2021年中国研究生数学建模竞赛A题（华为题目）

相关矩阵组的低复杂度计算和存储建模

一、问题背景

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，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视为一个矩阵，连续的多帧图像组成了相关矩阵组，而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。随着成像传感器数量/雷达阵列/通信阵列的持续扩大，常规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。因此，充分挖掘矩阵间关联性，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。

二、建模描述

给定一组复数矩阵$\boldsymbol{H}=\{\boldsymbol{H}_{j,k}\}$，$\boldsymbol{H}_{j,k}\in\mathbb{C}^{M\times N},j=1,\dots,J,k=1,\dots,K$。其中，矩阵之间以及同一矩阵的元素之间有一定的相关性，包括：相同j下标、不同k下标的矩阵间存在一定的关联，即$\{\boldsymbol{H}_{j,1},\boldsymbol{H}_{j,2},\boldsymbol{H}_{j,3},\cdots,\boldsymbol{H}_{j,K}\}$间存在关联[1]；且矩阵

$$\boldsymbol{H}_{j,k}=\begin{bmatrix}h_{1,1}^{(j,k)} & h_{1,2}^{(j,k)} & h_{1,3}^{(j,k)} & \cdots & h_{1,N}^{(j,k)}\\ h_{2,1}^{(j,k)} & h_{2,2}^{(j,k)} & h_{2,3}^{(j,k)} & \cdots & h_{2,N}^{(j,k)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ h_{M,1}^{(j,k)} & h_{M,2}^{(j,k)} & h_{M,3}^{(j,k)} & \cdots & h_{M,N}^{(j,k)}\end{bmatrix}$$

的各个元素间$\left\{h_{m,n}^{(j,k)}\right\},m=1,\dots,M,n=1,\dots,N$，也存在关联。

定义矩阵组$\boldsymbol{H}=\{\boldsymbol{H}_{j,k}\}$上的一组数学运算，其中间结果$\boldsymbol{V}=\{\boldsymbol{V}_{j,k}\}$由如下的公式给出：

$$\boldsymbol{V}_{j,k}=svd(\boldsymbol{H}_{j,k}),\text{or }\boldsymbol{H}_{j,k}=\boldsymbol{U}_{j,k}\boldsymbol{S}_{j,k}\widetilde{\boldsymbol{V}}_{j,k}^{H},\boldsymbol{V}_{j,k}=\widetilde{\boldsymbol{V}}_{j,k}^{H}(:,1:L)$$

$$j=1,\dots,J;\ k=1,\dots,K$$

[1] 在本问题中，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。

其中，$svd(\cdot)$为矩阵的奇异值分解（即，SVD 分解）中求解右奇异向量的过程，其简要说明可参考附录一；$\boldsymbol{V}_{j,k}$是由$\boldsymbol{H}_{j,k}$的前L个右奇异向量构成的矩阵，L 个之后的右奇异向量可以忽略，维度为$N \times L$。

进一步，为得到最终输出结果$\boldsymbol{W} = \{\boldsymbol{W}_{j,k}\}$，先将不同$j$下标、相同$k$下标的$\boldsymbol{V}_{j,k}$进行**横向的**拼接，得到维度为$N \times LJ$的$\boldsymbol{V}_k = [\boldsymbol{V}_{1,k} \quad \cdots \quad \boldsymbol{V}_{j,k} \quad \cdots \quad \boldsymbol{V}_{J,k}]$，然后根据如下公式获取$\boldsymbol{W}_k$：

$$\boldsymbol{W}_k = \boldsymbol{V}_k(\boldsymbol{V}_k^H\boldsymbol{V}_k + \sigma^2\boldsymbol{I})^{-1}$$

其中，σ^2为固定常数；$\boldsymbol{W}_k$维度同$\boldsymbol{V}_k$；$\boldsymbol{I}$为单位矩阵，维度为$LJ \times LJ$。

最后，将各$\boldsymbol{W}_k$按如下的公式进行拆解：

$$\boldsymbol{W}_k = [\boldsymbol{W}_{1,k}, \dots, \boldsymbol{W}_{j,k}, \dots, \boldsymbol{W}_{J,k}]$$

其中，$\boldsymbol{W}_{j,k}$是$\boldsymbol{W}_k$中顺序排列的子矩阵，维度为$N \times L$。上述各流程亦可参考附录二中的图示说明。

为了降低计算和储存的复杂度，分析相关矩阵组的关联性，通过数学建模对输出结果$\boldsymbol{W}$进行估计，建模过程表示为：

$$\widehat{\boldsymbol{W}} = f(\boldsymbol{H})$$

其中$\widehat{\boldsymbol{W}}$即为对输出结果$\boldsymbol{W}$的建模估计。这一建模过程又可以拆分为 2 个步骤：

$$\widehat{\boldsymbol{V}} = f_1(\boldsymbol{H}), \qquad \widehat{\boldsymbol{W}} = f_2(\widehat{\boldsymbol{V}})$$

其中$f_1(\cdot)$表示从输入矩阵组$\boldsymbol{H}$到中间结果$\boldsymbol{V}$的建模过程，$\widehat{\boldsymbol{V}}$表示中间结果$\boldsymbol{V}$的建模估计，$f_2(\cdot)$表示从中间结果$\boldsymbol{V}$到最终结果$\boldsymbol{W}$的建模过程，$\widehat{\boldsymbol{W}}$表示最终结果$\boldsymbol{W}$的建模估计。定义$\boldsymbol{W}$的建模估计精度为：

$$\rho_{l,j,k}(\boldsymbol{W}) = \frac{\left\|\widehat{\boldsymbol{W}}_{l,j,k}^H\boldsymbol{W}_{l,j,k}\right\|_2}{\left\|\widehat{\boldsymbol{W}}_{l,j,k}\right\|_2\left\|\boldsymbol{W}_{l,j,k}\right\|_2}, l = 1, \dots, L$$

其中，$\|\cdot\|_2$表示矢量的欧几里得范数（也即 2 范数，对于列矢量$\boldsymbol{a}$，$\|\boldsymbol{a}\|_2 = \sqrt{\boldsymbol{a}^H\boldsymbol{a}}$）；$\boldsymbol{W}_{l,j,k}$表示$\boldsymbol{W}_{j,k}$的第$l$列。上式中，$\widehat{\boldsymbol{W}}_{l,j,k}^H\boldsymbol{W}_{l,j,k}$为复数标量，此处取其欧几里得范数即获取其模值。

为描述方便，额外定义$\boldsymbol{W}$的最低建模精度$\rho_{\min}(\boldsymbol{W})$为：

$$\rho_{\min}(\boldsymbol{W}) \triangleq \min_{\substack{l\in\{1,2,\ldots,L\}\\ j\in\{1,2,\ldots,J\}\\ k\in\{1,2,\ldots,K\}}} \rho_{l,j,k}(\boldsymbol{W})$$

其中，$\min_{l,j,k}(\cdot)$表示在l,j,k三个维度上取最小值。另外，中间结果$\boldsymbol{V}$的建模估计精度$\rho_{l,j,k}(\boldsymbol{V})$的定义及最低建模精度$\rho_{\min}(\boldsymbol{V})$的定义与此相同。

下面对建模过程中涉及的**计算复杂度**、**存储复杂度**的定义进行说明：

计算复杂度定义为由矩阵组$\boldsymbol{H}$计算得到结果矩阵组$\boldsymbol{W}$所需要的总计算复杂度。复数矩阵运算可拆解为基本的复数运算，而基本的复数运算又可进一步拆解为基本的实数运算。例如，复数乘法按照$(a+bj)(c+dj)=(ac-bd)+(ad+bc)j$计算的复杂度为 4 次实数乘法和 2 次实数加(减)法。实数基本运算的复杂度按照下表计算，其中，实数的加(减)法运算与乘法运算的计算复杂度对比可参考文献[1]。

表 1 实数基本运算的计算复杂度

运算类型	计算复杂度
加(减)法	1
乘法	3
倒数	25
平方根	25
自然指数	25
自然对数	25
正弦	25
余弦	25
其它	100

如果使用常规的算法完成本问题中的运算流程，获取输出结果$\boldsymbol{W}=\{\boldsymbol{W}_{j,k}\}$的计算复杂度由如下几个部分构成：

- 求解右奇异向量过程中的计算复杂度。当使用文献[2]中使用的方法时，基于双对角化（bi-diagonalization）结合 QR 分解的操作，对于 SVD 分解本身可以达到近似$O(NM^2+M^3)$的复杂度 [2]，其主要流程可参考附录

[2] 在本问题中，N的值大于M的取值 10 倍以上

三。

- 获取逆矩阵$(\boldsymbol{V}_k^H\boldsymbol{V}_k+\sigma^2\boldsymbol{I})^{-1}$过程中的计算复杂度。当使用高斯消元法时，求解维度$LJ\times LJ$的矩阵的逆矩阵的复杂度近似为$O((LJ)^3)$；当矩阵求逆过程中使用的矩阵乘法使用文献[3]中的 Strassen's 方法时，可以将上述复杂度降低到$O((LJ)^{2.807})$；进一步地，当求逆过程结合了文献[4]中提出的 Coppersmith–Winograd 方法时，理论上可以将上述复杂度降低到$O((LJ)^{2.376})$，对此部分的说明可参考附录四；
- 求解各个$\boldsymbol{W}_k$过程中的矩阵乘法的计算复杂度。请注意，将$\boldsymbol{W}_k$拆解为若干子矩阵的过程并不涉及计算复杂度，一般认为可以通过$\boldsymbol{W}_k$直接获取$\boldsymbol{W}_{j,k}$。

如果使用以上方法，当矩阵组$\boldsymbol{H}=\{\boldsymbol{H}_{j,k}\}$的维度，或者，矩阵组内各个矩阵$\boldsymbol{H}_{j,k}$的维度继续提升时，整个系统将承受愈发巨大的计算负担。为此，需要通过恰当的建模方法，使得在近似获得输出结果的同时明显地降低计算复杂度。

在本题目中，利用相关矩阵组的关联性降低计算复杂度可以从如下基础方向（或你认为其他更合适的方向）中的一个或者多个方向切入完成建模题目：

1）利用矩阵组$\boldsymbol{H}$内部各个矩阵间的关联性，减少前述一组数学运算的整体计算复杂度。例如，矩阵组$\boldsymbol{H}$中包括存在关联性的 5 个矩阵$\{\boldsymbol{H}_{j,1},\boldsymbol{H}_{j,2},\boldsymbol{H}_{j,3},\boldsymbol{H}_{j,4},\boldsymbol{H}_{j,5}\}$，部分情况下可以仅针对$\boldsymbol{H}_{j,1}$、$\boldsymbol{H}_{j,3}$、$\boldsymbol{H}_{j,5}$进行相应的运算过程并获取$\widehat{\boldsymbol{W}}_{j,1}$、$\widehat{\boldsymbol{W}}_{j,3}$、$\widehat{\boldsymbol{W}}_{j,5}$，然后通过恰当的插值操作获取$\widehat{\boldsymbol{W}}_{j,2}$和$\widehat{\boldsymbol{W}}_{j,4}$，使得$\rho_{l,j,k}(\boldsymbol{W})$均满足前述建模估计精度的需求。

2）利用矩阵$\boldsymbol{H}_{j,k}$内部各个矩阵元素间的关联性，降低奇异值分解过程、矩阵求逆过程的计算复杂度。关于奇异值分解的低计算复杂度实现，方法不做限定。例如，可以参考文献[5]中提出的随机奇异值分解（Randomized SVD）方法，其简要实现过程可以参考网页[6]，部分摘录于附录五。

3）针对计算流程进行合理的构造和转化，避免执行相对复杂的矩阵运算步骤。例如，矩阵求逆的计算复杂度较高，而本问题的最终需求为$\boldsymbol{W}_k=$

$\boldsymbol{V}_k(\boldsymbol{V}_k^H\boldsymbol{V}_k+\sigma^2\boldsymbol{I})^{-1}$，如果将该步骤转化为$(\boldsymbol{V}_k^H\boldsymbol{V}_k+\sigma^2\boldsymbol{I})\boldsymbol{W}_k^H=\boldsymbol{V}_k^H$形式来求解$\boldsymbol{W}_k$，则潜在地可以降低整体计算流程的计算复杂度。

4）请注意，在完成本问题时，推荐根据上述表格给出更细化的计算复杂度评估。例如，统计建模过程中复数乘法的使用次数、复数加法的使用次数等，然后分别与复数乘法的计算复杂度（可以为 14）及复数加法的计算复杂度（可以为 2）相乘，累加后得到总体计算复杂度。

存储复杂度定义为矩阵组$\boldsymbol{H}$和$\boldsymbol{W}$占用的存储空间的大小，以比特为单位计算。对于复数矩阵中单个复数元素，其实部和虚部均采用 32 比特单精度浮点表示。因此，整个矩阵组$\boldsymbol{H}$和$\boldsymbol{W}$占用的存储空间分别为$64MNJK$比特和$64NLJK$比特。

为了节省存储开销，考虑对$\boldsymbol{H}$和$\boldsymbol{W}$分别进行压缩。独立设计$\boldsymbol{H}$和$\boldsymbol{W}$的压缩函数$P_1(\cdot)$和$P_2(\cdot)$，对压缩后的数据$P_1(\boldsymbol{H})$和$P_2(\boldsymbol{W})$进行存储。压缩函数的设计请主要考虑挖掘并利用矩阵内部或矩阵间的关联性，或者，矩阵表达的稀疏性；思路可参考但不限于图像的变换域压缩算法或视频的帧间压缩算法。压缩后的每个元素仍然以 32bit 单精度浮点数存储，不考虑位宽的压缩。

$\boldsymbol{H}$和$\boldsymbol{W}$以压缩的形式存储在存储器中。在进行处理之前，需要从存储器中读出$P_1(\boldsymbol{H})$和$P_2(\boldsymbol{W})$进行解压缩处理：

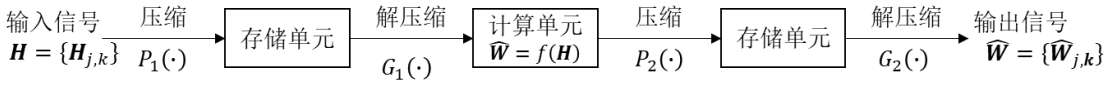
$$\widehat{\boldsymbol{H}}=G_1\big(P_1(\boldsymbol{H})\big),\ \widehat{\boldsymbol{W}}=G_2\big(P_2(\boldsymbol{W})\big)$$

其中，$G_1(\cdot)$和$G_2(\cdot)$分别表示矩阵组$\boldsymbol{H}$和$\boldsymbol{W}$的解压缩函数。经过压缩和解压缩之后得到的$\widehat{\boldsymbol{H}}$和$\widehat{\boldsymbol{W}}$与原始的$\boldsymbol{H}$和$\boldsymbol{W}$存在误差，定义为

$$err_{\boldsymbol{H}}=10*\log_{10}\frac{E\left\{\left\|\widehat{\boldsymbol{H}}_{j,k}-\boldsymbol{H}_{j,k}\right\|_F^2\right\}}{E\left\{\left\|\boldsymbol{H}_{j,k}\right\|_F^2\right\}},\ err_{\boldsymbol{W}}=10*\log_{10}\frac{E\left\{\left\|\widehat{\boldsymbol{W}}_{j,k}-\boldsymbol{W}_{j,k}\right\|_F^2\right\}}{E\left\{\left\|\boldsymbol{W}_{j,k}\right\|_F^2\right\}}$$

其中，$\|\cdot\|_F$表示矩阵的 Frobenius 范数；$E\{\cdot\}$表示求期望运算，即在所提供矩阵组的所有矩阵求平均。注意：标准输出矩阵组$\boldsymbol{W}$是 0 相位，即其中矩阵$\boldsymbol{W}_{j,k}$的每个列向量的首个元素是 0 相位复数（实数）。若建模得到的$\widehat{\boldsymbol{W}}_{j,k}$带有随机初始相位，可增加一步相位拉齐处理，将$\widehat{\boldsymbol{W}}_{j,k}$拉齐到与$\boldsymbol{W}_{j,k}$相同的 0 相位，然后再计算误差。

综上所述，相关矩阵组的整体处理流程如下图所示。



三、建模问题

输入矩阵组$\boldsymbol{H}$、标准中间矩阵组$\boldsymbol{V}$和标准输出矩阵组$\boldsymbol{W}$的数据及其维度如下，数据采用十进制格式：

- 第一组：$M=4, N=64, L=2, J=4, K=384, \sigma^2=0.01$，详细数据见附件；
- 第二组：$M=4, N=64, L=2, J=4, K=384, \sigma^2=0.01$，详细数据见附件；
- 第三组：$M=4, N=64, L=2, J=4, K=384, \sigma^2=0.01$，详细数据见附件；
- 第四组：$M=4, N=64, L=2, J=4, K=384, \sigma^2=0.01$，详细数据见附件；
- 第五组：$M=4, N=64, L=2, J=4, K=384, \sigma^2=0.01$，详细数据见附件；
- 第六组：$M=4, N=64, L=2, J=4, K=384, \sigma^2=0.01$，详细数据见附件；

附件中提供.mat 及.csv 文件格式的数据，按需使用其中的一种文件格式即可。

请基于以上提供的数据，采用适当的方法，解决以下相关矩阵组的低复杂度计算和存储建模问题。注意，提交结果及论文需要完成以下问题 1、问题 2 中的至少一题，同时完成两题将适当加分。问题 3 为开放式问题，不作为必选，但鼓励尝试，有新意的算法模型设计将得到加分。

在完成以下问题时，仅需考虑各个问题本身申明的建模需求，不需要考虑其他问题产生的建模需求。例如，在完成问题 3 时，**不需要**额外考虑问题 1 中$\rho_{\min}(V)$

的建模估计精度需求。

问题 1：相关矩阵组的低复杂度计算

1）基于给定的所有矩阵数据$\boldsymbol{H}$，分析其数据间的关联性，设计相应的近似分析模型$\widehat{\boldsymbol{V}} = f_1(\boldsymbol{H})$，在满足$\rho_{\min}(\boldsymbol{V}) \geq \rho_{th} = 0.99$的情况下，使得根据表格计算的总计算复杂度最低。

- 注 1：在本问题中，暂不考虑矩阵数据$\boldsymbol{H}$的压缩与解压缩。
- 注 2：参考下述 2），当采用直接设计分析模型$\widehat{\boldsymbol{W}} = f(\boldsymbol{H})$的方式时，1）可以跳过不做。

2）基于给定的所有矩阵数据$\boldsymbol{H}$，分析其数据间的关联性，综合设计相应的近似分析模型$\widehat{\boldsymbol{W}} = f(\boldsymbol{H})$，在满足$\rho_{\min}(\boldsymbol{W}) \geq \rho_{th} = 0.99$的情况下，使得根据表格计算的总计算复杂度最低。

- 注 1：在本问题中，暂不考虑矩阵数据$\boldsymbol{H}$的压缩与解压缩。
- 注 2：在本问题中，可以：
  - 在 1）的基础上考虑$\widehat{\boldsymbol{W}} = f_2(\widehat{\boldsymbol{V}})$，因此$\widehat{\boldsymbol{W}} = f(\boldsymbol{H}) = f_2(f_1(\boldsymbol{H}))$；或者，
  - 不需考虑中间结果$\boldsymbol{V}$，直接设计分析模型$\widehat{\boldsymbol{W}} = f(\boldsymbol{H})$，此时 1）可以跳过不做。
- 注 3：如果使用常规数学方法，则考虑所有运算过程涉及的计算复杂度；如果使用人工智能方法，则优先考虑推理阶段的计算复杂度，但需要充分考虑泛化性。

问题 2：相关矩阵组的低复杂度存储

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，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，复杂度计算同问题 1。

- 注 1：压缩与解压缩函数的设计方法不做限定。
- 注 2：计算复杂度考虑压缩和解压缩函数的所有运算过程。
- 注 3：在本问题中，暂不考虑分析模型$\widehat{\boldsymbol{W}} = f(\boldsymbol{H})$对输出矩阵组$\boldsymbol{W}$的影响。
- 注 4：两个建模优化目标（存储复杂度，压缩与解压缩的计算复杂度）的优先级相同。

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。

- 注 1：在本问题中，可以考虑从矩阵输入信号$\boldsymbol{H}$到近似矩阵输出信号$\widehat{\boldsymbol{W}}$为一个整体流程，而非依次经过“压缩/解压缩、计算单元、压缩/解压缩”的过程。
- 注 2：如果使用人工智能方法，需要考虑人工智能模型的存储复杂度以及推理阶段的计算复杂度。

参考文献

[1] Swartzlander, Earl E., and Hani H. Saleh. "Floating-point implementation of complex multiplication." *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*. IEEE, 2009.

[2] 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.

[3] Strassen, Volker. "Gaussian elimination is not optimal." *Numerische mathematik* 13.4 (1969): 354-356.

[4] Coppersmith, Don, and Shmuel Winograd. "Matrix multiplication via arithmetic progressions." *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987.

[5] Halko, Nathan, Per-Gunnar Martinsson, and Joel A. Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions." *SIAM review* 53.2 (2011): 217-288.

[6] 网址：https://gregorygundersen.com/blog/2019/01/17/randomized-svd/