中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生数学建模竞赛

学　　校	中国科学技术大学
参赛队号	21103580003
队员姓名	1.梁静思
	2.李喆昊
	3.王博铖

中国研究生创新实践系列大赛
“华为杯”第十八届中国研究生
数学建模竞赛

题 目　相关矩阵组的低复杂度计算和存储建模

摘　　要：

进入大数据时代以来，计算机视觉、射电天文、无线通信等领域的数据采集能力不断提高。随着待处理数据量的高速扩大，常规的数据处理算法对计算和存储的需求成倍增长，从而对处理器件或算法的实现成本和功耗提出了巨大的挑战。由于这些领域的信号往往以相关矩阵组的形式出现，因此，充分挖掘信号矩阵之间的关联性，以实现低复杂度的计算和存储，具有十分重要的价值和意义。

本文围绕着一种在输入信号的相关矩阵组上定义的特殊数学运算，提出了一套充分利用信号矩阵之间关联性的低复杂度计算与存储方案。

针对问题一，我们首先观察到输入信号的相关矩阵组在矩阵之间数值上的连续性。基于这一观察，我们定义了合理的矩阵间相关性的评价指标，并基于该评价指标筛选出与相邻帧矩阵相关度很高的输入信号矩阵，在保证建模精度的前提下，通过具有更低计算复杂度的插值运算估计对应的输出矩阵$\widehat{\boldsymbol{W}}$。在中间结果$\widehat{\boldsymbol{V}}$的计算上，我们实现了快速 SVD 算法以减少计算规模，其原理在于先通过基于 Lawson-Hanson-Chan 的算法对待分解矩阵进行降维，再根据 Sturm 理论直接求出前 L 个最大特征值，从而避免了代价昂贵的 QR 迭代，极大地减少了中间步骤的计算复杂度。在由中间结果计算得到输出的步骤上，我们基于改进的 Cholesky 分解的高斯消元法，避免了直接求解矩阵逆，实现了低复杂度的计算。并且，我们使用了 Strassen 算法以加速过程中的矩阵乘法。

针对问题二，我们基于问题一中定义的矩阵间相关性，我们结合了基于相关性的插值方法与基于 SVD 算法的压缩与解压缩策略，实现了对输入输出矩阵的低复杂度的压缩存储与解压缩计算，实现了最高 80%的压缩比。

关键词：相关矩阵组，相关性分析，Strassen 算法，快速 SVD 算法，改进 Cholesky 分解，高斯消元

一、问题重述

1.1 问题背景

如今人类早已步入大数据时代，海量的数据在带给我们便利的同时，也对各个领域的数据处理与分析能力提出了新的挑战。

一方面，随着数据采集能力的发展，例如成像传感器数量、雷达阵列、通信阵列的持续扩大，计算机视觉、射电天文、无线通信等领域面对的数据已是海量级别，并且仍然保持高速增长。

但另一方面，计算机硬件的计算与存储能力的发展遇到了物理层面的瓶颈，其发展速度已经远远落后于数据增长的速度。

因此，基于常规方法的数据处理算法对计算和存储的需求出现成倍增长，从而对处理器件或算法的实现成本和功耗提出了巨大的挑战。如何从算法层面减少数据处理与分析的计算与存储复杂度已然成为了迫在眉睫的需求。

通过分析人们发现，计算机视觉、相控阵雷达、声呐、射电天文、无线通信等领域的信号通常呈现为矩阵的形式，而这一系列的矩阵间通常在某些维度存在一定的关联性，因此数学上可用相关矩阵组表示。例如，视频信号中的单帧图像可以用矩阵进行表示与存储，连续的多帧图像组成了相关矩阵组，相邻图像帧或图像帧内像素间的关联性则反映在矩阵间的相关性上。

通过充分挖掘数据间的关联性，我们能够大大降低常规方法的计算与存储复杂度，从而有效解决数据的高速增长给处理器件与算法带来的挑战。

1.2 问题提出

本题将相关领域的信号表示为复数相关矩阵组$\boldsymbol{H}=\{\boldsymbol{H}_{j,k}\}$，$\boldsymbol{H}_{j,k}\in\mathbb{C}^{M\times N},j=1,\ldots,J,k=1,\ldots,K$的形式，其中，矩阵之间以及同一矩阵的元素之间有一定的相关性，但仅考虑同一行块内部的 K 个矩阵间的相关性。

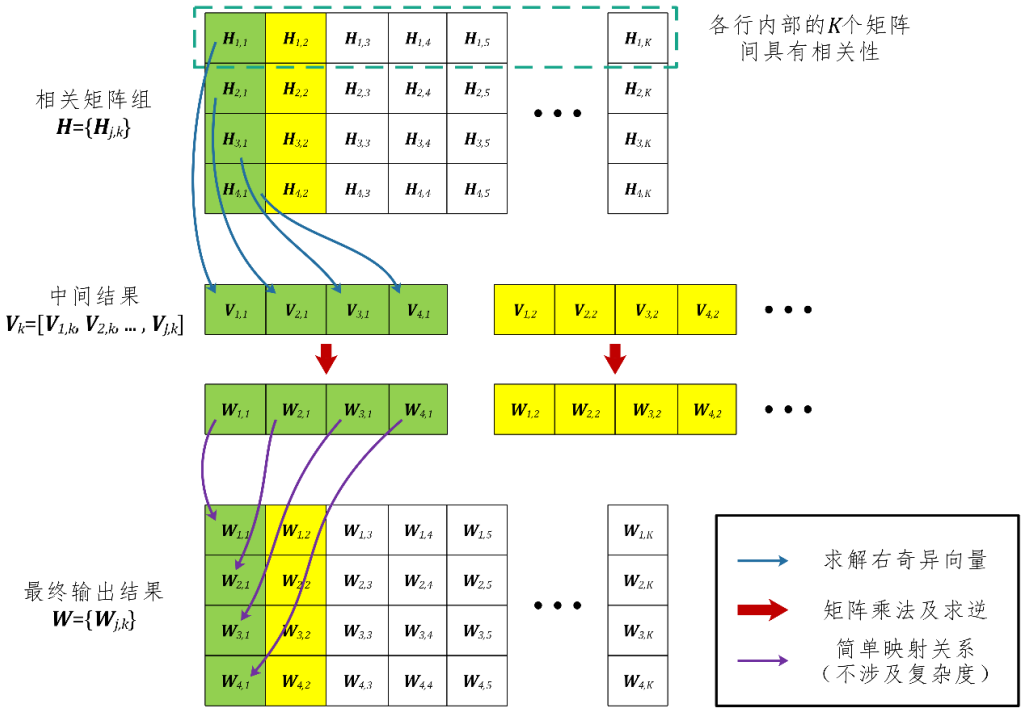


图 1 相关矩阵组的计算流程图

定义矩阵组$\boldsymbol{H}=\{\boldsymbol{H}_{j,k}\}$上的一组数学运算 $\boldsymbol{W}=f(\boldsymbol{H})$如上图所示。其中间结果$\boldsymbol{V}=\{\boldsymbol{V}_{j,k}\}$是由$\boldsymbol{H}_{j,k}$的前$L$个右奇异向量构成的矩阵。

- 问题 1：相关矩阵组的低复杂度计算
  该问题的主要任务是，利用矩阵组的相关性建模$\boldsymbol{H}$上定义的数学计算$\widehat{\boldsymbol{W}}=f(\boldsymbol{H})$，在保证建模精度$\rho_{\min}(\boldsymbol{V})\geq\rho_{th}=0.99$的前提下，降低从$H$计算得到$W$的复杂度。

- 问题 2：相关矩阵组的低复杂度存储
  该问题的主要任务是，基于给定的所有矩阵数据$\boldsymbol{H}$和$\boldsymbol{W}$，分析各自数据间的关联性，分别设计相应的压缩$P_1(\cdot)$、$P_2(\cdot)$和解压缩$G_1(\cdot)$、$G_2(\cdot)$模型，其算法流程如下图。在满足误差$err_{\boldsymbol{H}}\leq E_{th1}=-30\text{dB}$、$err_{\boldsymbol{W}}\leq E_{th2}=-30\text{dB}$的情况下，使得存储复杂度和压缩与解压缩的计算复杂度最低。

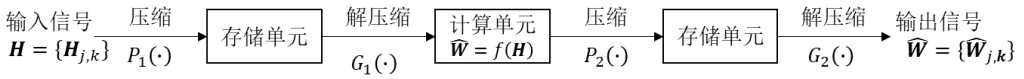


图 2 压缩与解压缩算法流程

- 问题 3：相关矩阵组的低复杂度计算和存储
  基于给定的所有矩阵数据$\boldsymbol{H}$，分析其数据间的关联性，在满足$\rho_{\min}(\boldsymbol{W})\geq\rho_{th}=0.99$的情况下，设计低复杂度计算和存储的整体方案，完成从矩阵输入信号$\boldsymbol{H}$到近似矩阵输出信号$\widehat{\boldsymbol{W}}$的端到端流程。

二、问题假设

假设 1：

$\boldsymbol{H}=\{\boldsymbol{H}_{j,k}\}$矩阵之间以及同一矩阵的元素之间有一定的相关性，但仅考虑同一行块内部的 K 个矩阵间的相关性。不考虑矩阵组$\boldsymbol{H}$中属于不同行块的矩阵（即，不同j下标的矩阵）间的相关性。

假设 2：

我们使用$a+b\,\mathbf{i}$的形式表示复数，因此复数计算的复杂度如下表：

复数操作	计算复杂度
复数加(减)	2
复数乘	14
复数除	52

其中，复数除的复杂度的计算如下式，涉及到一次倒数操作与多次加减乘操作，总计算复杂度为 52。

$$\frac{a+b\,\mathbf{i}}{c+d\,\mathbf{i}}=\frac{(a+b\,\mathbf{i})(c-d\,\mathbf{i})}{c^2+d^2}$$

四、问题一

四、问题一模型的建立与求解

4.1 矩阵关联度分析及插值

4.1.1 矩阵关联度分析

根据题目文档所述，输入信号矩阵组的每一行，即K指标所在维度的矩阵之间具有一定的关联性。对该行矩阵组做关联性分析，考虑不同矩阵之间相同位置元素的值及其变化情况，作图分析，下图为部分结果。

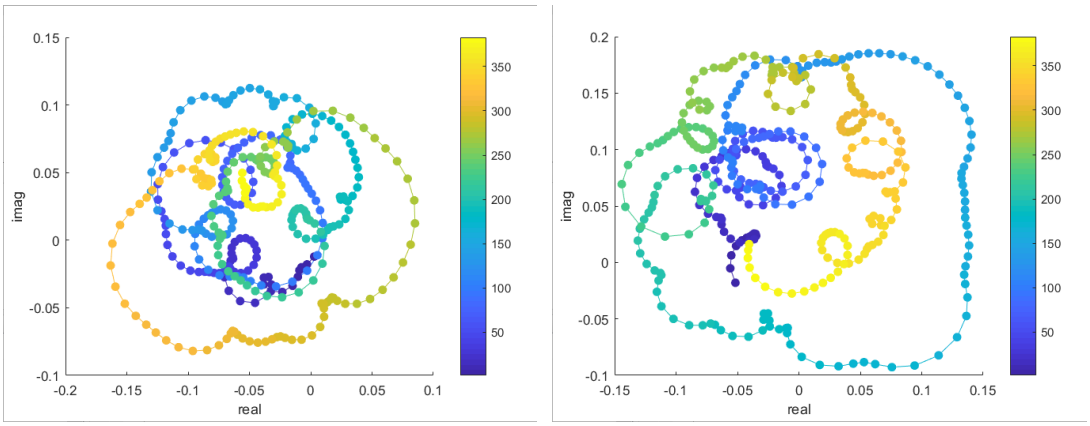


图 4.1 输入矩阵组 H 行间矩阵$H_{jk}(j$固定)内部相同位置元素的值与其变化情况，

横坐标为实部，纵坐标为虚部，

颜色代表该行矩阵组的 K 指标，K 从 1 到 384 变化

图中横轴为实轴，纵轴为虚轴，不同颜色的点表示不同矩阵的相同位置点的值，共 384 个点。

图 4.1 仅展示 Data1 的输入 H 矩阵组第一行的两个位置元素的变化情况。可以看到元素值的变化范围较小，实部与虚部都在原点附近；其次，随着 K 指标的变化，行间相邻矩阵相同位置元素的变化程度较小，可认为其是连续变化的，在图中即展示为一条连续曲线。

根据此考虑通过判断输入矩阵 H 的行间矩阵关联程度，并将其量化。主要思想是取关联度高的几个矩阵，如$\{\boldsymbol{H}_{j,1},\boldsymbol{H}_{j,2},\boldsymbol{H}_{j,3},\boldsymbol{H}_{j,4},\boldsymbol{H}_{j,5}\}$，去除中间位置的矩阵如$\boldsymbol{H}_{j,2},\boldsymbol{H}_{j,4}$，通过对$\boldsymbol{H}_{j,1},\boldsymbol{H}_{j,3},\boldsymbol{H}_{j,5}$的计算结果进行插值得到被去除的矩阵的计算结果。以此降低数据量，从而减少计算复杂度。

为了评价矩阵关联度，我们使用了将矩阵平铺后向量间的余弦相似度。

$$\mathbf{similar}(\boldsymbol{H}_{jk_1},\boldsymbol{H}_{jk_2})=\frac{\left\|\boldsymbol{H}_{jk_1}{}^{\boldsymbol{H}}\cdot\boldsymbol{H}_{jk_2}\right\|_2}{\left\|\boldsymbol{H}_{jk_1}\right\|_2\left\|\boldsymbol{H}_{jk_2}\right\|_2}$$

考虑求取矩阵组某行中相邻矩阵间的相似度值 similar，则对输入矩阵组，其矩阵间的相似度值构成一个相似度向量，该向量长度为$(384-1)=383$，取其中前 361 个元素构成热力图直观表示相邻矩阵相似度。下图展示了 Data1 数据集中$j=1,2,3,4$ 时每行矩阵组相邻矩阵之间的相似度。

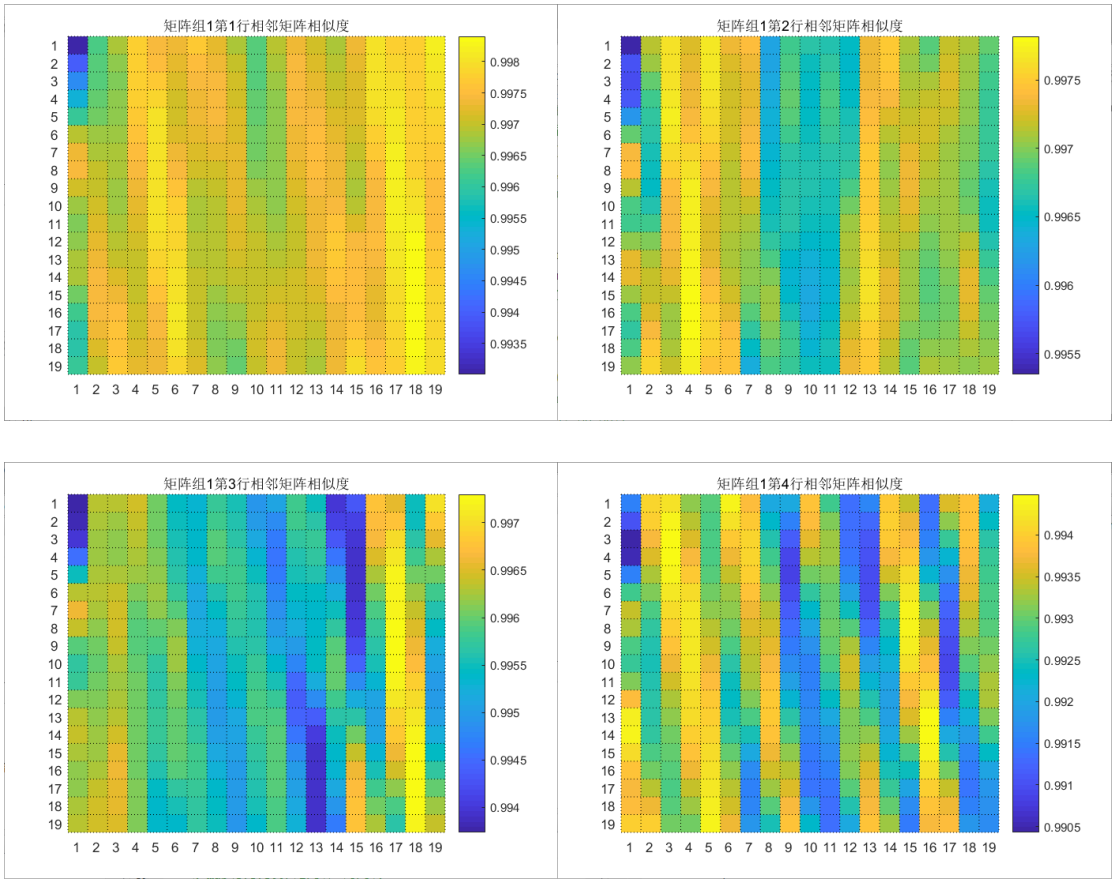
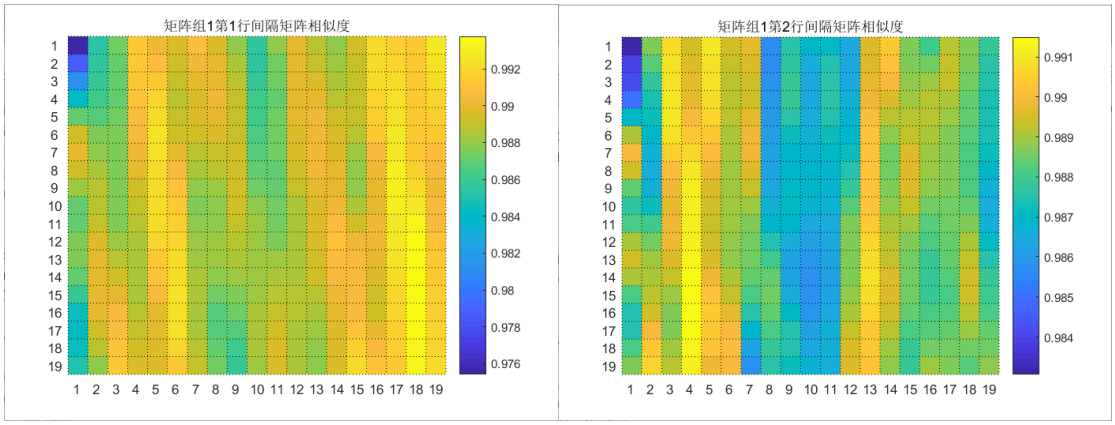


图 4.2 Data1 中矩阵组不同行中相邻矩阵（$\boldsymbol{H}_{j,k}$与$\boldsymbol{H}_{j,k+1}$）的相似度

注意到几乎所有相邻矩阵间的相似度度量均达到 0.99 以上，这也证明了矩阵间的连续性变化。为寻找合适的插值点矩阵，考虑间隔矩阵间的相似度，即考虑$\boldsymbol{H}_{j,k}$与矩阵$\boldsymbol{H}_{j,k+2}$的相似度，若此相似度度量达到阈值，则后续计算只利用矩阵$\boldsymbol{H}_{j,k}$与$\boldsymbol{H}_{j,k+2}$，矩阵$\boldsymbol{H}_{j,k+1}$对应的计算结果将通过插值获得。

类似上述，利用相似度构造一个相似度向量，该向量长度为$(384-2)=382$，同样取其中前 361 个元素构成热力图直观表示间隔矩阵相似度。



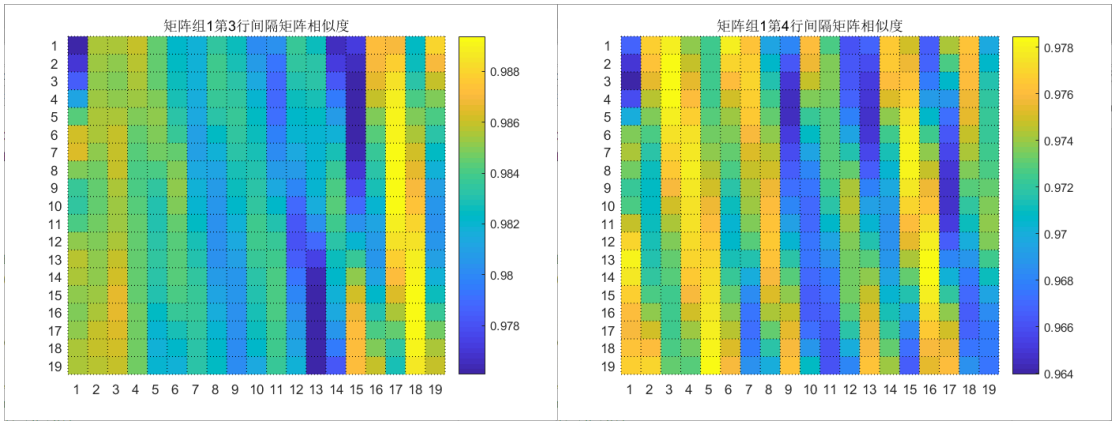


图 4.3 矩阵组 1 不同行中间隔矩阵（$\boldsymbol{H}_{j,k}$与$\boldsymbol{H}_{j,k+2}$）的相似度

选取较大的阈值，即当间隔矩阵的相似度非常高时，则将此间隔矩阵中间的矩阵作为插值点，后续计算中将不使用该矩阵。此时最大可去除的计算数据量为原数据量的一半。

此时问题的计算复杂度为

$$\boldsymbol{C_{similar}} = \boldsymbol{6 \times J \times (K-2) \times \{[(2 \times 3+1) \times (M \times N) + 1 \times (M \times N - 1)] \times 3\}}$$
$$\boldsymbol{= 18J(8MNK - K - 16MN + 2)}$$

4.1.2 插值及其复杂度分析

下面考虑对最终结果采用的插值方法。首先分析结果矩阵 W 的数据分布。

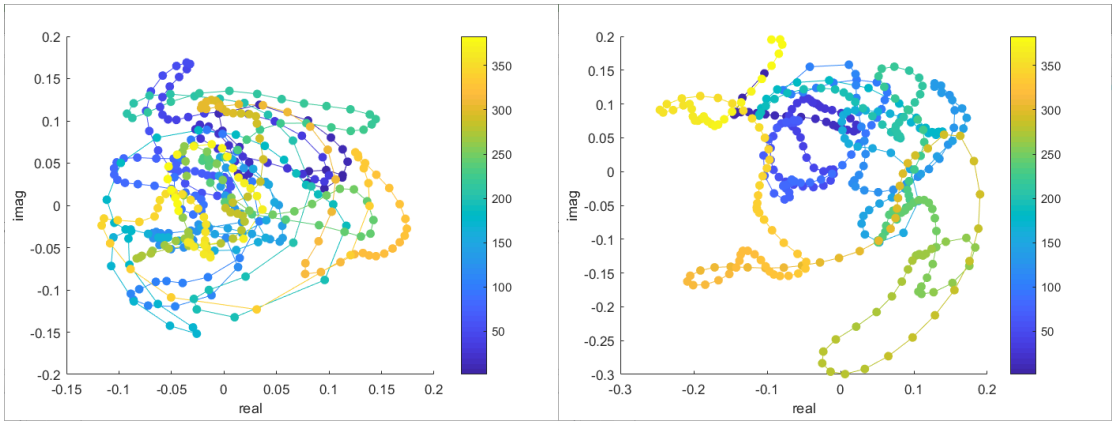


图 4.4 输出矩阵组行间矩阵W_{jk}(j固定)相同位置元素的值与其变化情况，

横坐标为实部，纵坐标为虚部，
颜色代表该行矩阵组的 K 指标，K 从 1 到 384 变化

类似于输入信号矩阵组的行间相关关系，输出矩阵组行间矩阵相同位置的不同元素值的变化范围较小，实部与虚部都在原点附近；其次不同矩阵间对应元素每次的变化程度小且是连续变化的，在图中即展示为一条连续曲线。

从计算复杂程度上考虑，由于矩阵间元素变化小且连续，因此使用计算复杂度最低的线性插值法，通过相邻点的平均值直接插值得到中间矩阵的结果。

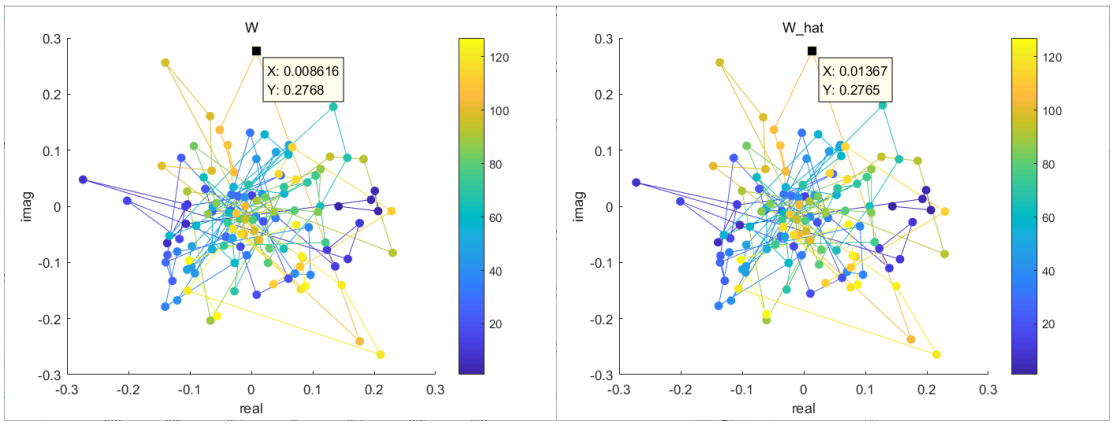


图 4.5 插值后得到的输出矩阵$\widehat{\boldsymbol{W}}_{\boldsymbol{jk}}$(jk 固定)与原输出矩阵元素$\boldsymbol{W}_{\boldsymbol{jk}}$的对比，

横坐标为实部，纵坐标为虚部，
颜色代表矩阵内部的元素下标 i，i 从 1 到 128 变化

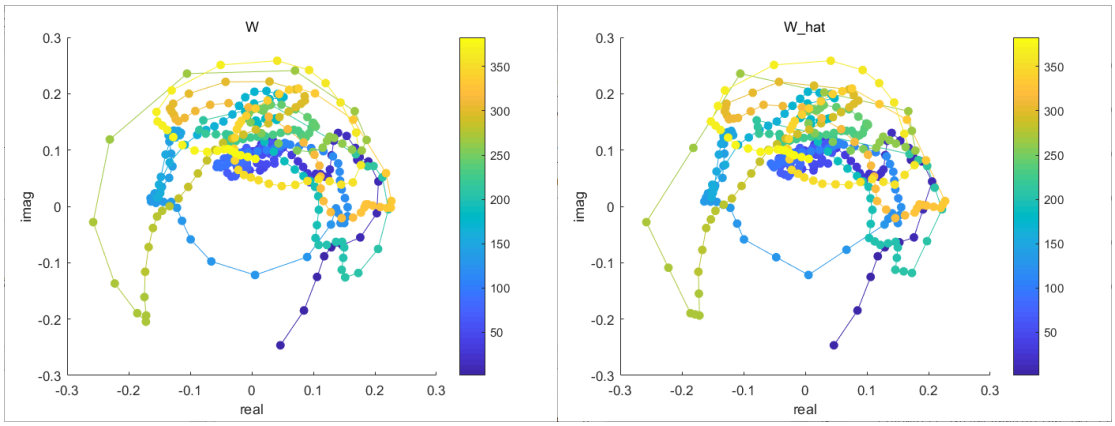


图 4.6 原输出数据矩阵$\widehat{\boldsymbol{W}}_{\boldsymbol{jk}}$(j 固定)与插值后矩阵$\boldsymbol{W}_{\boldsymbol{jk}}$对应元素变化关系，

横坐标为实部，纵坐标为虚部，
颜色代表该行矩阵组的 K 指标，K 从 1 到 384 变化

插值的结果在图 4.5 中展示，可以看到插值矩阵与原矩阵的元素间对应关系基本没有发生改变，误差很小；插值直观效果如图 4.6，可以看到通过线性插值部分元素，矩阵间对应元素的变化关系没有发生过大的改变。

针对六组数据矩阵，通过计算矩阵组每一行间隔矩阵相似度，选择插值点，然后对输出矩阵的线性插值得到插值点矩阵的结果。针对不同数据选择不同的插值点阈值，最终保证建模精度 ρ 满足要求。结果如表 4. 1。

记上述得到的六个数据矩阵组的总插值点个数为 a，此时计算复杂度为

$$\boldsymbol{C_{insert} = a \times [N \times L \times (3 \times 1 + 1 \times 3)] = 6aNL}$$

整体计算复杂度为

$$\boldsymbol{C_{simi\_ins} = 6aNL + 18J(8MNK - K - 16MN + 2)}$$

表 4.1 六组数据的最终阈值选择、插值点占比与建模精度

数据矩阵	阈值	插值占比(%)	建模精度
Data1	0.990	6.64	0.9939
Data2	0.988	0.72	0.9994
Data3	0.976	1.11	0.9950
Data4	0.991	49.48	0.9952
Data5	0.990	38.80	0.9961
Data6	0.993	17.38	0.9903

4.2 奇异值分解(SVD)算法优化

4.2.1 传统 SVD 算法及其复杂度分析

为了介绍传统 SVD 算法及其复杂度分析，需要先给出部分基础矩阵算法，包括 Househoder 变化、QR 分解和 Golub-Kahan 双对角化变换和双对角矩阵的奇异值分解。最后我们给出整个传统 SVD 算法过程和计算复杂度分析。

1). Householder 变换

Householder 变换是一种能将n维向量x变换到任一n维向量y的正交变换，特别地，Householder 变换能够将 x 变换成任意一个等长的若干个分量为 0 的向量。Householder 变换是 QR 分解算法、SVD 算法以及其它众多矩阵算法的基础。

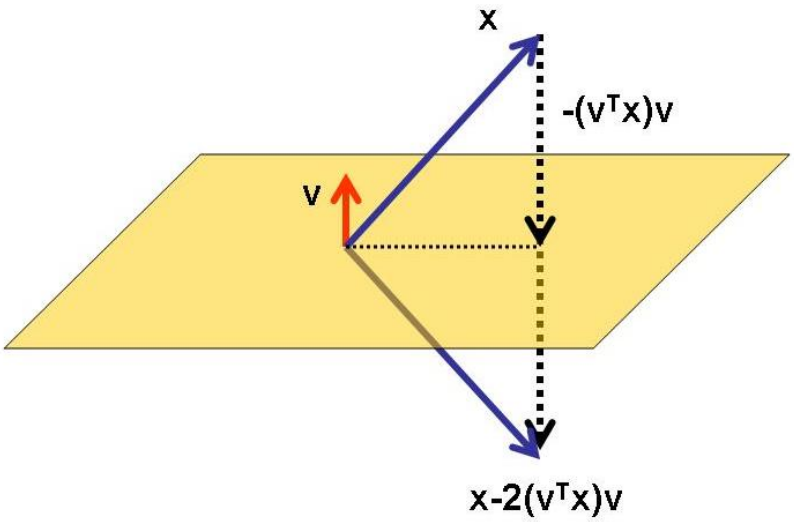


图　Householder 变换的几何意义是镜像变换

设 v 是单位向量，定义 Householder 变换

$$\boldsymbol{H = I - 2vv^T}$$

其中 H 是酉矩阵。

特别地：当

$$\boldsymbol{x = (x_1, x_2, \ldots x_n)^T}$$

$$\boldsymbol{\sigma = \left(\sum_{i=p+1}^{n} x_i^2 \right)^{\frac{1}{2}}}$$

$$\boldsymbol{y = (x_1, \dots x_p, -sign(x_{p+1})\sigma, \dots, 0)^T, p < n}$$

取$v = \frac{x-y}{||x-y||_2} = \frac{(0,\dots,0,x_{p+1}+sign(x_{p+1})\sigma,x_{p+2},\dots,x_n)^T}{\sqrt{\sigma(\sigma+sign(x_{p+1})-x_{p+1})}}$，$H = I - 2vv^T$,使得$Hx = y$。

主要计算复杂度来源是$n-p$次复数乘法、$n-p$次复数加法和$n-p$次复数除法，所以 Householder 变换的计算复杂度

$$\boldsymbol{C_{House} = (n-p) \times (2+14+52) = 68(n-p)}$$

2). QR 分解

QR 分解是一种将矩阵分解为酉矩阵 Q 和上三角矩阵 R 的矩阵分解算法，其核心思想是利用 Householder 算法的置零作用。

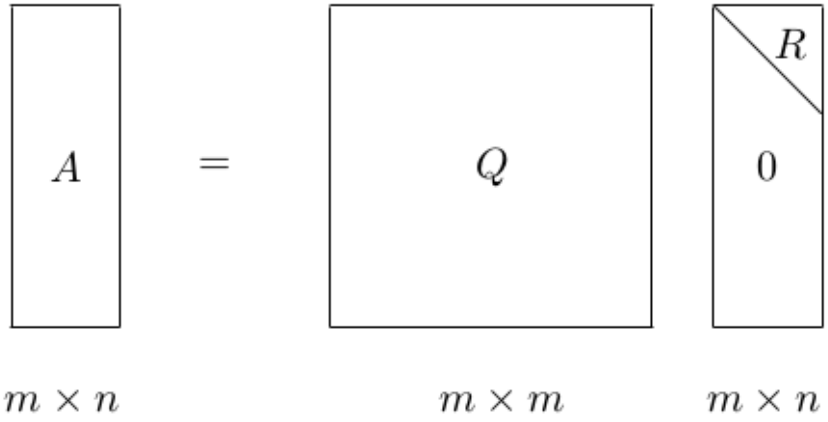


图 QR 分解

对于任意一个复数矩阵 A∈$\boldsymbol{A} \in \mathbb{C}^{m \times n}$，其中 m≥n，其 QR 分解步骤如下所示

Algorithm QR

Input:A

Output:Q,R

1: $Q = I_m, R = A$

2: **for** $k = 1 : n$ **do**

3: $x_k = A(k : m, k)$

4: $v_k = Householder(x_k)$

5: $R(k : m, k : n) = A(k : m, k : n) - 2v_k(v_k^* A(k : m, k : n))$

6: $Q(1 : m, k : m) = Q(1 : m, k : m) - 2(Q(1 : m, k : m)v_k)v_k^*$

7: end

单次循环的复杂度为

$$\boldsymbol{C_{House} + 32(m-k)(m+n-k) + 28(m-k)}$$

总计算复杂度为（忽略 2 次及更低阶项）

$$\boldsymbol{C_{QR} = 32m^2n - \frac{16}{3}n^3}$$

3). Golub-Kahan 双对角化

Golub-Kahan 算法是一种经典的双对角化算法，其作用是将任意一个复数矩阵$\boldsymbol{A} \in \mathbb{C}^{m\times n}$（其中 m≥n）转化为双对角矩阵，算法基本思想是不断地使用 Householder 变换，将矩阵 A 的特定行向量或列向量的部分元素置零，最终达到双对角化的目的。其算法步骤如下所示。

Algorithm Golub-Kahan

Input:A

Output:P,J,Q

1: $P = I_m, Q = I_n$
2: **for** $k = 1 : n$ **do**
3: $x_k = A(k : m, k)$
4: $u_k = Householder(x_k)$
5: $A(k : m, k : n) = A(k : m, k : n) - 2u_k(u_k^* A(k : m, k : n))$
6: $P(k : m, 1 : m) = P(k : m, 1 : m) - 2u_k(u_k^* P(k : m, 1 : m))$
7: **if** $k \leq n - 2$ **then**
8: $y_k = A(k, k + 1 : n)^T$
9: $u_k = Householder(y_k)$
10: $A(k : m, k+1 : n) = A(k : m, k+1 : n)-2(A(k : m, k+1 : n)v_k)v_k^*$
11: $Q(1 : n, k + 1 : n) = Q(1 : n, k + 1 : n) - 2(Q(1 : n, k + 1 : n)v_k)v_k^*$
12: end
13: J=A
14: end

根据上述算法步骤描述，单次循环复杂度为

$$\boldsymbol{C_{House} + 32(m-k)(n+m-k) + 28(m-k) +}$$
$$\boldsymbol{C_{House} + 32(n-k-1)(m+n-k) + 28(n-k-1)}$$

Golub-Kahan 算法的算法复杂度为

$$\boldsymbol{C_{GK} = 32m^2n + \frac{16}{3}n^3}$$

4). 双对角矩阵的 SVD 分解

双对角矩阵对角化的典型算法是 QR 迭代。其过程如下：

设 B 为双对角矩阵，$B_1 = B$,

$$\boldsymbol{B_iB_i^H = Q_{i1}R_{i1}}\ ,\ \boldsymbol{B_i^HB_i = Q_{i2}R_{i2}}$$

$$\boldsymbol{B_{i+1} = Q_{1i}^H B_i Q_{2i}} \qquad \boldsymbol{i = 1, 2, 3, \ldots}$$

当迭代次数足够大，B_i将会趋于对角矩阵Σ，$U = Q_{11}Q_{21}\ldots Q_{p1}$，$V = Q_{12}Q_{22}\ldots Q_{p2}$，$p$为迭代次数，$B = U\Sigma V^H$，算法计算复杂度为

$$\boldsymbol{C_{QRIT} = (80m^2n + 16mn^2 - \frac{29n^3}{3} + m^3)p}$$

5). 传统 SVD 算法。

传统 SVD 包括两部分，于任意一个复数矩阵$A \in \mathbb{C}^{m\times n}$，其中$m \geq n$，算法流程总结为：

Step1	利用 Golub-Kahan 算法将矩阵 A 进行双对角化，得 $\boldsymbol{A = U_1 B V_2^H}$
Step2	利用 QR 迭代算法对 B 进行 SVD 分解，得 $\boldsymbol{B = U_2 \Sigma V_2^H}$
Step3	最终的 SVD 分解结果为 $\boldsymbol{A = U \Sigma V^H}$ 其中$U = U_1 U_2,\ V = V_1 V_2$

总算法复杂度为

$$\boldsymbol{C_{SVD} = (80m^2n + 16mn^2 - \frac{29n^3}{3} + m^3)p + \frac{16}{3}n^3 + 32m^2n}$$

4.2.2 快速 SVD 算法

通过 4.2.1 的分析，SVD 算法的算法复杂度与矩阵 A 的尺寸 m、n 相关，为了预先降低被分解矩阵的尺度，本文使用了如下措施：

- 使用更高效的 Lawson-Hanson-Chan 算法，预先降低待分解矩阵尺寸。

同时，由于本问题中仅仅需要提取V的前L个向量，并不需要将进行完整的 SVD 运算，依据这个原则，我们又对 SVD 算法做了如下改进：

- 在对双对角矩阵 B 进行奇异值分解时，先采用 Sturm 算法计算出前L个特征值（从大到小排列），再利用反幂法求出对应的特征向量

1) Lawson-Hanson-Chan 算法

Lawson-Hanson-Chan 算法的核心思想在于，当$m \gg n$时，对 A 进行 QR 分解

$$\boldsymbol{A = QR}$$

其中 R 为上三角矩阵，设

$$R = \begin{bmatrix} R_0 \\ O \end{bmatrix}$$

其中R_0为$n \times n$的上三角矩阵。**可以证明，后续对于$\boldsymbol{R_0}$进行奇异值分解，即可达到本题目的**。证明如下：

设

$$R_0 = U_0 \Sigma_0 V_0^H$$
$$Q = [Q_0\ Q_1]$$

其中Q_0为$m \times n$矩阵，Q_1为$m \times (n-m)$矩阵。

$$A = QR = [Q_0\ Q_1]\begin{bmatrix} R_0 \\ O \end{bmatrix} = Q_0 R_0 = Q_0 U_0 \Sigma_0 V_0^H$$

对比 A 的 SVD 分解结果

$$A = U \Sigma V^H$$

我们求出了 U 的前 n 列和全部的Σ和 V，只有 U 是未全部求出，但是根据本问题需求，我们已经达到了目的。

综合来看，Lawson-Hanson-Chan 算法的作用在于将$m \times n$矩阵的 SVD 问题转化为$n \times n$矩阵的 SVD 问题，代价是多了一次 QR 分解。根据 4.2.1 中的分析，SVD 分解的代价远远大于 QR 分解，因此，这种改进是值得的。

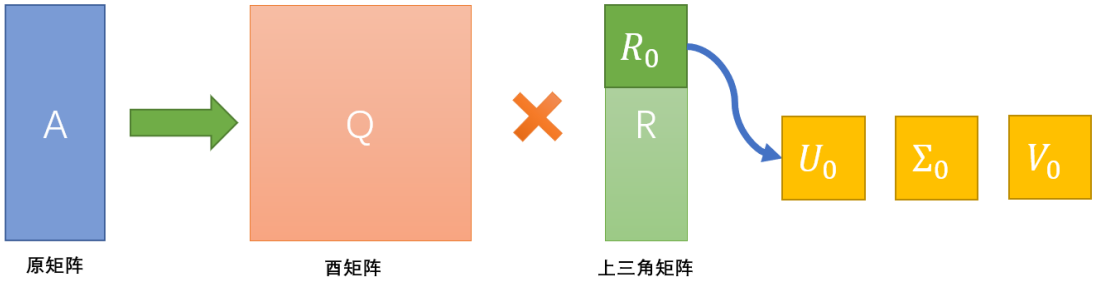


图 4.7　Lawson-Hanson-Chan 算法核心示意图

2)　Sturm 算法

双对角方阵的奇异值求解问题可以化为对称三对角矩阵的特征值求解问题。设 A 为 n×n 双对角方阵，取 B=A^H A，则 B 为三对角矩阵

$$B = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

用$f_i(\lambda)$表示$B - \lambda I$的前i阶主子式，$f_i(\lambda)$有如下联系

$$\boldsymbol{f_0(\lambda) = 1}$$
$$\boldsymbol{f_1(\lambda) = \alpha_1 - \lambda}$$
$$\cdots\cdots$$
$$\boldsymbol{f_i(\lambda) = (\alpha_i - \lambda) f_i(\lambda) - \beta_{i-1}^2 f_{i-1}(\lambda), \quad i = 2, 3 \ldots n}$$

序列 $S(\lambda) = \{f_0(\lambda), f_1(\lambda), \ldots, f_i(\lambda)\}$称为 Sturm 序列。

定义整数函数$V(x)$表示$\{f_0(\lambda), f_1(\lambda), \ldots, f_i(\lambda)\}$在$x = \lambda$时的正负符号跳变次数，则$V(x)$是特征方程$f_n(\lambda)$在$[x, +\infty]$上的根的个数。

文献 [4] 中利用了 Gerschgorin 理论进行特征值区间范围的估计，根据文献[4],矩阵 B 特征值上界为

$$\boldsymbol{\lambda_{max} \le max(\alpha_1 + |\beta_1|, \alpha_2 + |\beta_i + \beta_{i-1}|, \ldots, \alpha_{n-1} + |\beta_{n-1} + \beta_{n-2}|, \alpha_n + |\beta_{n-1}|)}$$

特征值下界为

$$\boldsymbol{\lambda_{min} \ge min(\alpha_1 - |\beta_1|, \alpha_2 - |\beta_i + \beta_{i-1}|, \ldots, \alpha_{n-1} - |\beta_{n-1} + \beta_{n-2}|, \alpha_n - |\beta_{n-1}|)}$$

据此可以设计出一种迭代算法快速计算出前 L 个特征值而不必求出所有特征值。其算法流程图如图 4.8 所示。

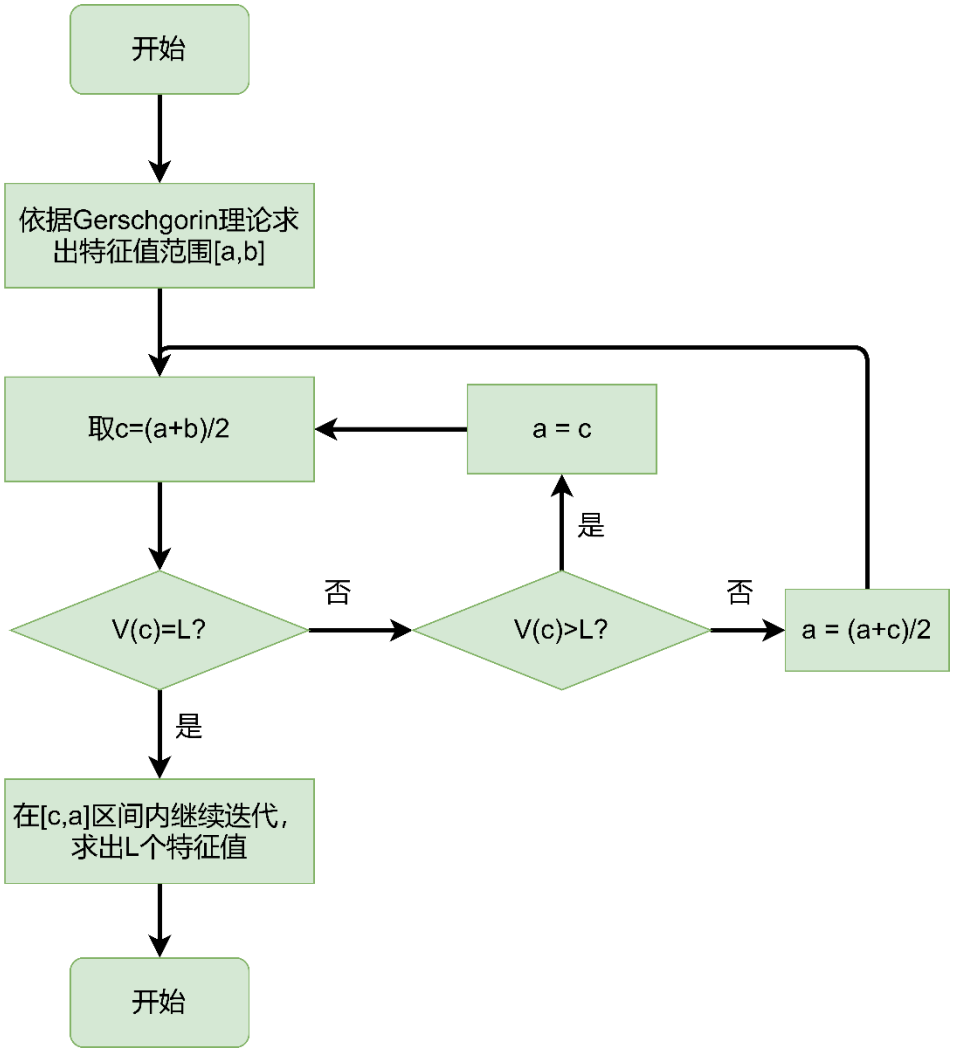


图 4.8 基于 Sturm 理论的特征值求解算法流程图

假设已知[a,b]上有根，取$c = \frac{a+b}{2}$，根据$V(a)$，$V(b)$，$V(c)$的值判断$[a,b]$与$[b,c]$区间根的情况，进而缩小了特征根的范围，当区间长度足够小时，可以取该区间中点作为特征值的近似值。

利用 Sturm 算法可以很方便地求出前 L 个特征值（从大到小排列），假设迭代了 p 次，则整个 Sturm 算法的复杂度为

$$\boldsymbol{C_{Strum} = 32np}$$

求出 B 的近似特征值$\tilde{\lambda}$, 再用反幂法求出$B - \tilde{\lambda} I$的绝对值最小的特征值λ_0和对应的特征向量x，即有

$$\boldsymbol{\left(B - \tilde{\lambda} I\right) = \lambda_0 x}$$

则

$$\boldsymbol{Bx = (\tilde{\lambda} + \lambda_0)x}$$

反幂法亦是一种迭代算法，对于一个$n \times n$的矩阵 A，根据文献【----】，反幂法的计算复杂度

$$\boldsymbol{C_{IP} = (C_{\mathrm{Inv}} + 52n)p}$$

其中，$\boldsymbol{C_{\mathrm{Inv}}}$表示矩阵求逆运算复杂度，具体形式见 4.3 节。

3) 快速 SVD 算法

综合上述分析，最终的快速 SVD 算法流程图如图 4.9 所示，总复杂度为

$$\boldsymbol{C_{FastSVD} = (C_{Inv} + 52n)p + 32m^2n + 32n^3}$$

对于本问题中 M<N 的情况，需先把矩阵进行转置，再进行快速 SVD 操作。

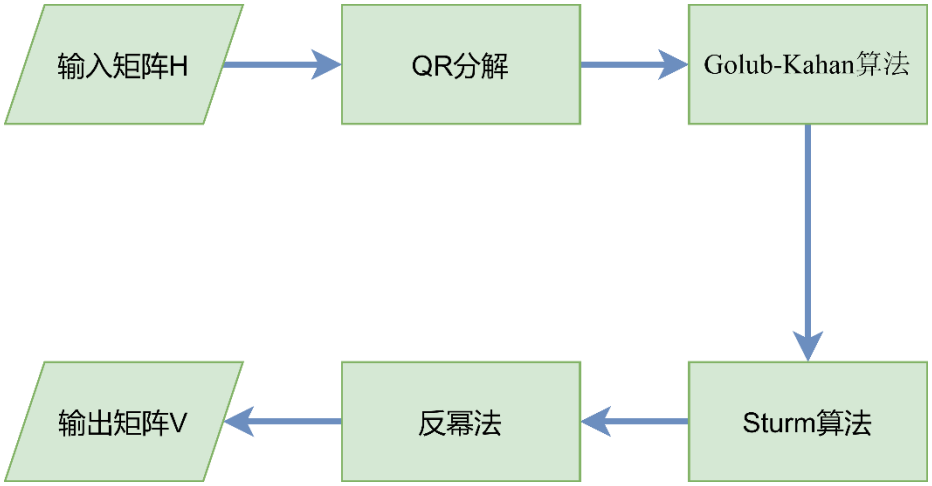


图 4.9 快速 SVD 算法流程图

4.3 从$\boldsymbol{V_k}$计算$\boldsymbol{W_k}$的低复杂度解法分析

题干给出的从$\boldsymbol{V_k}$计算$\boldsymbol{W_k}$的初始公式为：

$$\boldsymbol{W_k = V_k(V_k^H V_k + \sigma^2 I)^{-1}} \qquad \boldsymbol{(4.1)}$$

为了计算(4.1)式，我们需要使用矩阵乘法计算$\boldsymbol{V_k^H V_k}$。

首先，我们给出对传统矩阵乘法的复杂度的分析。

设 $A \in \mathbb{C}^{m\times n}, B \in \mathbb{C}^{n\times k}$，则使用传统矩阵乘法计算$A \cdot B$的算法如下：

Algorithm 传统矩阵乘法

Input: $A \in \mathbb{C}^{m\times n}, B \in \mathbb{C}^{n\times k}$

Output: $C \in \mathbb{C}^{m\times k}$

```
for i = 1 : m do
    for j = 1 : k do
        C_ij = Σ_{t=1}^{n} A_it B_tj
```

计算过程中有：$n \times k \times m$次复数乘法，$(n-1) \times k \times m$次复数加法。根据我们的假设，复数乘法复杂度为 14，复数加法复杂度为 2，则传统矩阵乘法的计算复杂度为 $\boldsymbol{14n \times k \times m + 2(n-1) \times k \times m}$。

当$m = k = n$时：复杂度为 $\boldsymbol{14n^3 + 2(n-1)n^2 = 16n^3 - 2n^2}$。

为了优化矩阵乘法的计算复杂度，我们使用了 Strassen 算法[4]。

Strassen 算法是一种基于分治思想的加速矩阵乘法。对于阶数为$n = 2^x$的矩阵乘法，其算法流程如下：

输入: $\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{bmatrix}\in\mathbb{C}^{n\times n}$，$\boldsymbol{B}=\begin{bmatrix}\boldsymbol{B}_{11} & \boldsymbol{B}_{12}\\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22}\end{bmatrix}\in\mathbb{C}^{n\times n}$

$$\boldsymbol{C}=\begin{bmatrix}\boldsymbol{C}_{11} & \boldsymbol{C}_{12}\\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22}\end{bmatrix}=\begin{bmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{bmatrix}\begin{bmatrix}\boldsymbol{B}_{11} & \boldsymbol{B}_{12}\\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22}\end{bmatrix}$$

$$=\begin{bmatrix}\boldsymbol{M}_1+\boldsymbol{M}_4-\boldsymbol{M}_5+\boldsymbol{M}_7 & \boldsymbol{M}_3+\boldsymbol{M}_5\\ \boldsymbol{M}_2+\boldsymbol{M}_4 & \boldsymbol{M}_1-\boldsymbol{M}_2+\boldsymbol{M}_3+\boldsymbol{M}_6\end{bmatrix}$$

其中，

$$\boldsymbol{M}_1=(\boldsymbol{A}_{11}+\boldsymbol{A}_{22})(\boldsymbol{B}_{11}+\boldsymbol{B}_{22})$$
$$\boldsymbol{M}_2=(\boldsymbol{A}_{21}+\boldsymbol{A}_{22})\boldsymbol{B}_{11}$$
$$\boldsymbol{M}_3=\boldsymbol{A}_{11}(\boldsymbol{B}_{12}-\boldsymbol{B}_{22})$$
$$\boldsymbol{M}_4=\boldsymbol{A}_{22}(\boldsymbol{B}_{21}-\boldsymbol{B}_{11})$$
$$\boldsymbol{M}_5=(\boldsymbol{A}_{11}+\boldsymbol{A}_{12})\boldsymbol{B}_{22}$$
$$\boldsymbol{M}_6=(\boldsymbol{A}_{21}-\boldsymbol{A}_{11})(\boldsymbol{B}_{11}+\boldsymbol{B}_{12})$$
$$\boldsymbol{M}_7=(\boldsymbol{A}_{12}-\boldsymbol{A}_{22})(\boldsymbol{B}_{21}+\boldsymbol{B}_{22})\ 。$$

此时，两个n阶矩阵间的矩阵乘法将使用$7^{\log_2 n}=n^{\log_2 7}$次复数乘法，以及$\frac{n^2\log_2 n}{4}\times 18$次复数加法，可知 Strassen's 方法下$n$阶矩阵乘法的计算复杂度为：

$$\boldsymbol{n^{log_2 7}\times 14+\frac{n^2\,log_2 n}{4}\times 18\times 2}。$$

让n在$0\sim100$区间中离散变化，我们绘制出了传统矩阵乘法与 Strassen 算法的计算复杂度的变化趋势图，如下。我们发现，在$n>4$时，Strassen 算法的复杂度已经低于传统矩阵乘法。

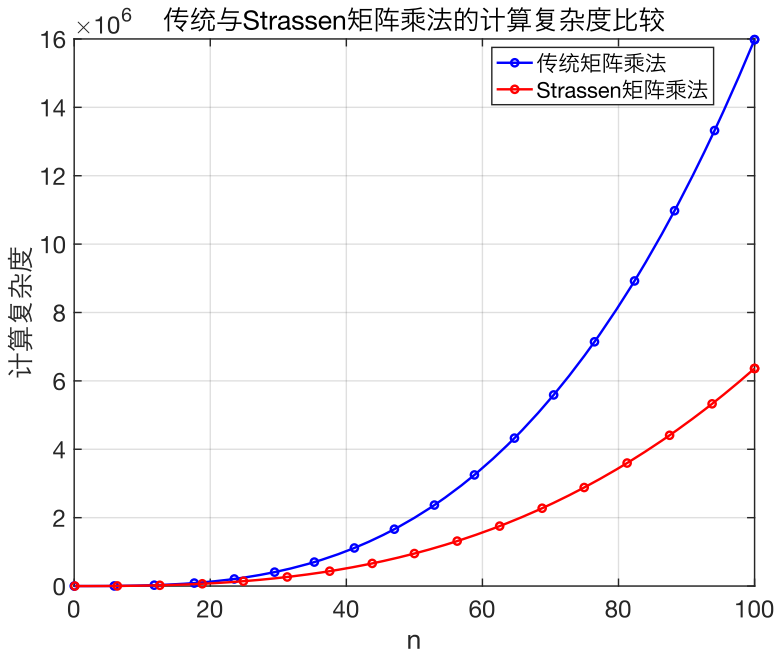


图 4.10 传统与 Strassen 矩阵乘法的计算复杂度随 n 变化时的比较

由于在我们需要计算的矩阵乘法$\boldsymbol{V}_k^H\boldsymbol{V}_k$中，$\boldsymbol{V}_k$的维度为$64\times 8$。所以我们选择使用 Strassen 矩阵乘法降低计算的复杂度。

此时计算一次$\boldsymbol{V}_k^H\boldsymbol{V}_k$的复杂度为

$$\frac{\boldsymbol{64}}{\boldsymbol{8}}\times \boldsymbol{8}^{\boldsymbol{log_2 7}}\times \boldsymbol{14}+\frac{\boldsymbol{64\times 8\,log_2 8}}{\boldsymbol{4}}\times \boldsymbol{18}\times \boldsymbol{2}\ =\boldsymbol{52240}$$

接下来，我们将介绍对于$(\boldsymbol{V}_k^H\boldsymbol{V}_k+\sigma^2\boldsymbol{I})^{-1}$求解的加速。

在数值计算中，我们一般避免对较大的矩阵直接进行矩阵求逆，而是将式(4.1)转化为：

$$(\boldsymbol{V}_k^H\boldsymbol{V}_k+\sigma^2\boldsymbol{I})\boldsymbol{W}_k^H=\boldsymbol{V}_k^H \tag{4.2}$$

可以证明，方程(4.2)的系数矩阵$(\boldsymbol{V}_k^H\boldsymbol{V}_k+\sigma^2\boldsymbol{I})$是复正定的。因此，我们可以使用下面介绍的基于改进 Cholesky 分解的高斯消元法求解方程组(4.2)，以实现较低的计算复杂度。

我们在此简要地介绍一下矩阵的 Cholesky 分解及其改进。

Cholesky 分解[5]是共轭对称的正定复方阵的一种分解方式。设$A\in\mathbb{C}^{n\times n}, A=A^H, A>0$，则$A$的 Cholesky 分解为：

$$\boldsymbol{A}=\boldsymbol{LL^H}$$

其中，L为下三角阵。

Cholesky 分解的一种计算算法如下：

Algorithm Cholesky 分解

Input: $A \in \mathbb{C}^{n\times n}$

Output: $L \in \mathbb{C}^{n\times n}$

```
for i = 1 : n do
    for j = 1 : i do
        sum = Σ_{k=1}^{j-1} L_jk L_jk^H
        if i == j then
            L_ij = sqrt(A_ii − sum)
        else
            L_ij = (1/L_jj)(A_ij − sum)
```

由于 Cholesky 分解在求解过程中需要使用平方根计算，为了降低计算复杂度，我们使用了改进后的 Cholesky 分解（又被称为 LDL 分解），其公式如下：

$$\boldsymbol{A = LDL^H}$$

其中L为下三角阵，D为对角阵。改进后的 Cholesky 分解的计算算法如下：

Algorithm 改进后的 Cholesky 分解（LDL 分解）

Input: $A \in \mathbb{C}^{n\times n}$

Output: $L \in \mathbb{C}^{n\times n}, D \in \mathbb{C}^{n}$

```
for i = 1 : n do
    for j = 1 : i do
        sum = Σ_{k=1}^{j-1} L_jk D_k L_jk^H
        if i == j then
            L_ij = A_ii − sum
        else
            L_ij = (1/D_j)(A_ij − sum)
```

记$a = \sum_{i=1}^{n-2}\big(i \times (n-i-1)\big)$, $b = \sum_{i=1}^{n-1} i$, 该改进 Cholesky 分解包含b次复数除法，$2\times a$ 次复数乘法，$n+b+a$次复数加减法，其计算复杂度为：

$$\begin{aligned}(\boldsymbol{52+2}) \times \sum_{\boldsymbol{i=1}}^{\boldsymbol{n-1}} \boldsymbol{i} + \boldsymbol{2\times n} + (\boldsymbol{14\times 2+2}) \times \sum_{\boldsymbol{i=1}}^{\boldsymbol{n-2}} \big(\boldsymbol{i \times (n-i-1)}\big) \\ = \boldsymbol{5n^3 + 12n^2 - 15\,n}\end{aligned} \tag{4.3}$$

基于改进 Cholesky 分解算法，我们对方程(4.2)通过高斯消元求解。

记$A \coloneqq (\boldsymbol{V_k^H V_k} + \sigma^2\boldsymbol{I}), B \coloneqq V_k^H, x \coloneqq W_k^H$，改进的求逆算法如下图：

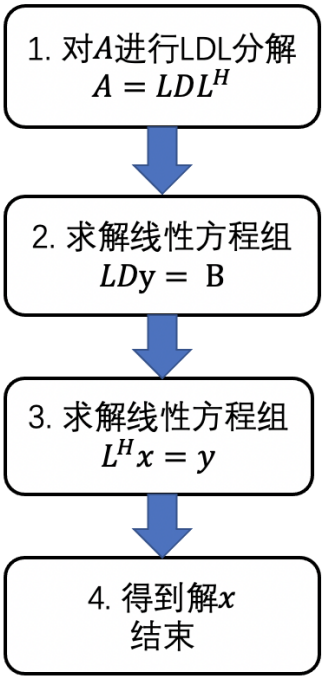


图 4.11 基于改进 Cholesky 分解的高斯分解的计算过程

第 2 步求解线性方程组$LDy = B$中LD为下三角阵，求解的计算复杂度很低，算法流程为：

Algorithm LDy=B 求解

Input: $L \in \mathbb{C}^{n\times n}, D \in \mathbb{C}^{n}, B \in \mathbb{C}^{n\times n},$

Output: $y \in \mathbb{C}^{n}$

```
for j = 1 : n − 1 do
    for i = j + 1 : n do
        y_i = y_i − L_{i,j} y_i
    y_i = y_i / D_i
y_n = y_n / D_n
```

由于 y 的维度为8×64，在 y 的每一列的计算过程包含n次复数除法，$\sum_{i=1}^{n-1} i$ 次复数乘法与减法。其总体复杂度公式为：

$$\mathbf{64} \times \left[(\mathbf{14}+\mathbf{2}) \times \sum_{\boldsymbol{i=1}}^{\boldsymbol{n-1}} \boldsymbol{i} + \mathbf{52} \times \boldsymbol{n}\right] = \mathbf{64} \times (\mathbf{8}\boldsymbol{n}^{\mathbf{2}} + \mathbf{44}\boldsymbol{n})$$

第 3 步求解线性方程组$L^H x = y$中，L^H为上三角阵，该步骤的求解复杂度同样很低，算法流程为：

Algorithm $L^H x = y$ 求解

Input: $L \in \mathbb{C}^{n\times n}, y \in \mathbb{C}^n,$

Output: $x \in \mathbb{C}^n$

$U = L^H$

for $j = n : -1 : 1$ **do**

  for $i = 1 : j-1$ **do**

    $x_i = x_i - U_{i,j}x_i$

由于 x 的维度为8×64，其总体复杂度公式为：

$$64 \times \left[(\mathbf{14}+\mathbf{2}) \times \sum_{\boldsymbol{i=1}}^{\boldsymbol{n-1}} \boldsymbol{i}\right] = \mathbf{64} \times (\mathbf{8}\boldsymbol{n}^{\mathbf{2}} - \mathbf{8}\boldsymbol{n})$$

最终，我们使用基于改进 Cholesky 分解的高斯消元法直接计算得到$\boldsymbol{W}_k^H$（相当于间接求$(\boldsymbol{V}_k^H\boldsymbol{V}_k + \sigma^2\boldsymbol{I})^{-1}$）的算法复杂度为：

$$\mathbf{5}\boldsymbol{n}^{\mathbf{3}} + \mathbf{1036}\boldsymbol{n}^{\mathbf{2}} + \mathbf{2289}\,\boldsymbol{n}$$

代入$n = 8$，则最终复杂度为 87176。

五、问题二模型建立与求解

5.1 矩阵 H 的压缩与解压缩

5.1.1 基于 SVD 主成分分析的压缩算法设计

在 4.1.1 中，我们对与 H 矩阵的相关性进行了详细分析，结果显示，$H_{j,1}, H_{j,2}, \dots H_{j,K}$， $j = 1,2,\dots,J$之间具有很强的关联，相邻矩阵的相似度达 99%以上。据此，我们设计一种基于 SVD 主成分提取的压缩算法。

首先，为了利用 H 在 K 维度上的关联性，需对矩阵进行重新排列。排列规则如下：

a) 将$M \times N$的一个$H_{j,k}$展开成一个行向量$a_{j,k}$

b) K 个行向量组成一个矩阵

$$\boldsymbol{A_j} = \begin{bmatrix} \boldsymbol{a_{j,1}} \\ \boldsymbol{a_{j,2}} \\ \vdots \\ \boldsymbol{a_{j,K}} \end{bmatrix}$$

c) J个A_j组成矩阵$A = [A_1, A_2, \dots, A_J]$

该重新排列将 A 的是一个三维张量，由 J 个$K \times MN$矩阵组成，如图所示，矩阵的列向量之间存在关联性。对于每一个A_j进行 SVD 分解可得

$$\boldsymbol{U_j\Sigma_j V_j^H = A_j}$$

Σ的对角元素为$\sigma_1, \sigma_2, \dots, \sigma_n$，称为奇异值，奇异值的大小表示该维度数据的重要程度。下面将根据奇异值对于U_j和V_j进行主成分截断，其步骤如下

a) 将奇异值从大到小排列$\sigma_1, \sigma_2, \dots, \sigma_n$
b) 设定阈值$0 < \sigma_{th} < 1$，求出一个最小的l，使得

$$\sum_{i=1}^{l} \boldsymbol{\sigma_i > \sigma_{th}}$$

c) 对U_j、Σ_j、V_j进行截断，取

$$\widetilde{\boldsymbol{U_j}} = \boldsymbol{U_j(:, 1{:}l)}, \qquad \widetilde{\boldsymbol{V_j}} = \boldsymbol{V_j(1{:}l, :)}, \qquad \widetilde{\boldsymbol{\Sigma_j}} = \boldsymbol{\Sigma_j(1{:}l, 1{:}l)}$$

根据奇异值分解的性质，容易证明，

$$\boldsymbol{\tilde{A} = \widetilde{U_j}\widetilde{\Sigma_j}\widetilde{V_j}^H \approx A}$$

在进行存储时，我们仅需存储$\widetilde{U_j}$、$\widetilde{V_j}$ 和$\widetilde{\Sigma_j}$的对角线元素。

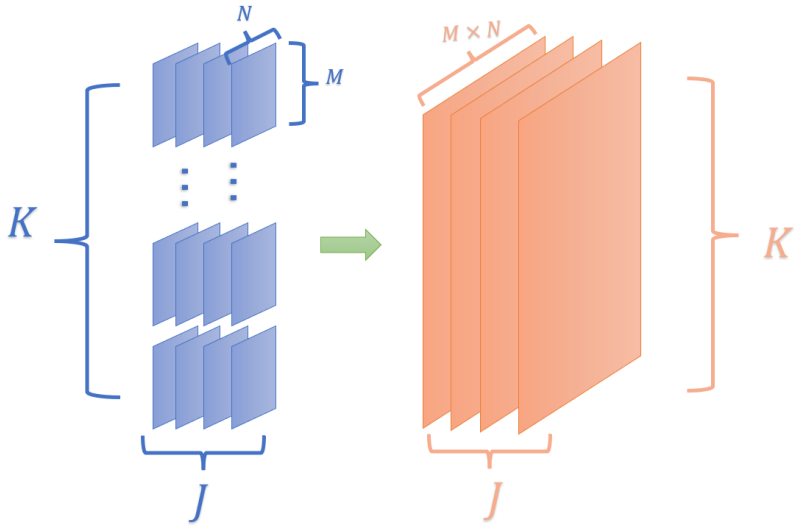


图 5.1 数据重排示意图

综上所述，可得 H 的压缩算法$P_1(\cdot)$，其流程图如图所示，即先对矩阵进行 SVD 分解，再对分解结果进行主成分截断，截断结果作为最终压缩结果。

图 5.2 矩阵数据 H 压缩函数流程图

原数据存储复杂度为

$$\boldsymbol{S_0 = 64MNJK}$$

压缩后的数据存储复杂度为

$$\boldsymbol{S_1 = 64(MN + K)Jl}$$

定义压缩比

$$\boldsymbol{r_p = \frac{S_1}{S_0} = \frac{(MN + K)l}{MNK}}$$

当 $l < \frac{NMK}{MN+K}$，压缩比小于 1，压缩算法起到压缩作用。

压缩算法的计算复杂度主要在于 SVD 分解的计算复杂度。根据问题 1 的描述，$MN \times K$复数矩阵的 SVD 分解计算复杂度为

$$\boldsymbol{(C_{\text{Inv}} + 52K)p + 32(MN)^2K + 32K^3}$$

总计算复杂度

$$C_{p1} = ((\boldsymbol{C_{\text{Inv}} + 52K)p + 32(MN)^2K + 32K^3})\boldsymbol{J}$$

解压函数$G_1(\cdot)$的流程图如图所示

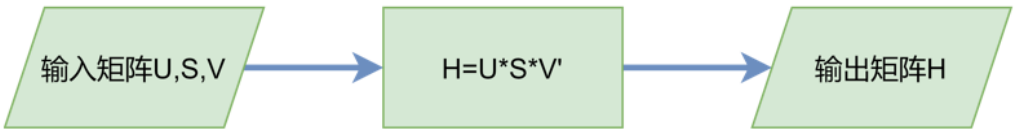


图 5.3 矩阵数据 H 解压函数流程图

5.1.2 算法验证与测试

先利用数据集 Data1 中的数据进行实验。读取其中的 H 矩阵，利用 5.1.1 所设计的算法对其进行压缩和解压缩，所得误差-奇异值截断阈值关系图如图 5.4 所示。，其中，误差的定义为

$$\boldsymbol{err_H = 10 * \log_{10} \frac{E\left\{\left\|\hat{H}_{j,k} - H_{j,k}\right\|_F^2\right\}}{E\left\{\left\|H_{j,k}\right\|_F^2\right\}}}$$

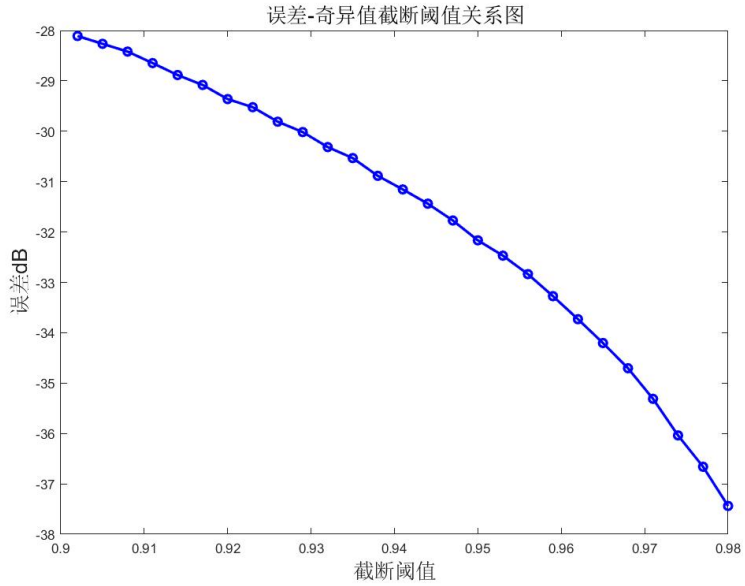


图 5.4 数据集 1 矩阵 H 压缩误差与截断阈值关系图

图 5.5 显示了压缩比随着截断阈值的变化。结合图 5.4 与图 5.5 可见，压缩误差与阈值呈现正相关，即截断阈值越高，压缩误差越小，信息损失越小。但信息损失减小的同时，压缩比也在升高，说明压缩的效果变差，存储复杂度变高了。为了权衡压缩误差和压缩比这对矛盾指标，需要合理地设置截断阈值。

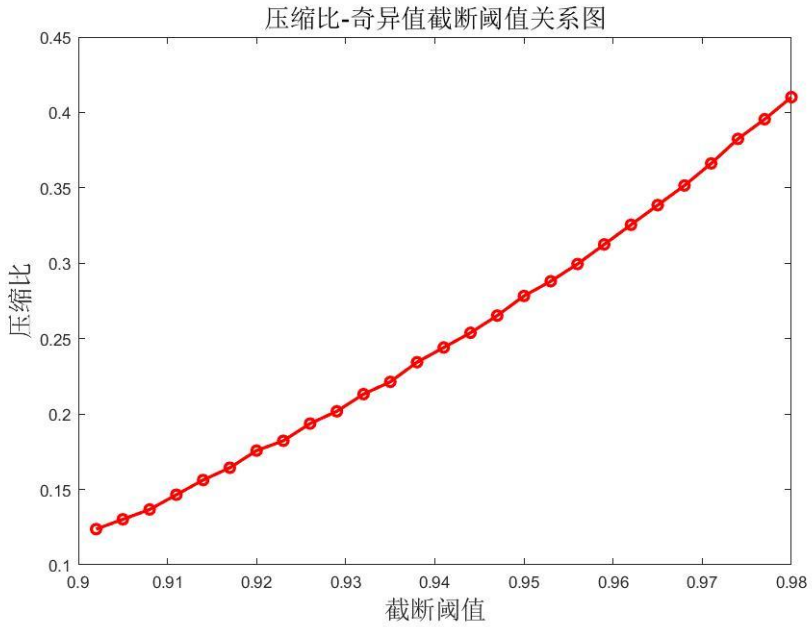


图 5.5 数据集 1 矩阵 H 压缩误差与截断阈值关系图

对于本问题，压缩误差上限$\max(err_H) = -30dB$，据此，可以定义指标最佳截断阈值为

$$\max\{\sigma_{th}\}$$
$$s.t. \max(err_H) \leq -30dB.$$

表 5.1 中显示了不同数据集的不同最佳截断阈值与对应的压缩比。可以看到，最佳阈值在 0.9~0.95 区间内，而压缩比在 0.2~0.3 区间内，说明在满足压缩误差条件的情况下，最高可节省约 75%的存储控件。

表 5.1 不同数据集最佳截断阈值与压缩比对比

数据集	最佳截断阈值	压缩比
Data1	0.929	0.2018
Data2	0.941	0.2817
Data3	0.941	0.2813
Data4	0.908	0.2360
Data5	0.923	0.2946
Data6	0.923	0.2832

5.2 矩阵数据 W 的压缩与解压

根据 4.1.1 中的分析，矩阵 W 与 H 的类似，$W_{j,1}, W_{j,2}, \ldots W_{j,K}$， $j = 1,2, \ldots, J$ 之间具有很强的关联，据此，基于 SVD 的压缩算法同样适用于 W。不同的是，W 数据间的相关性较弱，因而奇异值阈值σ_{th}与 5.1 有一定差异，其设定值比 5.1 中的值要大。

图 5.6 显示了所得误差-奇异值截断阈值关系，图 5.7 显示了压缩比随着截断阈值的变化。其结果与 5.1 类似，但是明显看出压缩效果落后于 5.1，为了满足压缩误差要求，压缩比需要控制在约 0.8 以上。

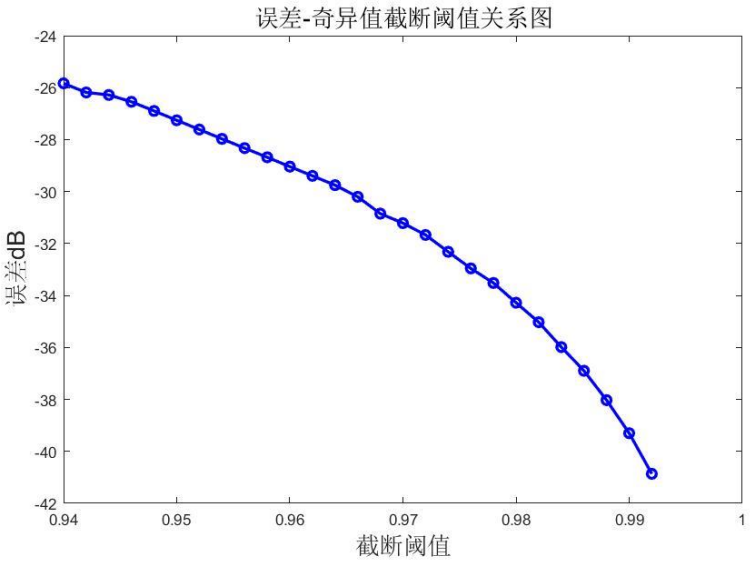


图 5.6 数据集 1 矩阵 W 压缩误差与截断阈值关系图

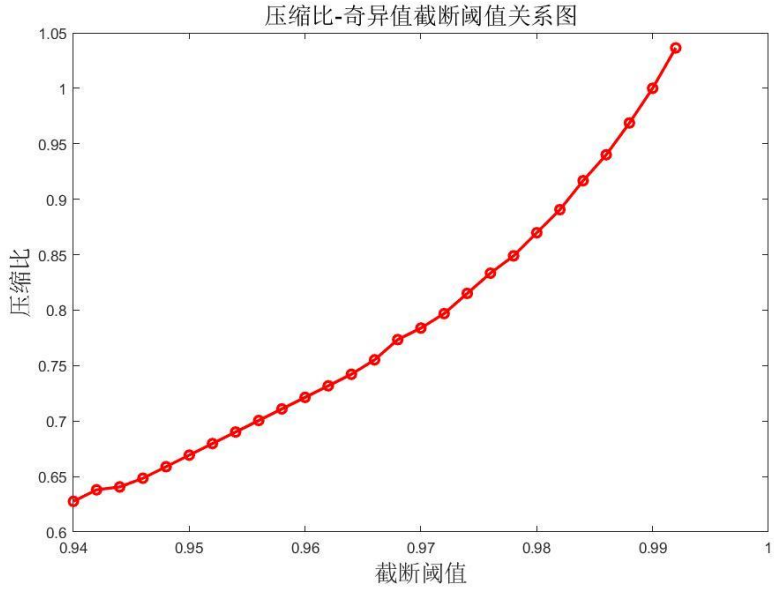


图 5.7 数据集 1 矩阵 W 压缩误差与截断阈值关系图

表 5.2 中显示了不同数据集的不同最佳截断阈值与对应的压缩比。可以看到，最佳阈值在 0.93~0.975 区间内，而压缩比在 0.49~0.95 区间内，其跨度范围明显大于 H，其原因在于 W 矩阵的的相关性小于 H。

表 5.2 不同数据集最佳截断阈值与压缩比对比

数据集	最佳截断阈值	压缩比
Data1	0.966	0.7552
Data2	0.970	0.8828
Data3	0.972	0.9427
Data4	0.934	0.4974
Data5	0.954	0.6172
Data6	0.956	0.6128

参考文献

[1] 王勇等，数值分析，北京：高等教育出版社，212~219，2009
[2] M. K. Jain, S. R. K. Iyengar and R. K. Jain, “Numerical Methods for Scientific and Engineering Computation”, Wiley Eastern Limited, 1993
[3] Golub, Gene, and William Kahan. "Calculating the singular values and pseudo-inverse of a matrix." Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis 2.2 (1965): 205-224.
[4]Strassen,Volker. "Gaussian elimination is not optimal." Numerische mathematik 13.4 (1969): 354-356.

[5] Unknown Author, Cholesky decomposition,.
https://en.wikipedia.org/wiki/Cholesky_decomposition
访问时间202