Artificial Intelligence and Robotics Research 人工智能与机器人研究, 2026, 15(5), 1276-1292
Published Online September 2026 in Hans. https://www.hanspub.org/journal/airr
https://doi.org/10.12677/airr.2026.155117

Hans汉斯

基于随机素描的单通高阶张量分解算法

李　翼，覃文金*

西南大学数学与统计学院，重庆

收稿日期：2026年8月3日；录用日期：2026年9月9日；发布日期：2026年9月20日

摘　要

基于随机素描技术与张量－张量乘积框架，本文研究了若干针对高阶张量LU分解的单通随机算法。这类算法仅需对原始张量进行一次访问遍历，因此非常适用于存储在核心内存之外、或以流方式生成的大规模高阶张量。文中已严谨推导了这些算法的误差上界理论与计算复杂度。数值实验表明，相比于当前前沿的随机张量分解算法，所提出的高效单通算法在精度与CPU运行时间上均具有优势性。

关键词

高阶张量分解，单通随机算法，高斯投影，误差上界理论

Single-Pass High-Order Tensor Decomposition Algorithms Based on Randomized Sketching Techniques

Yi Li, Wenjin Qin*

School of Mathematic and Statistics, Southwest University, Chongqing

Received: August 3, 2026; accepted: September 9, 2026; published: September 20, 2026

Abstract

In this paper, by leveraging tensor-tensor product framework, we develop some single-pass randomized algorithms for high-order tensor LU decomposition based on sketching techniques. These algorithms need only one pass over the original tensor and hence are very suitable for extremely large and high-dimensional tensor stored outside of core memory or generated in a streaming fashion. Rigorous error upper bound theory and complexity analysis of these algorithms have been derived. Numerical experiments show that the proposed efficient single-pass algorithms have the similar accuracy

*通讯作者。

文章引用: 李翼, 覃文金. 基于随机素描的单通高阶张量分解算法[J]. 人工智能与机器人研究, 2026, 15(5): 1276-1292. DOI: 10.12677/airr.2026.155117

and CPU runtime compared with the state-of-the-art randomized algorithms for high-order tensor decomposition.

Keywords

High-Order Tensor Decomposition, Single-Pass Randomized Algorithm, Gaussian Projection, Error Upper Bound Theory

1. 引言

张量(也称为多维数组)是现实世界中多种数据的自然表示形式，广泛存在于计算机视觉、信号图像处理、遥感、生物医学、交通网络等诸多研究领域[1]-[4]。然而这类张量数据通常具有体量庞大、维数极高、结构复杂等显著特点，这无疑给高效处理、传输与存储带来了极大的挑战。幸运的是，这些张量数据普遍蕴含可被挖掘利用的低秩内在结构，依托张量分解技术[5]-[10]即可实现对原始数据的高效近似表示。然而，传统的确定性张量分解方法在处理大规模张量数据时面临计算效率低、存储成本高等瓶颈。因此，如何发展高效且精确的大规模张量数据近似表示框架，是当前张量数据处理与分析中的核心问题。

随机素描(Randomized Sketching)是一种实用的数据压缩技术，它借助于随机投影、随机采样、频繁方向、核心集等方法来近似原始数据[11]-[13]。尽管素描技术可能会小幅降低近似精度，但它能大幅降低计算与存储复杂度。正因如此，素描技术被广泛应用于低秩张量近似领域。诸多研究者在不同张量分解框架下提出了高效的张量素描算法并成功应用于克罗内克积回归、多项式逼近、深度卷积神经网络构建等重要任务中。例如，在张量塔克 Tucker 框架[6]下，结合随机投影与幂迭代策略，文献[14]-[16]针对高阶奇异值分解(简称 HOSVD)与序列截断高阶奇异值分解(简称 ST-HOSVD)设计了若干自适应的随机 Tucker 算法。Ahmadi-Asl 等人针对张量环框架[8] (简称 TR)下的随机 TR 算法进行了系统性的综述[17]。在张量列车框架[7] (简称 TT)下，Shi 等人提出了若干可并行的随机素描算法，用于求解低秩 TT 张量近似[18]；Al Daas 等人先后提出了张量列车算术运算的并行算法，并进一步开发了面向 TT 张量舍入的新型随机化算法[19]。在张量奇异值分解框架[9] (简称 T-SVD)下，文献[20]-[22]基于高斯随机投影与随机采样方案，提出了系列面向低管秩张量近似的高效随机化算法。然而，上述提及的随机化算法至少需要对输入张量完成两次遍历访问。当张量数据规模极大、必须存储于核外内存时，反复遍历输入的开销会变得极高。

近年来，学界陆续提出了一些针对单通随机算法的相关研究。与前述提及的多通随机算法相比而言，单通随机算法是指只访问原始数据一次来完成所有处理的方法，其优势在于效率高、实时性强，由于数据只被访问一次，能够节省存储和计算资源，特别适合大数据场景。首个面向奇异值分解(简称 SVD)的单遍随机化算法见于文献[23]，该方法借助两个随机矩阵分别构造两组素描，以此获取原始矩阵数据的列空间与行空间信息。后续文献[24]提出了该算法的新型变体，它们在数学层面完全等价。文献[25] [26]进一步对这类算法开展了深入研究，其中文献的方法将第二个随机矩阵设置为由首个随机矩阵生成的素描结果。此外，文献[27]设计了几种面向大规模矩阵 LU 分解的单通随机算法。在张量－张量乘积框架下，文献[28] [29]研究了单通随机化的低管秩张量近似算法。尽管上述方法在某些应用场景上取得了一定的成就，然而它们只适用于低阶张量数据(二阶矩阵、三阶张量)。随着现代信息技术的持续迭代，实际应用场景中面临的张量数据实际上往往比矩阵或三阶张量更为复杂，例如四阶彩色视频、四阶多时遥感数据、

五阶光场图像数据集、六阶双向纹理函数图像。因此，本文考虑如何发展面向这些复杂高阶张量的单通随机化近似算法，以满足现代大规模数据在快速压缩表示、低开销存储与高效传输层面的核心需求。

本文的结构安排如下：第 2 节总结了本文用到的重要知识，包括基本的符号、张量代数基础、随机矩阵理论；第 3 节主要提供了几类面向大规模张量近似的单通随机算法，同时给出算法的误差界理论和复杂度分析；第 4 节主要给出了提出单通随机张量 LU 近似算法的误差界理论证明；第 5 节进行了大量的实验来验证提出算法的有效性；第 6 节对本文内容进行总结。

2. 预备知识

向量用小写黑体字母表示，如 $\boldsymbol{x}$，x_i 为 $\boldsymbol{x}$ 索引 i 处的值；矩阵用大写黑体字母表示，如 $\boldsymbol{A}$，$\boldsymbol{A}_{ij}$ 为 $\boldsymbol{A}$ 索引 (i,j) 处的值；d -阶张量($d\geq 3$)用大写黑体手写字母表示，如 $\mathcal{A}$，$\mathcal{A}(i_1,\cdots,i_d)$ 或 $\mathcal{A}_{i_1,\cdots,i_d}$ 为张量 $\mathcal{A}\in\mathbb{R}^{n_1\times\cdots\times n_d}$ 索引 $(i_1,\cdots,i_d)$ 处的值；$\mathcal{A}(:,:,i_3,\cdots,i_d)$、$\boldsymbol{A}^{(i_3,\cdots,i_d)}$ 或 $\boldsymbol{A}^{\langle j\rangle}$ 表示该位置的正面矩阵切片；$\mathcal{A}\Delta\mathcal{B}$ 表示按张量 $\mathcal{A}\in\mathbb{R}^{n_1\times\cdots\times n_d}$ 和 $\mathcal{B}\in\mathbb{R}^{n_1\times\cdots\times n_d}$ 的矩阵切片相乘，即 $\mathcal{C}=\mathcal{A}\Delta\mathcal{B}$ 满足 $\mathcal{C}^{(i_3,\cdots,i_d)}=\mathcal{A}^{(i_3,\cdots,i_d)}\cdot\mathcal{B}^{(i_3,\cdots,i_d)}$；$\boldsymbol{A}_{(k)}\in\mathbb{R}^{n_k\times\prod_{j\neq k}n_j}$ 表示 d -阶张量 $\mathcal{A}\in\mathbb{R}^{n_1\times\cdots\times n_d}$ 沿着第 k 模态展开的矩阵。

定义 1 (模 k 张量 – 矩阵乘法 tensor). 设 $\mathcal{A}\in\mathbb{R}^{n_1\times\cdots\times n_k\times\cdots\times n_d}$ 为 d -阶张量，$\boldsymbol{U}\in\mathbb{R}^{L\times n_k}$ 为矩阵，则 $\mathcal{A}\times_k\boldsymbol{U}$ 定义为 $\mathcal{A}$ 与 $\boldsymbol{U}$ 在第 k 模态的乘积，其大小为 $n_1\times\cdots\times n_{k-1}\times L\times n_{k+1}\times\cdots\times n_d$，其中每一个元素的计算方法如下给出：

$$(\mathcal{A}\times_k\boldsymbol{U})_{i_1,\cdots,i_{k-1},l,i_{k+1},\cdots,i_d}=\sum_{i_k=1}^{n_k}\mathcal{A}_{i_1\cdots i_d}\boldsymbol{U}_{li_k}.$$

同时存在如下的等价关系：

$$\mathcal{B}=\mathcal{A}\times_n\boldsymbol{U}\Leftrightarrow\boldsymbol{B}_{(n)}=\boldsymbol{U}\cdot\boldsymbol{A}_{(n)}.$$

定义 2 (张量的线性变换与逆线性变换[30]). D -阶张量 $\mathcal{A}\in\mathbb{R}^{n_1\times\cdots\times n_d}$ 的变换形式定义为：

$$\mathfrak{L}(\mathcal{A})=\mathcal{A}\times_3\boldsymbol{U}_{n_3}\times_4\boldsymbol{U}_{n_4}\cdots\times_d\boldsymbol{U}_{n_d},\tag{1}$$

其中，$\mathfrak{L}$ 为广义的可逆线性变换，$\boldsymbol{U}_{n_i}\in\mathbb{C}^{n_i\times n_i}$ 为线性变换 $\mathfrak{L}$ 对应的可逆变换矩阵，其满足

$$\boldsymbol{U}_{n_i}\cdot\boldsymbol{U}_{n_i}^H=\boldsymbol{U}_{n_i}^H\cdot\boldsymbol{U}_{n_i}=\alpha_i\boldsymbol{I}_{n_i},\forall i\in\{3,\cdots,d\},\tag{2}$$

其中，$\alpha_i>0$ 是一个常量。$\mathfrak{L}(\mathcal{A})$ 的逆算子定义为：$\mathfrak{L}^{-1}(\mathcal{A})=\mathcal{A}\times_d\boldsymbol{U}_{n_d}^{-1}\times_{d-1}\boldsymbol{U}_{nd-1}^{-1}\cdots\times_3\boldsymbol{U}_{n_3}^{-1}$，$\mathfrak{L}^{-1}(\mathcal{A}_{\mathfrak{L}})=\mathcal{A}$。

定义 3 (张量 – 张量乘法[30]). 令 $\mathfrak{L}$ 为任意的可逆线性变换，对于任意 $\mathcal{A}\in\mathbb{R}^{n_1\times l\times\cdots\times n_d}$ 和 $\mathcal{B}\in\mathbb{R}^{l\times n_2\times\cdots\times n_d}$，则基于线性变换 $\mathfrak{L}$ 的 d -阶张量乘积 $\mathcal{A}*_{\mathfrak{L}}\mathcal{B}$ 定义为：

$$\mathcal{C}=\mathcal{A}*_{\mathfrak{L}}\mathcal{B}=\mathfrak{L}^{-1}(\mathcal{A}_{\mathfrak{L}}\Delta\mathcal{B}_{\mathfrak{L}}),\tag{3}$$

定理 1 ([31]，定理 3.1). 假设 $\tau\geq 1$，$a>0$，$a'>0$。令 $\boldsymbol{\Omega}\in\mathbb{R}^{I\times J}$，其中 $J>\left(1+1/\ln(I)\right)I$。集合 $\mathbb{A}(\tau,a,a',I,J)$ 包含所有大小为 $I\times J$ 的随机矩阵 $\boldsymbol{\Omega}=(\omega_{ij})$，其中的所有元素均为独立同分布的实值随机变量，并满足如下条件：

(a) 矩：$\mathbf{E}\left(|\omega_{ij}|^3\right)\leq\tau^3$；

(b) 范数：$\mathbf{P}\left(\|\boldsymbol{\Omega}\|_2>a\sqrt{J}\right)\leq\exp(-a'J)$；

(c) 方差：$\mathbf{E}\left(|\omega_{ij}|^2\right)\leq 1$。

那么，存在正常数 b、b' 使得

$$\mathbf{P}\left(\sigma_I(\mathbf{\Omega}) \le b\sqrt{J}\right) \le \exp\left(-b'J\right).$$

注 1. 假设 $\mathbf{\Omega} \in \mathbb{R}^{I\times J}$ 是随机次高斯矩阵，并且 $I \ge J$，$\tau \ge 1$，$a' > 0$。那么有 $\mathbf{P}\left(\|\mathbf{\Omega}\|_2 > a\sqrt{J}\right) \le \exp\left(-a'J\right)$，其中 $a = 6\tau\sqrt{a'+4}$。

3. 随机化张量 LU 分解算法

基于随机素描技术与张量–张量乘积框架，本文研究了若干针对高阶张量 LU 分解的单通随机算法。这类算法仅需对原始张量进行一次访问遍历，因此非常适用于存储在核心内存之外、或以流方式生成的大规模高阶张量。

3.1. 基本的随机化张量 LU 分解

用于高阶张量 LU 分解的基本随机算法被呈现于算法 1。具体而言，随机化张量 LU 分解(Randomized Tensor LU, R-TLU)的计算可以细分为如下的两个子步骤：

步骤 I (随机步骤)：借助于高斯随机投影技术，R-TLU 产生一个下三角张量 $\mathcal{L}_{y_1}$。该张量与其伪逆配合使用，以生成张量 $\mathcal{A} \in \mathbb{R}^{N_1\times N_2\times N_3\times\cdots\times N_d}$ 的正交投影(经置换后的形式)。该过程类似于随机化高阶 T-SVD [22]的随机步骤：

$$\mathcal{P} *_{\mathfrak{L}} \mathcal{A} \approx \mathcal{L}_{y_1} *_{\mathfrak{L}} \left(\mathcal{L}_{y_1}\right)^{\dagger} *_{\mathfrak{L}} \mathcal{P} *_{\mathfrak{L}} \mathcal{A}. \tag{4}$$

步骤 II (确定性步骤)：对降维的张量 $\mathcal{B} = \left(\mathcal{L}_y\right)^{\dagger} *_{\mathfrak{L}} \mathcal{P} *_{\mathfrak{L}} \mathcal{A}$ 进行张量部分主元 LU 分解，即 $\mathcal{B} *_{\mathfrak{L}} \mathcal{Q} = \mathcal{L}_b *_{\mathfrak{L}} \mathcal{U}_b$。

将随机阶段与确定性阶段合并可得：

$$\mathcal{P} *_{\mathfrak{L}} \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} \approx \mathcal{L} *_{\mathfrak{L}} \mathcal{U} \ \left(\mathcal{L} \coloneqq \mathcal{L}_{y_1} *_{\mathfrak{L}} \mathcal{L}_b, \mathcal{U} \coloneqq \mathcal{U}_b\right), \tag{5}$$

其中，$\mathcal{P}$ 和 $\mathcal{Q}$ 为置换张量，$\mathcal{L}$ 和 $\mathcal{U}$ 分别为下三角和上三角张量。

算法 1. 随机化张量 LU 分解，简称为 RTLU

输入： $\mathcal{A} \in \mathbb{R}^{N_1\times\cdots\times N_d}$，可逆线性变换：$\mathfrak{L}$，T-SVD 秩：$R$，素描尺度：$L > R$。
输出： $\mathcal{P},\mathcal{Q},\mathcal{L},\mathcal{U}$ 使得 $\mathcal{P} *_{\mathfrak{L}} \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} \approx \mathcal{L} *_{\mathfrak{L}} \mathcal{U}$。

1. 计算线性变换 $\mathfrak{L}$ 作用于 $\mathcal{A}$ 得到的结果：$\mathcal{A}_{\mathfrak{L}} \leftarrow \mathfrak{L}(\mathcal{A})$；
2. **for** $k = 1,2,\cdots,N_3N_4\cdots N_d$ **do**
3. 创建随机素描矩阵：$\mathbf{G}_{1k} \in \mathbb{R}^{N_2\times L}$；
4. $\mathbf{Y}_{1k} \leftarrow \mathcal{A}_{\mathfrak{L}}(:,:,k)\cdot\mathbf{G}_{1k}$；
5. 应用 RRLU 分解到 $\mathbf{Y}_{1k}$ 中使得 $\mathcal{P}_{\mathfrak{L}}(:,:,k)\cdot\mathbf{Y}_{1k}\cdot\left(\mathcal{Q}_y\right)_{\mathfrak{L}}(:,:,k) = \left(\mathcal{L}_{y_1}\right)_{\mathfrak{L}}(:,:,k)\cdot\left(\mathcal{U}_{y_1}\right)_{\mathfrak{L}}(:,:,k)$；
6. $\mathbf{B} \leftarrow \left(\left(\mathcal{L}_{y_1}\right)_{\mathfrak{L}}(:,1:R,k)\right)^{\dagger}\cdot\mathcal{P}_{\mathfrak{L}}(:,:,k)\cdot\mathcal{A}_{\mathfrak{L}}(:,:,k)$；
7. 应用列旋转 LU 分解到 $\mathbf{B}$ 使得 $\mathbf{B}\cdot\mathcal{Q}(:,:,k) = \left(\mathcal{L}_b\right)_{\mathfrak{L}}(:,:,k)\cdot\left(\mathcal{U}_b\right)_{\mathfrak{L}}(:,:,k)$；
8. $\mathcal{L}(:,:,k) \leftarrow \left(\mathcal{L}_{y_1}\right)_{\mathfrak{L}}(:,:,k)\cdot\left(\mathcal{L}_b\right)_{\mathfrak{L}}(:,:,k)$；
9. $\mathcal{U}(:,:,k) \leftarrow \left(\mathcal{U}_b\right)_{\mathfrak{L}}(:,:,k)$；
10. **end**
11. $\mathcal{P} \leftarrow \mathfrak{L}^{-1}\left(\mathcal{P}_{\mathfrak{L}}\right)$，$\mathcal{Q} \leftarrow \mathfrak{L}^{-1}\left(\mathcal{Q}_{\mathfrak{L}}\right)$，$\mathcal{L} \leftarrow \mathfrak{L}^{-1}\left(\mathcal{L}_{\mathfrak{L}}\right)$，$\mathcal{U} \leftarrow \mathfrak{L}^{-1}\left(\mathcal{U}_{\mathfrak{L}}\right)$。

3.2. 单通随机化张量 LU 分解算法

在本节中，我们首先设计一种新型高效的单通随机张量 LU 分解算法(简称 SPRTLU，详见算法 2)，同时给出其对应的误差上界理论与复杂度分析。

算法 2. 单通随机化张量 LU 分解，简称为 SPRTLU

输入： $\mathcal{A}\in\mathbb{R}^{N_1\times\cdots\times N_d}$，可逆的线性变换：$\mathfrak{L}$，目标 T-SVD 秩：$R$，素描大小：$L$。
输出： $\mathcal{P},\mathcal{Q},\mathcal{L},\mathcal{U}$ 使得 $\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}\approx\mathcal{L}*_{\mathfrak{L}}\mathcal{U}$。

1. 对 $\mathcal{A}$ 做可逆线性变换域 $\mathfrak{L}$：$\mathcal{A}_{\mathfrak{L}}\leftarrow\mathfrak{L}(\mathcal{A})$；
2. **for** $k=1,2,\cdots,N_3N_4\cdots N_d$ **do**
3. 生成两个随机矩阵：$\boldsymbol{G}_{1k}\in\mathbb{R}^{N_2\times R}$，$\boldsymbol{G}_{2k}\in\mathbb{R}^{L\times N_1}$；
4. $\boldsymbol{Y}_{1k}\leftarrow\mathcal{A}_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{G}_{1k}$，$\boldsymbol{Y}_{2k}\leftarrow\boldsymbol{G}_{2k}\cdot\mathcal{A}_{\mathfrak{L}}(:,:,k)$；
5. 对 $\boldsymbol{Y}_{1k}$ 做 RRLU 分解使得 $\mathcal{P}_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{Y}_{1k}\cdot(\mathcal{Q}_y)_{\mathfrak{L}}(:,:,k)=(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_{y_1})_{\mathfrak{L}}(:,:,k)$；
6. $\boldsymbol{B}\leftarrow\left(\boldsymbol{G}_{2k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\right)^{\dagger}\cdot\boldsymbol{Y}_{2k}$；
7. 对 $\boldsymbol{B}$ 作列旋转 LU 分解使得 $\boldsymbol{B}\cdot\mathcal{Q}(:,:,k)=(\mathcal{L}_b)_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_b)_{\mathfrak{L}}(:,:,k)$；
8. $\mathcal{L}(:,:,k)\leftarrow(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{L}_b)_{\mathfrak{L}}(:,:,k)$；
9. $\mathcal{U}(:,:,k)\leftarrow(\mathcal{U}_b)_{\mathfrak{L}}(:,:,k)$；
10. **end**
11. $\mathcal{P}\leftarrow\mathfrak{L}^{-1}(\mathcal{P}_{\mathfrak{L}})$，$\mathcal{Q}\leftarrow\mathfrak{L}^{-1}(\mathcal{Q}_{\mathfrak{L}})$，$\mathcal{L}\leftarrow\mathfrak{L}^{-1}(\mathcal{L}_{\mathfrak{L}})$，$\mathcal{U}\leftarrow\mathfrak{L}^{-1}(\mathcal{U}_{\mathfrak{L}})$。

3.2.1. SPRTLU 算法的描述与理论结果

SPRTLU 的主要思想：如何利用已有较小规模的张量替换算法 1 中 $\mathcal{B}=\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}$ 的 $\mathcal{A}$。首先，在 $\mathcal{B}=\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}$ 两边同时左乘 $\mathcal{L}_{y_1}$ 得到

$$\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{B}=\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A},$$

该式子与 $\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\approx\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}$ 联立得出

$$\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{B}\approx\mathcal{P}*_{\mathfrak{L}}\mathcal{A}.$$

值得注意的是 $\mathcal{Y}_{2k}=\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{A}$。在 $\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{B}\approx\mathcal{P}*_{\mathfrak{L}}\mathcal{A}$ 两边同时乘以 $\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}$，我们进一步得到：

$$\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{B}\approx\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{A}=\mathcal{Y}_{2k}.$$

最终，我们得到 $\mathcal{B}$ 的一个近似形式：

$$\mathcal{B}\approx\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{Y}_{2k}. \tag{6}$$

根据上述推导，我们可以发现：与随机化 RTLU 算法相比，SPRTLU 算法有一个由 $\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\approx\mathcal{L}_y*_{\mathfrak{L}}\left(\mathcal{L}_y\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}$ 额外产生的误差。因此，算法 2 的误差通常会比算法 1 的误差大。具体地，考虑如下的结果：

$$
\begin{aligned}
&\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}*_{\mathfrak{L}}\mathcal{U}\right\|_F\\
&=\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{L}_b*_{\mathfrak{L}}\mathcal{U}_b\right\|_F\\
&=\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{B}*_{\mathfrak{L}}\mathcal{Q}\right\|_F\\
&=\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{Y}_{2k}\right\|_F\\
&\le\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\right\|_F\\
&\quad+\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{Y}_{2k}\right\|_F.
\end{aligned}
\tag{7}
$$

基于上述结果，我们给出算法 2 的误差界理论，其具体证明在下一节详细给出。

定理 2. 令 $\mathcal{A}$ 是大小为 $N_1\times N_2\times\cdots\times N_d$ ($N_1\ge N_2$)的实值张量， R 是预期的 T-SVD 秩，$L>\left(1+\dfrac{1}{\ln R}\right)R$。对任意的 k，令 $t_k=\left(\dfrac{2}{\Delta_k}\right)$，$\mu_k=\sqrt{2\log\dfrac{2}{\Delta_k}}$，$\Theta_{\Delta_k}=t_k e\sqrt{R}\left(\sqrt{N_2-R}+\sqrt{R}\right)+t_k e\sqrt{R}\mu_k$，其中 $0<\Delta_k\ll 1$。那么由算法 2 产生的随机张量 LU 分解满足：

$$
\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}*_{\mathfrak{L}}\mathcal{U}\right\|_F\le\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}\left(1+\frac{a_1\sqrt{N_1}}{c_1\sqrt{L}}\right)\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left(\Theta_{\Delta_k}\right)^2}{\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left(\Theta_{\Delta_k}\right)^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2},
\tag{8}
$$

以不小于 $1-\mathrm{e}^{-a_2N_1}-\mathrm{e}^{-c_2L}-\sum_{k=1}^{N_3\cdots N_d}\Delta_k$ 的概率成立，其中参数 a_1,c_1,a_2,c_2 按定理 1 确定，$\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2=\sum_{i=R+1}^{\min(N_1,N_2)}\left(\hat{\sigma}_i^{\langle k\rangle}\right)^2$，$\hat{\sigma}_i^{\langle k\rangle}=\left(\mathcal{S}_{\mathfrak{L}}\right)^{\langle k\rangle}(i,i)$，$\mathcal{S}$ 来自 $\mathcal{A}=\mathcal{U}*_{\mathfrak{L}}\mathcal{S}*_{\mathfrak{L}}\mathcal{V}^{\mathrm{T}}$，$\mathcal{A}_R$ 是 T-SVD 框架下的秩-R 最佳逼近。

3.2.2. SPRTLU 算法的复杂度分析

给定输入张量为 $\mathcal{A}\in\mathbb{R}^{N_1\times\cdots\times N_d}$，算法 2 的复杂度如下：1) 生成 $N_2\times R$，$L\times N_1$ 的随机矩阵，需要 $\mathcal{O}\left(N_2R+N_1L\right)$ 运算；2) 计算矩阵 $\boldsymbol{Y}_{1k},\boldsymbol{Y}_{2k}$ 需要 $\mathcal{O}\left((R+L)N_1N_2\right)$ 次运算；3) 对矩阵 $\boldsymbol{Y}_{1k}$ 做 RRLU 分解需要 $\mathcal{O}\left(N_1R^2\right)$ 次运算；4) 计算 $\boldsymbol{G}_{2k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot\left(\mathcal{L}_{y_1}\right)_{\mathfrak{L}}(:,:,k)$ 需要 $\mathcal{O}\left(N_1RL\right)$ 次运算，计算 $\boldsymbol{G}_{2k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot\left(\mathcal{L}_{y_1}\right)_{\mathfrak{L}}(:,:,k)$ 的穆尔－彭罗斯广义逆需要 $\mathcal{O}\left(R^2L+R^3+R^2L\right)$ 次运算，并乘以矩阵 $\boldsymbol{Y}_{2k}$ 需要 $\mathcal{O}\left(N_2RL\right)$ 次运算；5) 对 $\boldsymbol{B}$ 做列旋转的 LU 分解需要 $\mathcal{O}\left(N_2R^2\right)$ 次运算；6) 计算 L 需要 $\mathcal{O}\left(N_1R^2\right)$ 次运算。重复上述步骤 $N_3N_4\cdots N_d$ 次，我们得到算法 2 内部切片运算的整体复杂度为

$$
\mathcal{C}_{SPRTLU}=\mathcal{O}\left(N_3N_4\cdots N_d\left(N_1L+N_2R+(R+L)N_1N_2+(N_1+N_2+L)R^2+(N_1+N_2)RL+R^3\right)\right).
$$

对张量 $\mathcal{A}$ 进行可逆线性变换 $\mathfrak{L}$ 或者其可逆算子 $\mathfrak{L}^{-1}$，即 $\mathfrak{L}(\mathcal{A})$ 或 $\mathfrak{L}^{-1}(\mathcal{A})$，均需要 $\mathcal{O}\left(\prod_{i=1}^{d}N_i\cdot\sum_{j=3}^{d}N_j\right)$ 次浮点运算。对于一些特殊的可逆线性变换 $\mathfrak{L}$ (如快速傅里叶变换)，计算复杂度可以降低为 $\mathcal{O}\left(\prod_{i=1}^{d}N_i\cdot\sum_{j=3}^{d}\log N_j\right)$ 次浮点运算。

3.3. 子空间单通随机化张量 LU 分解算法

在本节中，为了进一步增强近似性能，我们给出算法 2 的第一种变体(简称为 SO-SPRTLU，见算法 3)，同时推导它们的误差上界理论和复杂度。

3.3.1. SO-SPRTLU 算法的描述与理论结果

在本部分，我们提出一种高效的子空间单通随机化张量 LU 分解算法(简称 SO-SPRTLU，详见算法 3)。通过选择算法 2 中的素描矩阵 $\mathcal{G}_{2k}$ 为 $(\mathcal{Y}_{1k})^{\mathrm{T}}=(\mathcal{A}*_{\mathfrak{L}}\mathcal{G}_{1k})^{\mathrm{T}}$，我们得到算法 2 的一种变体。也就是说，对于每个 k，我们进行如下的设置：

$$\boldsymbol{Y}_{1k}\leftarrow\mathcal{A}_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{G}_{1k},\boldsymbol{Y}_{2k}\leftarrow(\boldsymbol{Y}_{1k})^{\mathrm{T}}\cdot\mathcal{A}_{\mathfrak{L}}(:,:,k),$$

其中，$\boldsymbol{G}_{1k}$ 表示高斯随机素描矩阵。

下面，我们将分析算法 3 的理论误差界，其误差上界略比算法 2 的误差上界要紧致一些。

定理 3. 沿用定理 2 的概念和假设，由算法 3 生成的随机张量 LU 分解满足：

$$\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}*_{\mathfrak{L}}\mathcal{U}\right\|_F\le\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}2\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left(\Theta_{\Delta_k}\right)^2}{\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left(\Theta_{\Delta_k}\right)^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2},\tag{9}$$

以不小于 $1-\sum_{k=1}^{N_3\cdots N_d}\Delta_k$ 的概率成立。

算法 3. 子空间单通随机张量 LU 分解，简称 SORTLU

输入： $\mathcal{A}\in\mathbb{R}^{N_1\times\cdots\times N_d}$，可逆线性变换：$\mathfrak{L}$，目标 T-SVD 秩：$R$，素描大小：$L$。
输出： $\mathcal{P},\mathcal{Q},\mathcal{L},\mathcal{U}$ 使得 $\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}\approx\mathcal{L}*_{\mathfrak{L}}\mathcal{U}$。

1. 对 $\mathcal{A}$ 作可逆线性变换 $\mathfrak{L}$：$\mathcal{A}_{\mathfrak{L}}\leftarrow\mathfrak{L}(\mathcal{A})$；
2. **for** $k=1,2,\cdots,N_3N_4\cdots N_d$ **do**
3. 生成随机矩阵：$\boldsymbol{G}_{1k}\in\mathbb{R}^{N_2\times R}$；
4. $\boldsymbol{Y}_{1k}\leftarrow\mathcal{A}_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{G}_{1k}$，$\boldsymbol{Y}_{2k}\leftarrow\boldsymbol{Y}_{1k}^{\mathrm{T}}\cdot\mathcal{A}_{\mathfrak{L}}(:,:,k)$；
5. 对 $\boldsymbol{Y}_{1k}$ 作 RRLU 分解使得 $\mathcal{P}_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{Y}_{1k}\cdot(\mathcal{Q}_y)_{\mathfrak{L}}(:,:,k)=(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_{y_1})_{\mathfrak{L}}(:,:,k)$；
6. $\boldsymbol{B}\leftarrow\left(\boldsymbol{G}_{2k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\right)^{\dagger}\cdot\boldsymbol{Y}_{2k}$；
7. 对 $\boldsymbol{B}$ 作列旋转 LU 分解使得 $\boldsymbol{B}\cdot\mathcal{Q}(:,:,k)=(\mathcal{L}_b)_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_b)_{\mathfrak{L}}(:,:,k)$；
8. $\mathcal{L}(:,:,k)\leftarrow(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{L}_b)_{\mathfrak{L}}(:,:,k)$；
9. $\mathcal{U}(:,:,k)\leftarrow(\mathcal{U}_b)_{\mathfrak{L}}(:,:,k)$；
10. **end**
11. $\mathcal{P}\leftarrow\mathfrak{L}^{-1}(\mathcal{P}_{\mathfrak{L}})$，$\mathcal{Q}\leftarrow\mathfrak{L}^{-1}(\mathcal{Q}_{\mathfrak{L}})$，$\mathcal{L}\leftarrow\mathfrak{L}^{-1}(\mathcal{L}_{\mathfrak{L}})$，$\mathcal{U}\leftarrow\mathfrak{L}^{-1}(\mathcal{U}_{\mathfrak{L}})$。

3.3.2. SO-SPRTLU 算法的复杂度分析

给定输入张量为 $\mathcal{A}\in\mathbb{R}^{N_1\times\cdots\times N_d}$，算法 3 的复杂度如下：1) 生成 $N_2\times R$ 的随机矩阵，需要 $\mathcal{O}(N_2R)$ 浮点运算；2) 计算矩阵 $\boldsymbol{Y}_{1k},\boldsymbol{Y}_{2k}$ 需要 $\mathcal{O}(RN_1N_2)$ 浮点运算；3) 对矩阵 $\boldsymbol{Y}_{1k}$ 做 RRLU 分解需要 $\mathcal{O}(N_1R^2)$ 浮点运算；4) 计算 $\boldsymbol{Y}_{1k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)$ 需要 $\mathcal{O}(N_1R^2)$ 浮点运算，计算 $\boldsymbol{Y}_{1k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)$ 的穆尔－彭罗斯广义逆需要 $\mathcal{O}(R^3)$ 浮点运算，并乘以矩阵 $\boldsymbol{Y}_{2k}$ 需要 $\mathcal{O}(N_2R^2)$ 浮点运算；5) 对 $\boldsymbol{B}$ 做列旋转的 LU 分解需要 $\mathcal{O}(N_2R^2)$ 浮点运算；6) 计算 L 需要 $\mathcal{O}(N_1R^2)$ 浮点运算。重复上述步骤 $N_3N_4\cdots N_d$ 次我们得到算法 3 内部切片运算的整体复杂度为

$$\mathcal{C}_{SORTLU}=\mathcal{O}\left(N_3N_4\cdots N_d\left((N_1+N_2)R^2+N_1N_2R+N_2R+R^3\right)\right).$$

对张量 $\mathcal{A}$ 进行可逆线性变换 $\mathfrak{L}$ 或者其可逆算子 $\mathfrak{L}^{-1}$，即 $\mathfrak{L}(\mathcal{A})$ 或 $\mathfrak{L}^{-1}(\mathcal{A})$，均需要 $\mathcal{O}\left(\prod_{i=1}^{d} N_i \cdot \sum_{j=3}^{d} N_j\right)$ 次浮点运算。对于一些特殊的可逆线性变换 $\mathfrak{L}$ (如快速傅里叶变换)，计算复杂度可以降低为 $\mathcal{O}\left(\prod_{i=1}^{d} N_i \cdot \sum_{j=3}^{d} \log N_j\right)$ 次浮点运算。

3.4. 双边随机 LU 分解

3.4.1. TSRTLU 算法的描述与理论结果

为了推导算法 2 中的 $\mathcal{B}$，我们使用了如下的结果：$\mathcal{P} *_{\mathfrak{L}} \mathcal{A} \approx \mathcal{L}_{y_1} *_{\mathfrak{L}} \left(\mathcal{L}_{y_1}\right)^{\dagger} *_{\mathfrak{L}} \mathcal{P} *_{\mathfrak{L}} \mathcal{A}$。类似地，对另一边使用 $\mathcal{A} *_{\mathfrak{L}} \mathcal{Q} \approx \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} *_{\mathfrak{L}} \left(\mathcal{U}_{y_2}\right)^{\dagger} *_{\mathfrak{L}} \left(\mathcal{U}_{y_2}\right)$，其中 $\mathcal{L}_{y_2} *_{\mathfrak{L}} \mathcal{U}_{y_2}$ 是对 $\mathcal{P}_2 *_{\mathfrak{L}} \mathcal{Y}_{2k} *_{\mathfrak{L}} \mathcal{Q}$ 做 LU 分解后的结果。因此，类似于公式(6)的推导一样，我们得到

$$\mathcal{B} = \left(\mathcal{G}_2 *_{\mathcal{L}} \mathcal{P}^{\mathrm{T}} *_{\mathcal{L}} \mathcal{L}_{y_1}\right)^{\dagger} *_{\mathcal{L}} \mathcal{Y}_2 *_{\mathfrak{L}} \mathcal{Q} *_{\mathcal{L}} \mathcal{U}_{y_2}^{\dagger}.$$

使用上述推导关于 $\mathcal{B}$ 的公式，得到新型的双边随机张量 LU 算法 4。

由于使用了双边投影：$\mathcal{A} *_{\mathfrak{L}} \mathcal{Q} \approx \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} *_{\mathfrak{L}} \left(\mathcal{U}_{y_2}\right)^{\dagger} *_{\mathfrak{L}} \left(\mathcal{U}_{y_2}\right)$ 与 $\mathcal{P} *_{\mathfrak{L}} \mathcal{A} \approx \mathcal{L}_{y_1} *_{\mathfrak{L}} \left(\mathcal{L}_{y_1}\right)^{\dagger} *_{\mathfrak{L}} \mathcal{P} *_{\mathfrak{L}} \mathcal{A}$，因此算法 4 产生了额外的误差。具体地，我们有如下：

$$\begin{aligned}
&\left\|\mathcal{P} *_{\mathfrak{L}} \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} - \mathcal{L} *_{\mathfrak{L}} \mathcal{U}\right\|_F \\
&= \left\|\mathcal{P} *_{\mathfrak{L}} \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} - \mathcal{L}_{y_1} *_{\mathfrak{L}} \mathcal{L}_b *_{\mathfrak{L}} \mathcal{U}_b *_{\mathfrak{L}} \mathcal{U}_{y2}\right\|_F \\
&= \left\|\mathcal{P} *_{\mathfrak{L}} \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} - \mathcal{L}_{y_1} *_{\mathfrak{L}} \left(\mathcal{G}_2 *_{\mathcal{L}} \mathcal{P}^{\mathrm{T}} *_{\mathcal{L}} \mathcal{L}_{y_1}\right)^{\dagger} *_{\mathcal{L}} \mathcal{Y}_2 *_{\mathfrak{L}} \mathcal{Q} *_{\mathcal{L}} \mathcal{U}_{y_2}^{\dagger} *_{\mathcal{L}} \mathcal{U}_{y_2}\right\|_F \\
&\le \left\|\mathcal{P} *_{\mathcal{L}} \mathcal{A} - \mathcal{L}_{y_1} *_{\mathcal{L}} \mathcal{L}_{y_1}^{\dagger} *_{\mathcal{L}} \mathcal{P} *_{\mathcal{L}} \mathcal{A}\right\|_F \\
&\quad + \left\|\mathcal{L}_{y_1} *_{\mathcal{L}} \mathcal{L}_{y_1}^{\dagger} *_{\mathcal{L}} \mathcal{P} *_{\mathcal{L}} \mathcal{A} *_{\mathcal{L}} \mathcal{Q} - \mathcal{L}_{y_1} *_{\mathfrak{L}} \left(\mathcal{G}_2 *_{\mathcal{L}} \mathcal{P}^{\mathrm{T}} *_{\mathcal{L}} \mathcal{L}_{y_1}\right)^{\dagger} *_{\mathcal{L}} \mathcal{Y}_2 *_{\mathfrak{L}} \mathcal{Q} *_{\mathcal{L}} \mathcal{U}_{y_2}^{\dagger} *_{\mathcal{L}} \mathcal{U}_{y_2}\right\|_F
\end{aligned} \tag{10}$$

相比于式子(21)，该式子引入了额外的误差项。基于这一结果，我们得到算法 4 的误差界理论，详见定理 4。

定理 4. 令 $\mathcal{A}$ 是大小为 $N_1 \times N_2 \times \cdots \times N_d$ $(N_1 \ge N_2)$的实值张量，R 是预期的 T-SVD 秩，$L > \left(1 + \frac{1}{\ln R}\right) R$。对任意的 k，令 $t_{1k} = \frac{2}{\Delta_{1k}}$，$t_{2k} = \left(\frac{2}{\Delta_{2k}}\right)^{\frac{1}{p+1}}$，$\mu_{1k} = \sqrt{2\log\frac{2}{\Delta_{1k}}}$，$\mu_{2k} = \sqrt{2\log\frac{2}{\Delta_{2k}}}$，$\Theta_{\Delta_{1k}} = t_{1k} e\sqrt{R}\left(\sqrt{N_2 - R} + \sqrt{R}\right) + t_{1k} e\sqrt{R}$，$\Theta_{\Delta_{2k}} = t_{2k} e\sqrt{R}\left(\sqrt{N_2 - R} + \sqrt{R}\right) + t_{2k} e\sqrt{R}$，其中 $0 < \Delta_{1k} \ll 1$，$0 < \Delta_{2k} \ll 1$。那么由算法 4 产生的随机张量 LU 分解满足：

$$\begin{aligned}
&\left\|\mathcal{P} *_{\mathfrak{L}} \mathcal{A} *_{\mathfrak{L}} \mathcal{Q} - \mathcal{L} *_{\mathfrak{L}} \mathcal{U}\right\|_F \\
&\le \frac{1}{\rho} \sum_{k=1}^{N_3 N_4 \cdots N_d} \left(\left(1 + \frac{a_1\sqrt{N_1}}{c_1\sqrt{L}}\right) \sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2 \left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2 \left(\Theta_{\Delta_{1k}}\right)^2}{\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2 \left(\Theta_{\Delta_{1k}}\right)^2 + \left(\hat{\sigma}_1^{\langle k\rangle}\right)^2} + \left\|\mathcal{A} - \mathcal{A}_R\right\|_F^2} \right. \\
&\quad \left. + \frac{a_1\sqrt{N_1}}{c_1\sqrt{L}} \sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2 \left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2 \left(\Theta_{\Delta_{2k}}\right)^2}{\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2 \left(\Theta_{\Delta_{2k}}\right)^2 + \left(\hat{\sigma}_1^{\langle k\rangle}\right)^2} + \left\|\mathcal{A} - \mathcal{A}_R\right\|_F^2} \right),
\end{aligned} \tag{11}$$

以不小于 $1 - \mathrm{e}^{-a_2 N_1} - \mathrm{e}^{-c_2 L} - \sum_{k=1}^{N_3 \cdots N_d} \Delta_k$ 的概率成立，其中 a_1, c_1, a_2, c_2 是由定理 1 确定，其他符号的定义与定

理 2 是类似的。

算法 4. 双边随机张量 LU 分解，简称 TSRTLU

输入： $\mathcal{A}\in\mathbb{R}^{N_1\times\cdots\times N_d}$，可逆线性变换：$\mathfrak{L}$，目标 T-SVD 秩：$R$，素描大小：$L$。
输出： $\mathcal{P},\mathcal{Q},\mathcal{L},\mathcal{U}$ 使得 $\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}\approx\mathcal{L}*_{\mathfrak{L}}\mathcal{U}$。

1. 对张量 $\mathcal{A}$ 进行可逆线性变换 $\mathfrak{L}$：$\mathcal{A}_{\mathfrak{L}}\leftarrow\mathfrak{L}(\mathcal{A})$；
2. **for** $k=1,2,\cdots,N_3N_4\cdots N_d$ **do**
3. 生成两个随机矩阵：$\boldsymbol{G}_{1k}\in\mathbb{R}^{N_2\times R}$，$\boldsymbol{G}_{2k}\in\mathbb{R}^{L\times N_1}$；
4. $\boldsymbol{Y}_{1k}\leftarrow\mathcal{A}_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{G}_{1k}$，$\boldsymbol{Y}_{2k}\leftarrow\boldsymbol{G}_{2k}\cdot\mathcal{A}_{\mathfrak{L}}(:,:,k)$；
5. 分别对 $\boldsymbol{Y}_{1k}$ 与 $\boldsymbol{Y}_{2k}$ 作 RRLU 分解使得

$$\mathcal{P}_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{Y}_{1k}\cdot(\mathcal{Q}_{y})_{\mathfrak{L}}(:,:,k)=(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_{y_1})_{\mathfrak{L}}(:,:,k);$$

$$(\mathcal{P}_{2})_{\mathfrak{L}}(:,:,k)\cdot\boldsymbol{Y}_{2k}\cdot(\mathcal{Q})_{\mathfrak{L}}(:,:,k)=(\mathcal{L}_{y_2})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_{y_2})_{\mathfrak{L}}(:,:,k);$$

6. $\boldsymbol{B}\leftarrow\left(\boldsymbol{G}_{2k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\right)^{\dagger}\cdot\boldsymbol{Y}_{2k}\cdot\mathcal{Q}_{\mathfrak{L}}(:,:,k)\cdot\left[(\mathcal{U}_{y_2})_{\mathfrak{L}}(:,:,k)\right]^{\dagger}$；
7. 对 $\boldsymbol{B}$ 作列旋转 LU 分解使得 $\boldsymbol{B}\cdot\mathcal{Q}(:,:,k)=(\mathcal{L}_{b})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_{b})_{\mathfrak{L}}(:,:,k)$；
8. $\mathcal{L}(:,:,k)\leftarrow(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{L}_{b})_{\mathfrak{L}}(:,:,k)$；
9. $\mathcal{U}(:,:,k)\leftarrow(\mathcal{U}_{b})_{\mathfrak{L}}(:,:,k)\cdot(\mathcal{U}_{y_2})_{\mathfrak{L}}(:,:,k)$；
10. **end**
11. $\mathcal{P}\leftarrow\mathfrak{L}^{-1}(\mathcal{P}_{\mathfrak{L}})$，$\mathcal{Q}\leftarrow\mathfrak{L}^{-1}(\mathcal{Q}_{\mathfrak{L}})$，$\mathcal{L}\leftarrow\mathfrak{L}^{-1}(\mathcal{L}_{\mathfrak{L}})$，$\mathcal{U}\leftarrow\mathfrak{L}^{-1}(\mathcal{U}_{\mathfrak{L}})$。

3.4.2. TSRTLU 算法的复杂度分析

给定输入张量为 $\mathcal{A}\in\mathbb{R}^{N_1\times\cdots\times N_d}$，算法 4 的复杂度如下：1) 生成 $N_2\times R$，$L\times N_1$ 的随机矩阵，需要 $\mathcal{O}(N_2R+N_1L)$ 浮点运算；2) 计算矩阵 $\boldsymbol{Y}_{1k},\boldsymbol{Y}_{2k}$ 需要 $\mathcal{O}((R+L)N_1N_2)$ 浮点运算；3) 对矩阵 $\boldsymbol{Y}_{1k},\boldsymbol{Y}_{2k}$ 做 RRLU 分解需要 $\mathcal{O}(N_1R^2+N_2L^2)$ 浮点运算；4) 计算 $\boldsymbol{G}_{2k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)$ 需要 $\mathcal{O}(N_1RL)$ 浮点运算，计算 $\boldsymbol{G}_{2k}\cdot\mathcal{P}_{\mathfrak{L}}^{\mathrm{T}}(:,:,k)\cdot(\mathcal{L}_{y_1})_{\mathfrak{L}}(:,:,k)$ 的穆尔－彭罗斯广义逆需要 $\mathcal{O}(R^2L+R^3+R^2L)$ 浮点运算，并乘以矩阵 $\boldsymbol{Y}_{2k}(\mathcal{Q})_{\mathcal{L}}(:,:,k)$ 需要 $\mathcal{O}(N_2RL)$ 浮点运算，计算 $(\mathcal{U}_{y2})_{\mathcal{L}}(:,:,k)$ 的穆尔－彭罗斯广义逆需要 $\mathcal{O}(L^2N_2+L^3+L^2N_2)$ 浮点运算，然后乘以矩阵 $\left((\mathcal{U}_{y2})_{\mathcal{L}}(:,:,k)\right)^{\dagger}$ 需要 $\mathcal{O}(N_2RL)$ 浮点运算；5) 对 $\boldsymbol{B}$ 做列旋转的 LU 分解需要 $\mathcal{O}(R^2L)$ 浮点运算；6) 计算 $(\mathcal{L}_{y1})_{\mathcal{L}}(:,:,k)(\mathcal{L}_{b})_{\mathcal{L}}(:,:,k)$ 需要 $\mathcal{O}(N_1R^2)$ 浮点运算；7) 计算 $(\mathcal{U}_{b})_{\mathcal{L}}(:,:,k)(\mathcal{U}_{y2})_{\mathcal{L}}(:,:,k)$ 需要 $\mathcal{O}(N_2RL)$ 浮点运算。重复上述步骤 $N_3N_4\cdots N_d$ 次，我们得到算法 4 内部切片运算的整体复杂度为

$$\mathcal{C}_{TSRTLU}=\mathcal{O}\left(N_3N_4\cdots N_d\left(N_1N_2(R+L)+N_1L(R+1)+N_2R(L+1)+N_1R^2+N_2L^2+LR^2+R^3\right)\right).$$

对张量 $\mathcal{A}$ 进行可逆线性变换 $\mathfrak{L}$ 或者其可逆算子 $\mathfrak{L}^{-1}$，即 $\mathfrak{L}(\mathcal{A})$ 或 $\mathfrak{L}^{-1}(\mathcal{A})$，均需要 $\mathcal{O}\left(\prod_{i=1}^{d}N_i\cdot\sum_{j=3}^{d}N_j\right)$ 次浮点运算。对于一些特殊的可逆线性变换 $\mathfrak{L}$ (如快速傅里叶变换)，计算复杂度可以降低为 $\mathcal{O}\left(\prod_{i=1}^{d}N_i\cdot\sum_{j=3}^{d}\log N_j\right)$ 次浮点运算。

4. 算法的误差界理论证明

在给出定理 2、定理 3 以及定理 4 的证明之前，我们首先导入一些重要的引理和命题。

设 $\mathcal{A}=\mathcal{U}*_{\mathfrak{L}}\mathcal{S}*_{\mathfrak{L}}\mathcal{V}^{\mathrm{T}}$ 是大小为 $N_1\times N_2\times\cdots\times N_d$ 的张量 $\mathcal{A}$ 的 T-SVD，$\mathcal{G}$ 是大小为 $N_2\times L\times\cdots\times N_d$ 的随机张量[1]。设 P 是一个整数且 $0\le P\le L-R$，其中 R 是目标秩。对于所有的 k，将 $\left(\mathcal{A}_{\mathfrak{L}}\right)^{\langle k\rangle}$ 进行划分

$$\left(\mathcal{A}_{\mathfrak{L}}\right)^{\langle k\rangle}=\begin{bmatrix}\boldsymbol{U}_{1k} & \boldsymbol{U}_{2k}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_{1k} & & \\ & \boldsymbol{\Sigma}_{2k} & \\ & & \boldsymbol{\Sigma}_{3k}\end{bmatrix}\begin{bmatrix}\boldsymbol{V}_{1k}^{H}\\ \boldsymbol{V}_{2k}^{H}\end{bmatrix}, \tag{12}$$

$\boldsymbol{U}_{1k}=\left(\mathcal{U}_{\mathfrak{L}}\right)^{\langle k\rangle}(:,1:L-P)$，$\boldsymbol{U}_{2k}=\left(\mathcal{U}_{\mathfrak{L}}\right)^{\langle k\rangle}(:,L-P+1:N_1)$，$\boldsymbol{V}_{1k}=\left(\mathcal{V}_{\mathfrak{L}}\right)^{\langle k\rangle}(:,1:L-P)$，

$\boldsymbol{V}_{2k}=\left(\mathcal{V}_{\mathfrak{L}}\right)^{\langle k\rangle}(:,L-P+1:N_2)$，$\boldsymbol{\Sigma}_{1k}=\left(\mathcal{S}_{\mathfrak{L}}\right)^{\langle k\rangle}(1:R,1:R)$，$\boldsymbol{\Sigma}_{2k}=\left(\mathcal{S}_{\mathfrak{L}}\right)^{\langle k\rangle}(R+1:L-P,R+1:L-P)$，

$\boldsymbol{\Sigma}_{3k}=\left(\mathcal{S}_{\mathfrak{L}}\right)^{\langle k\rangle}(L-P+1:N_1,L-P+1:N_2)$。对于所有的 k，分别定义 $\boldsymbol{\Omega}_{1k}\in\mathbb{C}^{L-P\times L}$ 与 $\boldsymbol{\Omega}_{2k}\in\mathbb{C}^{N_2-L+P\times L}$ 为如下：

$$\boldsymbol{\Omega}_{1k}=\boldsymbol{V}_{1k}^{H}\boldsymbol{G},\boldsymbol{\Omega}_{2k}=\boldsymbol{V}_{2k}^{H}\boldsymbol{G}, \tag{13}$$

其中，$\boldsymbol{G}$ 表示 $\mathcal{G}$ 在变换域 $\mathfrak{L}$ 中的正面切片，即 $\boldsymbol{G}:=\left(\mathcal{G}_{\mathfrak{L}}\right)^{\langle k\rangle}$。

命题 4.1. 设 $\mathcal{A}=\mathcal{U}*_{\mathfrak{L}}\mathcal{S}*_{\mathfrak{L}}\mathcal{V}^{\mathrm{T}}$ 是大小为 $N_1\times N_2\times\cdots\times N_d$ 的张量 $\mathcal{A}$ 的满 T-SVD，$\mathcal{G}$ 是大小为 $N_2\times L\times\cdots\times N_d$ 的高斯随机张量，$P+R\le L$，$\mathcal{Q}*_{\mathfrak{L}}\mathcal{R}$ 是 $\mathcal{A}*_{\mathfrak{L}}\mathcal{G}$ 的瘦小版 T-QR 分解。对每一个 k，假设 $\boldsymbol{\Omega}_{1k}$ 是行满秩。则我们有如下：

$$\left\|\left(\mathcal{I}-\mathcal{Q}*_{\mathfrak{L}}\mathcal{Q}^{\mathrm{T}}\right)*_{\mathfrak{L}}\mathcal{A}\right\|_F\le\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{L-P+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2}{\left(\hat{\sigma}_{L-P+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2}, \tag{14}$$

其中，$\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2=\sum_{i=R+1}^{\min(N_1,N_2)}\left(\hat{\sigma}_i^{\langle k\rangle}\right)^2$，$\hat{\sigma}_i^{\langle k\rangle}=\left(\mathcal{S}_{\mathfrak{L}}\right)^{\langle k\rangle}(i,i)$，$\mathcal{A}_R$ 是 T-SVD 框架下的秩-R 最佳逼近。

值得注意的是，如果算法 2 中 $\mathcal{P}*_{\mathfrak{L}}\mathcal{Y}_{1k}*_{\mathfrak{L}}\mathcal{Q}_y$ 的瘦小版 T-QR 分解为 $\mathcal{P}*_{\mathfrak{L}}\mathcal{Y}_{1k}*_{\mathfrak{L}}\mathcal{Q}_y=\mathcal{Q}*_{\mathfrak{L}}\mathcal{R}$，则 $\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{L}_{y_1}^{\dagger}=\mathcal{Q}*_{\mathfrak{L}}\mathcal{Q}^{\mathrm{T}}$。因此，设命题 4.1 中的 $\mathcal{G}$ 是算法 2 中的 $\mathcal{G}_{1k}$，并设置算法 2 中的 P 为零，得到如下结果：

$$\left\|\left(\mathcal{I}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{L}_{y_1}^{\dagger}\right)*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\right\|_F\le\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2}{\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2}, \tag{15}$$

其中，$\boldsymbol{\Omega}_{1k}=\boldsymbol{V}_{1k}^{H}\boldsymbol{G}_{1k}$，$\boldsymbol{\Omega}_{2k}=\boldsymbol{V}_{2k}^{H}\boldsymbol{G}_{1k}$，$\boldsymbol{G}_{1k}$ 是 $\mathcal{G}_{1k}$ 在变换域 $\mathfrak{L}$ 上的正面切片，即 $\boldsymbol{G}_{1k}:=\mathfrak{L}\left(\mathcal{G}_{1k}\right)^{\langle k\rangle}$。

同命题 4.1 一样，考虑另一种情形：$\mathcal{U}_{y_2}^{\dagger}*_{\mathfrak{L}}\mathcal{U}_{y_2}=\tilde{\mathcal{Q}}*_{\mathfrak{L}}\tilde{\mathcal{Q}}^{\mathrm{T}}$，其中 $\tilde{\mathcal{Q}}*_{\mathfrak{L}}\tilde{\mathcal{R}}$ 是算法 4 中 $\left(\mathcal{P}_2*_{\mathfrak{L}}\mathcal{Y}_{2k}*_{\mathfrak{L}}\mathcal{Q}\right)^{\mathrm{T}}$ 的瘦小版 T-QR 分解，我们得到如下结果：

$$\left\|\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}*_{\mathfrak{L}}\left(\mathcal{I}-\mathcal{U}_{y_2}^{\dagger}*_{\mathfrak{L}}\mathcal{U}_{y_2}\right)\right\|_F\le\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{L-P+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2}{\left(\hat{\sigma}_{L-P+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2}. \tag{16}$$

其中，$\boldsymbol{\Omega}_{1k}=\boldsymbol{U}_{1k}^{H}\boldsymbol{G}_{2k}^{H}$，$\boldsymbol{\Omega}_{2k}=\boldsymbol{U}_{2k}^{H}\boldsymbol{G}_{2k}^{H}$，$\boldsymbol{G}_{2k}$ 表示为 $\mathcal{G}_{2k}$ 在变换域 $\mathfrak{L}$ 中的正面切片，即 $\boldsymbol{G}_{2k}:=\mathfrak{L}\left(\mathcal{G}_{2k}\right)^{\langle k\rangle}$。

引理 1. [31]对任意的 k，令两个高斯随机矩阵为 $\boldsymbol{\Omega}_{1k}\in\mathbb{C}^{L-P\times L}$ 与 $\boldsymbol{\Omega}_{2k}\in\mathbb{C}^{N_2-L+P\times L}$，其结构与式子(13)呈现的结构相似。假设 $\boldsymbol{\Omega}_{1k}$ 是行满秩的。令 $t=\left(\frac{2}{\Delta}\right)^{\frac{1}{P+1}}$，$\mu=\sqrt{2\log\frac{2}{\Delta}}$，$\varepsilon=\frac{e\sqrt{L}}{P+1}\left(\sqrt{N_2-L+P}+\sqrt{L}\right)$，$\nu=\frac{e\sqrt{L}}{P+1}$，

[1]对任意 k，$\left(\mathcal{G}_{\mathfrak{L}}\right)^{\langle k\rangle}$ 是随机列满秩的高斯素描矩阵。

$\Theta_\Delta = t\varepsilon + tv\mu$，其中 $0<\Delta\ll 1$。那么有

$$\left\|\boldsymbol{\Omega}_{2k}\right\|_2 \left\|\left(\boldsymbol{\Omega}_{1k}\right)^\dagger\right\|_2 \le \Theta_\Delta, \tag{17}$$

以不小于 $1-\Delta$ 的概率成立。

4.1. 定理 2 的证明

对于公式(7)中的第二项，通过次乘不等式和 Frobenius 范数的性质，我们可得：

$$\begin{aligned}
&\left\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{P}} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{Y}}_{2k}\right\|_F \\
&= \Big\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{P}} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}} \\
&\quad - \boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}}\Big\|_F \\
&\le \left\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k}\right\| \left\|\boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{P}} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{A}}\right\|_F .
\end{aligned} \tag{18}$$

因此，我们进一步得到：

$$\begin{aligned}
&\left\|\boldsymbol{\mathcal{P}} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}} *_{\mathfrak{L}} \boldsymbol{\mathcal{Q}} - \boldsymbol{\mathcal{L}} *_{\mathfrak{L}} \boldsymbol{\mathcal{U}}\right\|_F \\
&\le \left(1 + \left\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k}\right\|\right) \cdot \left\|\left(\boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}^\dagger\right) *_{\mathfrak{L}} \boldsymbol{\mathcal{P}} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}}\right\|_F
\end{aligned} \tag{19}$$

下面，我们主要推导项条 $\left\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k}\right\|$ 与另一项 $\left\|\left(\boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}^\dagger\right) *_{\mathfrak{L}} \boldsymbol{\mathcal{P}} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}}\right\|_F$ 的误差上界。利用命题 4.1 与引理 1，我们可证 $\left\|\left(\boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}^\dagger\right) *_{\mathfrak{L}} \boldsymbol{\mathcal{P}} *_{\mathfrak{L}} \boldsymbol{\mathcal{A}}\right\|_F$ 的误差上界。

接下来，我们主要估计范数项 $\left\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k}\right\|$。令 $\boldsymbol{\mathcal{L}}_{y_1} = \hat{\boldsymbol{\mathcal{U}}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{S}}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{V}}}^{\mathrm{T}}$ 是 $\boldsymbol{\mathcal{L}}_{y_1}$ 的瘦小版 T-SVD 分解，其中 $\hat{\boldsymbol{\mathcal{U}}} \in \mathbb{R}^{N_1\times R\times N_3\times\cdots\times N_d}$，$\hat{\boldsymbol{\mathcal{S}}} \in \mathbb{R}^{R\times R\times N_3\times\cdots\times N_d}$，$\hat{\boldsymbol{\mathcal{V}}} \in \mathbb{R}^{R\times R\times N_3\times\cdots\times N_d}$。定义 $\boldsymbol{G}_{1k} := \mathfrak{L}\left(\boldsymbol{\mathcal{G}}_{1k}\right)^{\langle k\rangle}$，$\boldsymbol{G}_{2k} := \mathfrak{L}\left(\boldsymbol{\mathcal{G}}_{2k}\right)^{\langle k\rangle}$。因此，我们最终得到：

$$\begin{aligned}
&\left\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k}\right\| \\
&\le \left\|\hat{\boldsymbol{\mathcal{U}}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{S}}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{V}}}^{\mathrm{T}} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{U}}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{S}}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{V}}}^{\mathrm{T}}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k}\right\| \\
&\le \left\|\boldsymbol{\mathcal{G}}_{2k}\right\| \cdot \left\|\left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{U}}}\right)^\dagger\right\| \\
&= \left\|\mathrm{bdiag}\left(\mathfrak{L}\left(\boldsymbol{\mathcal{G}}_{2k}\right)\right)\right\| \cdot \left\|\mathrm{bdiag}\left(\mathfrak{L}\left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \hat{\boldsymbol{\mathcal{U}}}\right)^\dagger\right)\right\| \\
&= \sigma_{\max}\left(\mathfrak{L}\left(\boldsymbol{\mathcal{G}}_{2k}\right)^{\langle k\rangle}\right) \cdot \sigma_{\max}\left(\left[\mathfrak{L}\left(\hat{\boldsymbol{\mathcal{U}}}\right)^{\langle k\rangle}\right]^\dagger \cdot \left[\mathfrak{L}\left(\boldsymbol{\mathcal{P}}^{\mathrm{T}}\right)^{\langle k\rangle}\right]^\dagger \cdot \left[\mathfrak{L}\left(\boldsymbol{\mathcal{G}}_{2k}\right)^{\langle k\rangle}\right]^\dagger\right) \\
&\quad : \sigma_{\max}\left(\boldsymbol{G}_{2k}\right) \cdot \sigma_{\max}\left(\left[\mathfrak{L}\left(\hat{\boldsymbol{\mathcal{U}}}\right)^{\langle k\rangle}\right]^\dagger \cdot \left[\mathfrak{L}\left(\boldsymbol{\mathcal{P}}^{\mathrm{T}}\right)^{\langle k\rangle}\right]^\dagger \cdot \left[\boldsymbol{G}_{2k}\right]^\dagger\right).
\end{aligned} \tag{20}$$

利用定理 1 与引理 1，我们最终推导得到 $\left\|\boldsymbol{\mathcal{L}}_{y_1} *_{\mathfrak{L}} \left(\boldsymbol{\mathcal{G}}_{2k} *_{\mathfrak{L}} \boldsymbol{\mathcal{P}}^{\mathrm{T}} *_{\mathfrak{L}} \boldsymbol{\mathcal{L}}_{y_1}\right)^\dagger *_{\mathfrak{L}} \boldsymbol{\mathcal{G}}_{2k}\right\|$ 的误差界。

4.2. 定理 3 的证明

首先，将公式(7)中的 $\boldsymbol{G}_{2k}$ 替换为 $\left(\boldsymbol{Y}_{1k}\right)^{\mathrm{T}}$，从而得到如下：

$$\begin{aligned}&\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}*_{\mathfrak{L}}\mathcal{U}\right\|_F\\&\le\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\right\|_F\\&\quad+\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\left(\mathcal{Y}_{1k}\right)^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{Y}_{2k}\right\|_F.\end{aligned}\tag{21}$$

对于式子(21)的第二项，注意到 $\mathcal{P}*_{\mathfrak{L}}\mathcal{Y}_{1k}*_{\mathfrak{L}}\mathcal{Q}_y=\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{U}_{y_1}$，通过次乘不等式和 Frobenius 范数的性质可得到：

$$\begin{aligned}&\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\left(\mathcal{Y}_{1k}\right)^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{-1}*_{\mathfrak{L}}\mathcal{Y}_{2k}\right\|_F\\&=\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{Q}_y*_{\mathfrak{L}}\left(\mathcal{U}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{-1}*_{\mathfrak{L}}\mathcal{Y}_{2k}\right\|_F\\&=\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{Q}_y*_{\mathfrak{L}}\left(\mathcal{U}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{-1}*_{\mathfrak{L}}\mathcal{Q}_y*_{\mathfrak{L}}\left(\mathcal{U}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right.\\&\quad\left.*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{Q}_y*_{\mathfrak{L}}\left(\mathcal{U}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{-1}*_{\mathfrak{L}}\mathcal{Y}_{2k}\right\|_F\\&\le\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{Q}_y*_{\mathfrak{L}}\left(\mathcal{U}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{-1}*_{\mathfrak{L}}\mathcal{Q}_y*_{\mathfrak{L}}\left(\mathcal{U}_{y_1}\right)^{\mathrm{T}}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\mathrm{T}}\right\|\\&\quad\cdot\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}-\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\right\|_F.\end{aligned}\tag{22}$$

因此，综合式子(15)、(21)与(22)得到如下：

$$\begin{aligned}\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}*_{\mathfrak{L}}\mathcal{U}\right\|_F&\le2\left\|\left(\mathcal{I}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{L}_{y_1}^{\dagger}\right)*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\right\|_F\\&\le\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}2\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left\|\mathbf{\Omega}_{2k}\right\|_2^2\left\|\left(\mathbf{\Omega}_{1k}\right)^{\dagger}\right\|_2^2}{\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left\|\mathbf{\Omega}_{2k}\right\|_2^2\left\|\left(\mathbf{\Omega}_{1k}\right)^{\dagger}\right\|_2^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2}.\end{aligned}\tag{23}$$

将引理 1 应用到公式(23)即得到定理 3 的结果。

4.3. 定理 4 的证明

对于式子(10)的第二项，通过次乘不等式、三角不等式以及 Frobenius 范数的性质可得到：

$$\begin{aligned}&\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{Y}_{2k}*_{\mathfrak{L}}\mathcal{Q}*_{\mathfrak{L}}\left(\mathcal{U}_{y_2}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{U}_{y_2}\right\|_F\\&=\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}\right.\\&\quad\left.-\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{Y}_{2k}*_{\mathfrak{L}}\mathcal{Q}*_{\mathfrak{L}}\left(\mathcal{U}_{y_2}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{U}_{y_2}\right\|_F\\&\le\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{G}_{2k}\right\|\left\|\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}\right.\\&\quad\left.-\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}*_{\mathfrak{L}}\left(\mathcal{U}_{y_2}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{U}_{y_2}\right\|_F\\&\le\left(\left\|\left(\mathcal{I}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{L}_{y_1}^{\dagger}\right)*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\right\|_F+\left\|\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}*_{\mathfrak{L}}\left(\mathcal{I}-\mathcal{U}_{y_2}*_{\mathfrak{L}}\mathcal{U}_{y_2}^{\dagger}\right)\right\|_F\right)\\&\quad\cdot\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{G}_{2k}\right\|.\end{aligned}\tag{24}$$

进一步，得到如下的结果：

$$\begin{aligned}&\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}*_{\mathfrak{L}}\mathcal{U}\right\|_F\\&\le\left(1+\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{G}_{2k}\right\|\right)\left\|\left(\mathcal{I}-\mathcal{L}_{y_1}*_{\mathfrak{L}}\mathcal{L}_{y_1}^{\dagger}\right)*_{\mathfrak{L}}\mathcal{P}*_{\mathfrak{L}}\mathcal{A}\right\|_F\\&\quad+\left\|\mathcal{L}_{y_1}*_{\mathfrak{L}}\left(\mathcal{G}_{2k}*_{\mathfrak{L}}\mathcal{P}^{\mathrm{T}}*_{\mathfrak{L}}\mathcal{L}_{y_1}\right)^{\dagger}*_{\mathfrak{L}}\mathcal{G}_{2k}\right\|\left\|\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}*_{\mathfrak{L}}\left(\mathcal{I}-\mathcal{U}_{y_2}*_{\mathfrak{L}}\mathcal{U}_{y_2}^{\dagger}\right)\right\|_F.\end{aligned}\tag{25}$$

因此，综合式子(15)、(16)、(10)以及(24)得到：

$$\begin{aligned}&\left\|\mathcal{P}*_{\mathfrak{L}}\mathcal{A}*_{\mathfrak{L}}\mathcal{Q}-\mathcal{L}*_{\mathfrak{L}}\mathcal{U}\right\|_F\\&\le\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}\left(1+\frac{a_1\sqrt{N_1}}{c_1\sqrt{L}}\right)\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2}{\left(\hat{\sigma}_{R+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2}\\&\quad+\frac{1}{\rho}\sum_{k=1}^{N_3N_4\cdots N_d}\frac{a_1\sqrt{N_1}}{c_1\sqrt{L}}\sqrt{\frac{R\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2\left(\hat{\sigma}_{L-P+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2}{\left(\hat{\sigma}_{L-P+1}^{\langle k\rangle}\right)^2\left\|\boldsymbol{\Omega}_{2k}\right\|_2^2\left\|\left(\boldsymbol{\Omega}_{1k}\right)^{\dagger}\right\|_2^2+\left(\hat{\sigma}_1^{\langle k\rangle}\right)^2}+\left\|\mathcal{A}-\mathcal{A}_R\right\|_F^2},\end{aligned}\tag{26}$$

以不小于$1-\mathrm{e}^{-a_2N_1}-\mathrm{e}^{-c_2L}$的概率成立，其中常数$a_1,c_1,a_2,c_2$按定理 1 所定义。将引理 1 应用到公式(26)即得到定理 4 的结果。

5. 数值实验

在本节当中，我们将在几类大规模张量数据上进行实验。为了验证所提出随机单通随机化张量 LU 分解算法的有效性和优势性，将 RTLU (算法 1)、SPRTLU (算法 2)、SORTLU (算法 3)、TSRTLU (算法 4)与高阶 T-SVD [22]进行对比。实验在 Windows 操作系统中 Matlab R2023b 环境下运行，使用的实验设备配置如下：Intel(R) Xeon(R) Gold-5122 CPU @ 3.6 GHz 以及 192 GB 内存。

Table 1. Experimental datasets used for evaluation and their detailed information
表 1. 用于评估的实验数据集以及它们详细的信息

数据类型	数据名称	数据维度
三阶彩色图像	CI 1: Sunrise	3040 × 4056 × 3
	CI 2: Watzmann	6529 × 8029 × 3
四阶彩色视频	CV 1: Rush-hour	1080 × 1920 × 3 × 100
	CV 2: In-to-tree	1080 × 1920 × 3 × 100
四阶多时遥感图像	MRSI 1: Landsat-7	4500 × 4500 × 6 × 11
	MRSI 2: T22LGN	5001 × 5001 × 4 × 7
五阶光场图像数据集	LFI 1: Lego-Bulldozer	1152 × 1536 × 3 × 17 × 17
	LFI 2: Eucalyptus-Flowers	1536 × 1280 × 3 × 17 × 17

5.1. 实验设置

实验数据主要包括彩色图像(Color Images，简称 CI[2])、彩色视频(Color Videos，简称 CV[3])、多时遥感

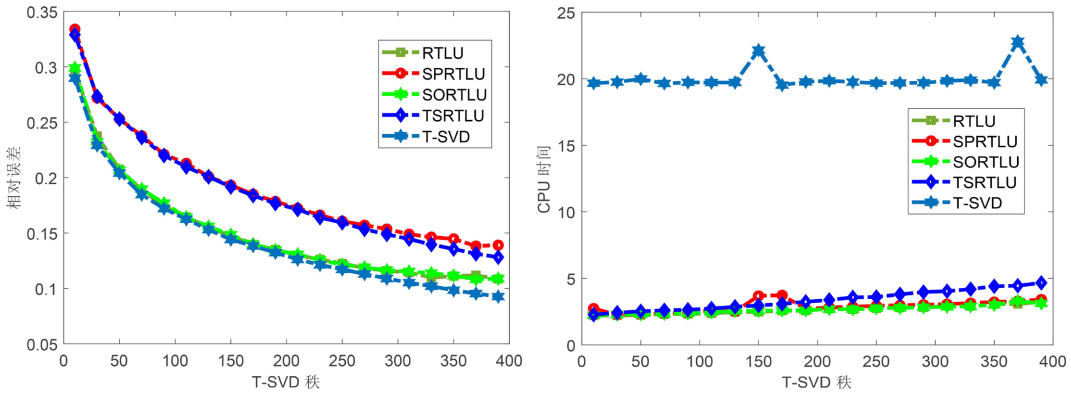
[2]https://commons.wikimedia.org/wiki/Google_Art_Project
[3]https://media.xiph.org/video/derf/

图像(Multi-Temporal Remote Sensing Images，简称 MRSI[4])以及光场图像数据集(Light Field Images，简称 LFI[5])，这些张量数据的详细信息见表 1。在进行逼近实验之前，我们对输入张量进行归一化操作。在实验中，主要考虑两种不同的可逆线性变换 $\mathfrak{L}$：(a) 离散傅里叶变换(Discrete Fourier Transform, DFT)；(b) 离散余弦变换(Discrete Cosine Transform, DCT)。对于每一种张量数据类型的逼近，若给定预期的 T-SVD 秩为 R，则设置算法 RTLU 和算法 SORTLU 中的素描大小 L 为 $R+10$，设置算法 SPRTLU 和算法 TSRTLU 中的素描大小 L 为 $2R+50$。为了评价实验效果，我们采用相对误差作为评估输入张量低秩近似精度的指标，而 CPU 运行时间用来衡量近似的计算效率。其中，相对误差定义为：

$$\text{Error} := \frac{\left\| \mathcal{A} - \mathcal{P}^{\mathrm{T}} *_{\mathfrak{L}} \mathcal{L} *_{\mathfrak{L}} \mathcal{U} *_{\mathfrak{L}} \mathcal{Q}^{\mathrm{T}} \right\|_F^2}{\left\| \mathcal{A} \right\|_F^2}. \tag{27}$$

5.2. 近似结果

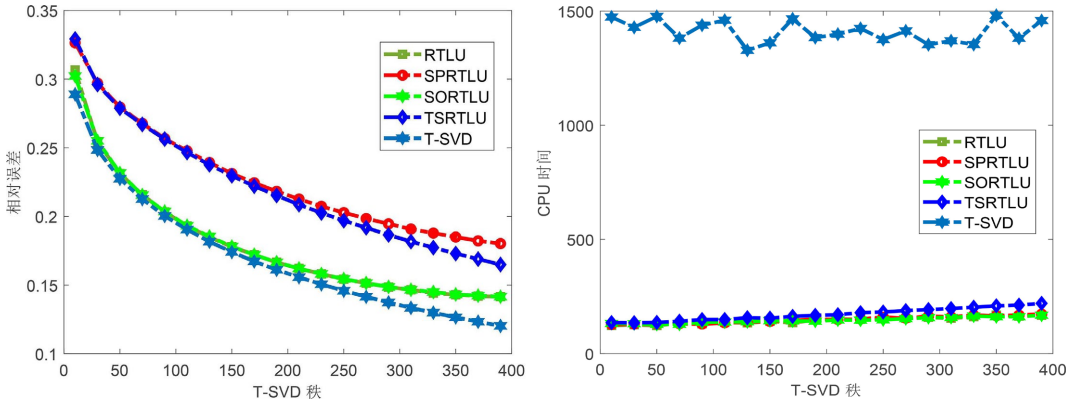
图 1~4 分别展示了所提出单通随机算法与竞争对比算法在几类大规模张量数据(包括彩色图像、彩色视频、多时遥感图像、光场图像数据集)上的逼近精度和计算效率对比。从这些实验结果，我们可以得到如下的对比结论。第一、相比于现有确定性的高阶张量近似算法 T-SVD，所提出的单通随机化张量 LU 算法在精度稍有损失的情况下，计算效率得到大幅度的提升；在彩色视频和光场图像数据上计算速度提升 3 倍左右(见图 3 和图 4)，而在彩色图像和多时遥感数据上计算速度提升 10 倍左右(见图 1 和图 2)。第二、由于数据固有的冗余性，大张量数据通常表现出较强的低秩性，因此仅需要较小的 T-SVD 秩就能将原始数据进行精确逼近；所提出的单通高阶张量 LU 算法主要是在变换域上对切片进行投影降维，空间维度参数与用于投影的 T-SVD 秩参数(或素描参数)差距越大，所用的 CPU 运行时间越小；这进一步解释了为什么算法在某些空间维度较低的数据集(如彩色视频)上的加速比要低于在某些空间维度较高的数据集上的加速比(如彩色图像、遥感图像)。第三、在所有的单通随机化张量 LU 分解算法当中，基准的 RTLU 算法与子空间单通随机算法 SORTLU 的精度比较接近，而 SPRTLU 算法与双边随机算法 TSRTLU 的精度比较接近；整体来看，RTLU 算法与 SORTLU 算法的精度要优于 SPRTLU 算法与 TSRTLU 算法的精度，这恰好与误差界理论所揭示的结论是一致的。第四、随着投影参数(包括 T-SVD 秩参数和素描大小



(a) (b)

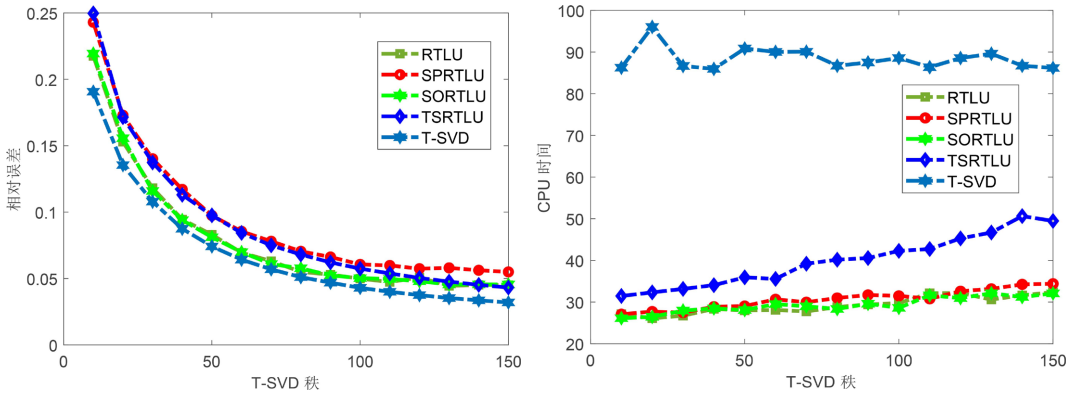
Figure 1. The comparison between the proposed single-pass randomized algorithms and the competitive algorithm in terms of approximation accuracy and computational efficiency on large-scale color images
图 1. 所提出单通随机算法与竞争对比算法在大规模彩色图像上的逼近精度和计算效率对比

[4]https://theia.cnes.fr/atdistrib/rocket/#/home
[5]http://lightfield.stanford.edu/lfs.html



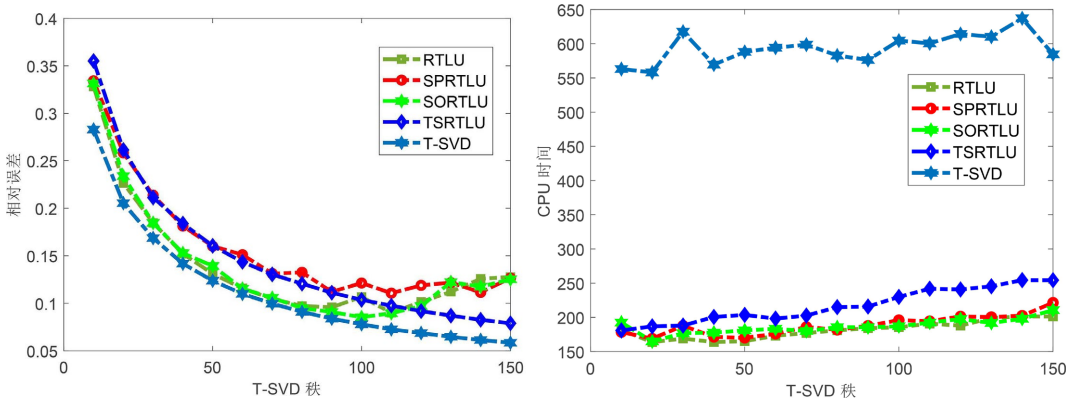
(a) (b)

Figure 2. The comparison between the proposed single-pass randomized algorithms and the competitive algorithm in terms of approximation accuracy and computational efficiency on large-scale remote-sensing image datasets
图 2. 所提出单通随机算法与竞争对比算法在大规模遥感数据上的逼近精度和计算效率对比



(a) (b)

Figure 3. The comparison between the proposed single-pass randomized algorithms and the competitive algorithm in terms of approximation accuracy and computational efficiency on large-scale color videos
图 3. 所提出单通随机算法与竞争对比算法在大规模彩色视频上的逼近精度和计算效率对比



(a) (b)

Figure 4. The comparison between the proposed single-pass randomized algorithms and the competitive algorithm in terms of approximation accuracy and computational efficiency on large-scale light-field image datasets
图 4. 所提出单通随机算法与竞争对比算法在大规模光场图像数据集上的逼近精度和计算效率对比

参数)越大，时间复杂度越高，呈单调递增关系；相对误差与之相反，其与投影参数呈现单调递减关系。也就是说，所提出的随机算法能够较好地权衡折衷精度和运行时间，允许我们选择一个较为折中的加速投影参数使得尽可能误差小且时间短。

6. 结论

借助于随机数值代数领域的随机素描技术，本文在张量－张量乘积框架下提出了若干面向大规模高阶张量快速分解的单通随机算法。基于非渐近随机矩阵理论，我们进一步严谨推导了所提算法的误差上界理论与计算复杂度。在一系列大规模高维张量数据上的实验结果已经表明：我们所提出的随机算法大幅度地提升了计算效率。

参考文献

[1] Shu, H., Wang, H., Peng, J. and Meng, D. (2024) Low-Rank Tensor Completion with 3-D Spatiotemporal Transform for Traffic Data Imputation. *IEEE Transactions on Intelligent Transportation Systems*, **25**, 18673-18687. https://doi.org/10.1109/tits.2024.3422214

[2] 李德仁, 张良培, 夏桂松. 遥感大数据自动分析与数据挖掘[J]. 测绘学报, 2014, 43(12): 1211-1216.

[3] Liu, C., Li, S., Hu, D., Wang, J., Qin, W., Liu, C., *et al*. (2024) Nonlocal Tensor Decomposition with Joint Low Rankness and Smoothness for Spectral CT Image Reconstruction. *IEEE Transactions on Computational Imaging*, **10**, 613-627. https://doi.org/10.1109/tci.2024.3384812

[4] Wang, Y., Hou, H., Yi, X., Wang, W. and Jin, S. (2025) Toward Unified AI Models for MU-MIMO Communications: A Tensor Equivariance Framework. *IEEE Transactions on Wireless Communications*, **24**, 10517-10533. https://doi.org/10.1109/twc.2025.3580321

[5] Kolda, T.G. and Bader, B.W. (2009) Tensor Decompositions and Applications. *SIAM Review*, **51**, 455-500. https://doi.org/10.1137/07070111x

[6] Tucker, L.R. (1966) Some Mathematical Notes on Three-Mode Factor Analysis. *Psychometrika*, **31**, 279-311. https://doi.org/10.1007/bf02289464

[7] Oseledets, I.V. (2011) Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, **33**, 2295-2317. https://doi.org/10.1137/090752286

[8] Zhao, Q., Zhou, G., Xie, S., Zhang, L. and Cichocki, A. (2016) Tensor Ring Decomposition. arXiv: 1606.05535.

[9] Kilmer, M.E., Horesh, L., Avron, H. and Newman, E. (2021) Tensor-Tensor Algebra for Optimal Representation and Compression of Multiway Data. *Proceedings of the National Academy of Sciences*, **118**, e2015851118. https://doi.org/10.1073/pnas.2015851118

[10] Zheng, Y.-B., Huang, T.-Z., Zhao, X.-L., Zhao, Q. and Jiang, T.-X. (2021) Fully-Connected Tensor Network Decomposition and Its Application to Higher-Order Tensor Completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**, 11071-11078. https://doi.org/10.1609/aaai.v35i12.17321

[11] Woodruff, D.P. (2014) Sketching as a Tool for Numerical Linear Algebra. *Foundations and Trends® in Theoretical Computer Science*, **10**, 1-157. https://doi.org/10.1561/0400000060

[12] Martinsson, P. and Tropp, J.A. (2020) Randomized Numerical Linear Algebra: Foundations and Algorithms. *Acta Numerica*, **29**, 403-572. https://doi.org/10.1017/s0962492920000021

[13] Murray, R., Demmel, J., Mahoney, M.W., *et al*. (2023) Randomized Numerical Linear Algebra: A Perspective on the Field with an Eye to Software. arXiv: 2302.11474.

[14] Minster, R., Saibaba, A.K. and Kilmer, M.E. (2020) Randomized Algorithms for Low-Rank Tensor Decompositions in the Tucker Format. *SIAM Journal on Mathematics of Data Science*, **2**, 189-215. https://doi.org/10.1137/19m1261043

[15] Che, M., Wei, Y. and Yan, H. (2020) The Computation of Low Multilinear Rank Approximations of Tensors via Power Scheme and Random Projection. *SIAM Journal on Matrix Analysis and Applications*, **41**, 605-636. https://doi.org/10.1137/19m1237016

[16] Che, M., Wei, Y. and Yan, H. (2025) Efficient Randomized Algorithms for Fixed Precision Problem of Approximate Tucker Decomposition. *SIAM Journal on Matrix Analysis and Applications*, **46**, 256-297. https://doi.org/10.1137/23m1594066

[17] Ahmadi-Asl, S., Cichocki, A., Huy Phan, A., Asante-Mensah, M.G., Musavian Ghazani, M., Tanaka, T., *et al*. (2020) Randomized Algorithms for Fast Computation of Low Rank Tensor Ring Model. *Machine Learning*: *Science and Technology*, **2**, Article 011001. https://doi.org/10.1088/2632-2153/abad87

[18] Shi, T., Ruth, M. and Townsend, A. (2023) Parallel Algorithms for Computing the Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*, **45**, C101-C130. https://doi.org/10.1137/21m146079x

[19] Al Daas, H., Ballard, G., Cazeaux, P., Hallman, E., Międlar, A., Pasha, M., *et al*. (2023) Randomized Algorithms for Rounding in the Tensor-Train Format. *SIAM Journal on Scientific Computing*, **45**, A74-A95. https://doi.org/10.1137/21m1451191

[20] Zhang, J., Saibaba, A.K., Kilmer, M.E. and Aeron, S. (2018) A Randomized Tensor Singular Value Decomposition Based on the T-Product. *Numerical Linear Algebra with Applications*, **25**, e2179. https://doi.org/10.1002/nla.2179

[21] Tarzanagh, D.A. and Michailidis, G. (2018) Fast Randomized Algorithms for T-Product Based Tensor Operations and Decompositions with Applications to Imaging Data. *SIAM Journal on Imaging Sciences*, **11**, 2629-2664. https://doi.org/10.1137/17m1159932

[22] Qin, W., Wang, H., Zhang, F., Ma, W., Wang, J. and Huang, T. (2024) Nonconvex Robust High-Order Tensor Completion Using Randomized Low-Rank Approximation. *IEEE Transactions on Image Processing*, **33**, 2835-2850. https://doi.org/10.1109/tip.2024.3385284

[23] Halko, N., Martinsson, P.G. and Tropp, J.A. (2011) Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, **53**, 217-288. https://doi.org/10.1137/090771806

[24] Tropp, J.A., Yurtsever, A., Udell, M. and Cevher, V. (2017) Practical Sketching Algorithms for Low-Rank Matrix Approximation. *SIAM Journal on Matrix Analysis and Applications*, **38**, 1454-1485. https://doi.org/10.1137/17m1111590

[25] Bjarkason, E.K. (2019) Pass-Efficient Randomized Algorithms for Low-Rank Matrix Approximation Using Any Number of Views. *SIAM Journal on Scientific Computing*, **41**, A2355-A2383. https://doi.org/10.1137/18m118966x

[26] Kaloorazi, M.F. and de Lamare, R.C. (2018) Subspace-Orbit Randomized Decomposition for Low-Rank Matrix Approximations. *IEEE Transactions on Signal Processing*, **66**, 4409-4424. https://doi.org/10.1109/tsp.2018.2853137

[27] Li, H. and Yin, S. (2020) Single-Pass Randomized Algorithms for LU Decomposition. *Linear Algebra and Its Applications*, **595**, 101-122. https://doi.org/10.1016/j.laa.2020.03.001

[28] 孙峻聪. 基于 DCT 变换的分块单通张量素描算法及其应用[D]: [硕士学位论文]. 杭州: 杭州电子科技大学, 2024.

[29] 程志光. 基于变换域下的素描算法在张量低秩近似中的应用[D]: [硕士学位论文]. 杭州: 杭州电子科技大学, 2023.

[30] Qin, W., Wang, H., Zhang, F., Wang, J., Luo, X. and Huang, T. (2022) Low-Rank High-Order Tensor Completion with Applications in Visual Data. *IEEE Transactions on Image Processing*, **31**, 2433-2448. https://doi.org/10.1109/tip.2022.3155949

[31] Litvak, A.E., Pajor, A., Rudelson, M. and Tomczak-Jaegermann, N. (2005) Smallest Singular Value of Random Matrices and Geometry of Random Polytopes. *Advances in Mathematics*, **195**, 491-523. https://doi.org/10.1016/j.aim.2004.08.004