing our algorithm, which means that our algorithm has a lower computational complexity than the SVD algorithm.

7. Concluding Remarks

The main purpose of this paper is to introduce the approximation conjugate gradient method, which avoids the complex SVD decomposition for solving the low-rank matrix recovery. The idea is to extend the approximation conjugate gradient method and the conjugate gradient parameter update technique to the matrix calculation and obtain the search direction. The backtracking linear search technique ensures the global convergence of the algorithm. At the same time, this technique ensures that the objective function decreases monotonously. The local superlinear convergence of the algorithm is also given, and the numerical results are reported to show the effectiveness of the algorithm.

Author Contributions: Methodology, P.W. and D.Z.; validation, Z.C.; formal analysis, Z.C. and P.W.; writing original draft preparation, Z.C.; writing review and editing, P.W. and Z.C.; visualization, Z.C. and P.W.; supervision, P.W. and D.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation (11371253) and Hainan Natural Science Foundation (120MS029).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xu, J.; Zhu, W.; Cai, L.; Liao, B.; Meng, Y.; Xiang, J.; Yuan, D.; Tian, G.; Ynag, J. LRMCMDA: Predicting miRNA-diease association by interating low-rank matrix completion with miRNA and disease similarity information. *IEEE Access* **2020**, *8*, 80728–80738.
2. Sun, R.; Luo, Z. Guaranteed matrix completion via non-convex factorization. *IEEE Trans. Inf. Theory* **2016**, *62*, 6535–6579.
3. Tu, S.; Boczar, R.; Simchowitz, M.; Soltanolkotabi, M.; Recht, B. Low-rank solutions of linear matrix equations via procrustes flow. In Proceedings of the International Conference on Machine Learning, PMLR, New York City, NY, USA, 19–24 June 2016; pp. 964–973.
4. Keshavan, R.H.; Montanari, A.; Oh, S. Matrix completion from a few entries. *IEEE Trans. Inf. Theory* **2010**, *56*, 2980–2998.
5. Ngo, T.; Saad, Y. Scaled gradients on grassmann manifolds for matrix completion. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1412–1420.
6. Vandereycken, B. Low-rank matrix completion by Riemannian optimization. *SIAM J. Optim.* **2013**, *23*, 1214–1236.
7. Mishra, B.; Sepulchre, R. R3MC: A Riemannian three-factor algorithm for low-rank matrix completion. In Proceedings of the 53rd IEEE Conference on Decision and Control, Los Angeles, CA, USA, 15–17 December 2014; pp. 1137–1142.
8. Mishra, B.; Meyer, G.; Bonnabel, S.; Sepulchre, R. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Comput. Stat.* **2014**, *29*, 591–621.
9. Boumal, N.; Absil, P. Low-rank matrix completion via preconditioned optimization on the grassmann manifold. *Linear Algebra Appl.* **2015**, *475*, 200–239.
10. Wei, K.; Cai, J.; Chan, T.; Leung, S. Guarantees of Riemannian optimization for low rank matrix recovery. *SIAM J. Matrix Anal. Appl.* **2016**, *37*, 1198–1222.
11. Najafi, S.; Hajarian, M. An improved Riemannian conjugate gradient method and its application to robust matrix completion. *Numer. Algorithms* **2023**, 1–14. [CrossRef]
12. Duan, S.; Duan, X.; Li, C.; Li, J. Riemannian conjugate gradient method for low-rank tensor completion. *Adv. Comput. Math.* **2023**, *49*, 41. [CrossRef]
13. Wen, Z.; Yin, W.; Zhang, Y. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Math. Prog. Comput.* **2012**, *4*, 333–361.
14. Hardt, M. Understanding alternating minimization for matrix completion. In Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, Philadelphia, PA, USA, 18–21 October 2014; pp. 651–660.
15. Jain, P.; Netrapalli, P. Fast exact matrix completion with finite samples. In Proceedings of the 28th Conference on Learning Theory, Paris, France, 3–6 July 2015; pp. 1007–1034.
16. Yi, X.; Park, D.; Chen, Y.; Caramanis, C. Fast algorithms for robust PCA via gradient descent. In Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 4159–4167.
17. Zheng, Q.; Lafferty, J. Convergence analysis for rectangular matrix completion using burer-monteiro factorization and gradient descent. *arXiv* **2016**, arXiv:1605.07051
18. Chen, J.; Liu, D.; Li, X. Nonconvex rectangular matrix completion via gradient descent without $l_{2\infty}$ regularization. *IEEE Trans. Inf. Theory* **2020**, *66*, 5806–5841.
19. Haldar, J.P.; Hernando, D. Rank-constrained solutions to linear matrix equations using powerfactorization. *IEEE Signal Process. Lett.* **2009**, *16*, 584–587.
20. Ma, C.; Wang, K.; Chi, Y.; Chen, Y. Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval, matrix completion and blind deconvolution. *Found. Comput. Math.* **2019**, *20*, 451–632.
21. Ma, C.; Li, Y.; Chi, Y. Beyond procrustes: Balancing-free gradient descent for asymmetric low-rank matrix sensing. *IEEE Trans. Signal Process.* **2021**, *69*, 867–877.
22. Tong, T.; Ma, C.; Chi, Y. Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. *J. Mach. Learn. Res.* **2021**, *22*, 1–63.
23. Zilber, P.; Nadler, B. GNMR: A provable one-line algorithm for low rank matrix recovery. *SIAM J. Math. Data Sci.* **2022**, *4*, 909–934.
24. Li, H.; Peng, Z.; Pan, C.; Zhao, D. Fast Gradient Method for Low-Rank Matrix Estimation. *J. Sci. Comput.* **2023**, *96*, 41. [CrossRef]

25. Upadhyay, B.B.; Pandey, R.K.; Liao, S. Newton's method for intervalvalued multiobjective optimization problem. *J. Ind. Manag. Optim.* **2024**, *20*, 1633–1661.
26. Upadhyay, B.B.; Pandey, R.K.; Pan, J.; Zeng, S. Quasi-Newton algorithms for solving interval-valued multiobjective optimization problems by using their certain equivalence. *J. Comput. Appl. Math.* **2024**, *438*, 115550.
27. Fletcher, R. *Practical Methods of Optimization*; John Wiley & Sons: New York, NY, USA, 2013.
28. Zhang, H.; Conn, A.R.; Scheinberg, K. A derivative-free algorithm for least-squares minimization. *SIAM J. Optim.* **2010**, *20*, 3555–3576.
29. Cambier, L.; Absil, P.-A. Roubst low-rank matrix completion by riemannian optimization. *SIAM J. Sci. Comput.* **2016**, *38*, S440–S460.
30. Yan, M.; Yang, Y.; Osher, S. Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit. *J. Sci. Comput.* **2013**, *56*, 433–449.
31. He, J.; Balzano, L.; Szlam, A. Incremental gradient on the Grassmannian for online foreground and background separation in subsampled video. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 1568–1575.
32. Dolan, E.D.; More, J.J. Benchmarking optimization software with performance profiles. *Math. Profiles* **2002**, *91*, 201–213.
33. Bennett, J.; Lanning, S. The Netflix Prize. In Proceedings of the KDD Cup and Workshop, San Jose, CA, USA, 12 August 2007; p. 35.