**《数值代数》第三次上机作业 实验报告**

匡亚明学院 211240021 田铭扬

**摘要**

笔者使用C++语言（利用OpenBLAS库）编程，对MGS、Householder变换、Givens变换等种QR分解方法的表现进行了比较，并且比较了它们与法方程组方法等在求解最小二乘(LS)问题时的表现。

**正文**

**前言**

在统计等领域的实际应用中，常常需要“求解”方程数多于变量个数的矛盾方程组，即最小二乘(LS)问题。对系数矩阵进行QR分解是求解LS问题的一种方便的方法，而在其中诞生了修改的Gram-Schmidt正交化(MGS)、Householder镜像变换、Givens平面旋转变换等经典的算法。

本次数值实验的目的，即是对上述算法的误差表现、运行效率等进行比较。这有助于提高对于这些经典算法的认识的深度，对于《数值代数》课程的学习和未来在计算数学应用领域的进一步学习，都有重要意义。

**问题**

**·第1题** 随机构造100个5000∼10000阶的可逆方阵，分别利用CGS方法、MGS方法、Householder方法和Givens方法给出相应的QR分解。统计和比较它们在列正交性、CPU时间以及向后稳定性表现方面的差异。

**·第2题** 以500为间隔，将n从500增加到2500；利用不同方法（法方程组、扩展法方程组、MGS、Householder和Givens）求解最小二乘问题：  
并绘制计算机时和数值精度关于 n 的关系。这里，要求  
其中U是有限（1000∼2000）个随机生成的Given平面旋转阵乘积，  
真解为。

**·第3题** 参考讲义中的图 3.2.1和图 3.4.2，（可选取不同的例子）进行数值计算和绘制其效果。

**程序设计**

大部分算法都可以按照讲义[1]或教材[1]上的讲解按步骤，故在此不再详述。但本次的几种算法都有很大的优化空间，能够将计算速度（系数级）大幅提高，具体可以优化的内容如下：

1.(通用)由于第1题要求对同一个矩阵使用不同算法进行对比，算法执行时不能修改原矩阵的值，需要开辟新的内存空间储存结果。算法执行前“一次性”将原矩阵复制，相算法进行时在一处读、另一处写，能节省一定时间。

2.(CGS/MGS/Householder)注意MGS算法需要反复使用直交阵Q的列向量，因而将Q改为按列存储，能将耗时缩减到原来的1/5左右；同理，Householder要多次访问矩阵矩阵R（最初由原矩阵复制而来）的列向量，将其改为列储存也能节省一定的时间。**以上两点聚焦于削减“跳跃式”访问内存的开销。**

3.(Householder/Givens)由于原矩阵会随着算法的进行逐渐变成上三角矩阵，**减少“乘0”的操作也能节省很多无效的计算。**具体而言，执行Householder变换时，矩阵R只需对右下角进行秩1修正（如需还原矩阵Q，也只需对下半部分秩1修正）；执行Givens变换时，矩阵R不需对当前列左侧地部分进行操作。这样为两种算法的相应部分，分别减少了2/3与1/2的操作。

此外还有一些注意事项：

1.第1题由于需要对比各算法所得的矩阵Q的直交性，H方法、G方法需在执行时对单位矩阵执行相同的变换以生成Q。但是在实际应用（如第2题）中Q的计算不是必须的，故不应记入算法的耗时。（事实上，由于Givens方法还原矩阵Q只涉及行向量的操作，而Householder涉及矩阵—向量乘法，后者的内存开销更大，因而若记入还原Q的耗时，会出现H算法耗时比G算法更多的“奇特”结果，相应图表见“实验结果分析”部分。）

2.MGS算法中会出现向量数乘的操作。本想利用软件包对“axpy”操作的并行优化，将操作改由实现，却导致算法的直交性和向后误差都变差了2个数量级。这是因为为矩阵列向量的模，矩阵阶数较大时也会较大，导致出现了“大数减小数”的精度损失。

3.第2题中，可以将右端向量储存在系数矩阵的最后一列，与系数矩阵一起进行H、G等变换。这样能使代码的编写得到简化。

最后，由于机器性能限制，第1题要求的“5000∼10000阶可逆方阵”耗时过长，故改用500~2000阶的矩阵进行预算与比较。生成可逆矩阵的方法是：  
 <1> 随机取阶数n，生成n阶数量矩阵，对角线元素取100；  
 <2> 执行sqrt(n)次一下操作：  
 <3> 将0,1,…,n-1执行一次“洗牌”[复杂度O(n)]，排列成a0,…,an-1;  
 <4> 对于角标 k从0到(n-1)/2，将第a2k乘以b加到第a2k+1行上，b取  
 [-1,-0.2]∪[0.2,1]之间的随机数。  
以上算法能保证随机矩阵的稠密性，并且保证各元素的大小较为适宜。

**实验环境**

使用虚拟机软件VMWare 17运行deepin 20.9操作系统，为其分配8GB内存，同时开启了Intel VT-x/EPT和IOMMU选项，以提升虚拟机性能。使用了OpenBLAS库，对向量内积、赋值、求范数等运算在8个CPU核心上进行并行加速。使用GCC 8.3.0-1 release编译器。

**实验结果分析**

**第一题**

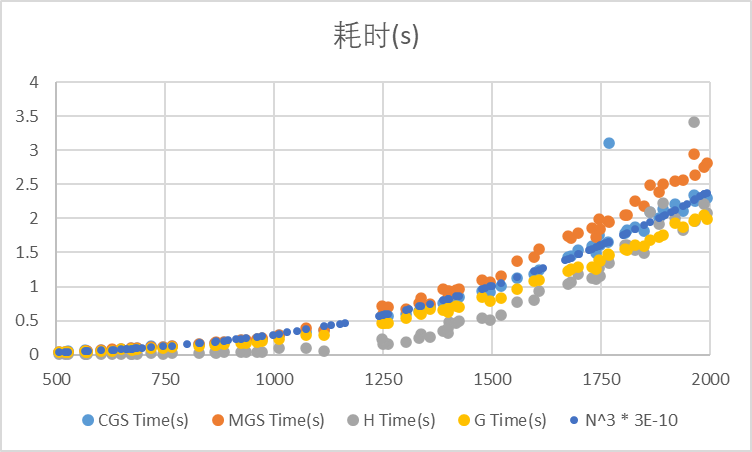
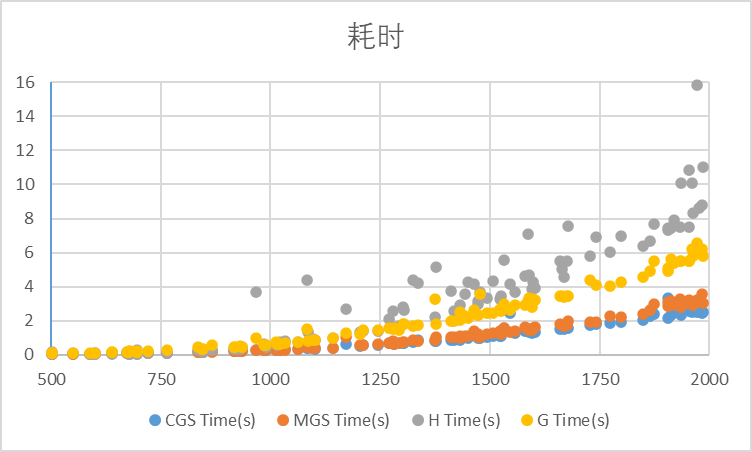
****

图1、2：CGS、MGS、H、G方法的耗时对比图，包含(左图)/不包含(右图)还原直交阵

由于CGS和MGS的计算过程必须将直交阵Q求出，故可以视为“不变量”，作为H、G方法的参照。（图2中还画出一条三次函数曲线，作为零一参照）

如图所示，包含还原Q的耗时 H＞G＞MGS＝CGS，相较于理论（H＜GS＜G）出现较大差异，其原因已在“程序设计”部分进行过了分析。但改为不记还原Q的耗时，得到的结果也与理论不太相符。具体而言：  
1.在矩阵阶数较小时，耗时GS＞G＞H；  
2.O(n3)的增长率大体没有问题，但H方法的好耗时在阶数较大时显著增加；  
3.在矩阵阶数较大时，耗时MGS＞CGS＝H＞G。  
 以上现象出现的原因暂不清楚。在对算法优化的过程中，笔者注意到内存开销对于算法复杂度的“系数”（即在乘除法次数的阶数相同时）有显著的影响影响，这是笔者认为最有可能的原因。

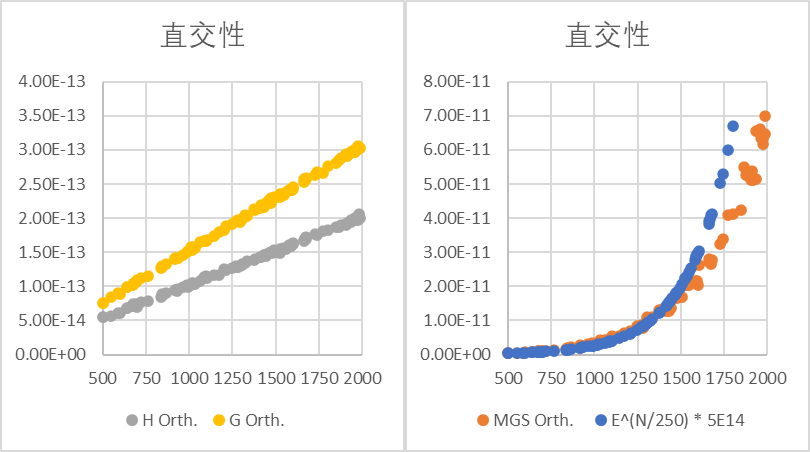
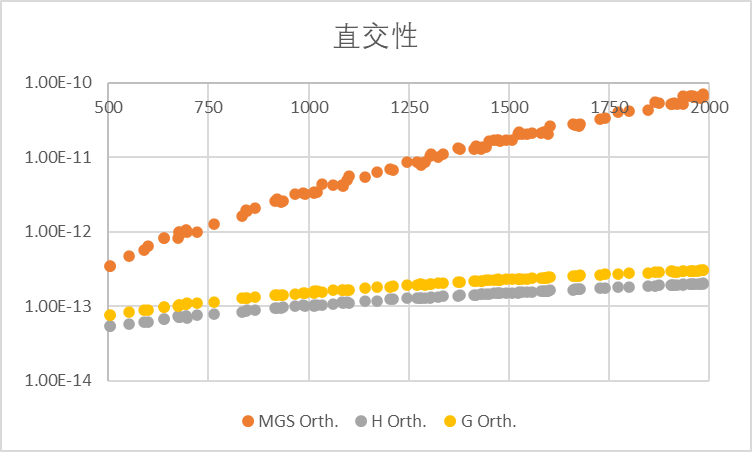


图3、4、5：MGS、H、G方法的直交性对比图

上图数据为矩阵的F-范数，以此反应Q的直交性。MGS与CGS方法的数值结果在保留5位有效数字时完全相同，暂时（见第3题）猜测是因为系数矩阵的条件数较好，因而未体现出CGS直交性的劣势。故图表中只显示MGS。

从图中可以看出，H、G方法的直交性表现类似，G方法略好；而GS方法的直交性表现，相比它们有数量级的差距。而且，随着阶数的增长，H、G的直交性是“线性变差”的，GS方法却近乎于指数，这也是其劣势。

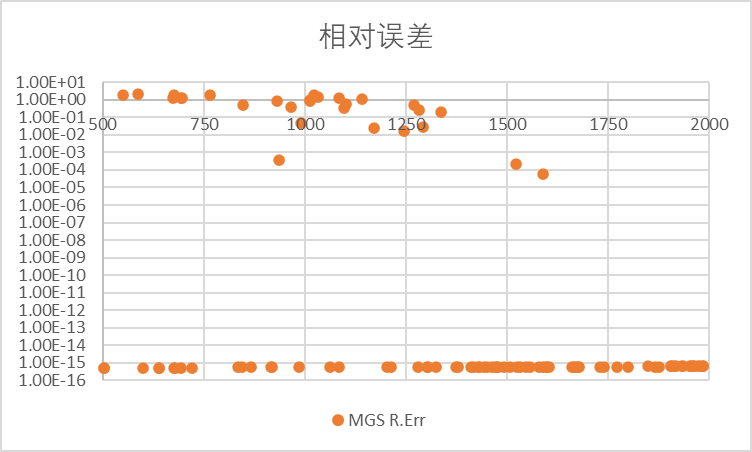
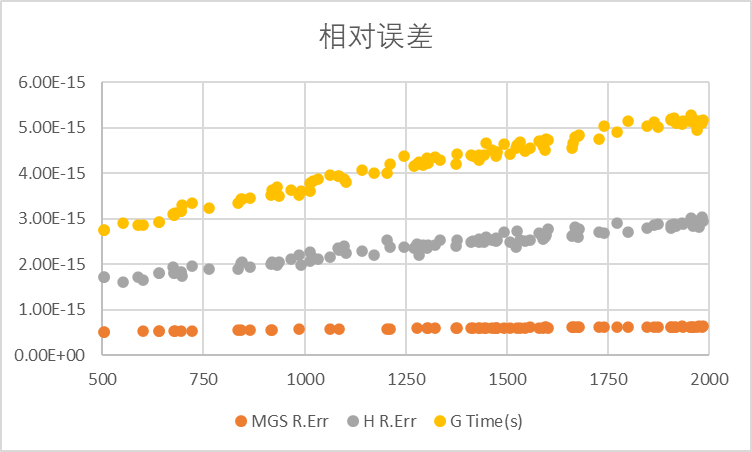


图6、7：MGS、H、G方法的向后误差对比图

上图数据为矩阵的F-范数，以此反应QR分解的向后误差。

注意到大多数情况下的误差表现 GS＜H＜G，三者均随矩阵阶数的增长而线性增加，且三者的差别是系数级别的。但是，MGS算法的后验误差并不稳定，在不少情况下（尤其矩阵结束较小时）有极大的后验误差，具体原因暂不清楚。

综上，在实际应用中，应当根据问题规模、对计算精度的要求、对计算速度的要求等方面，在G、H中选择一种算法使用。（另外，虽然题中未要求比较算法的内存占用，但简单分析可知，在三种算法中，H方法的内存占用是最低的。）

**第二题**

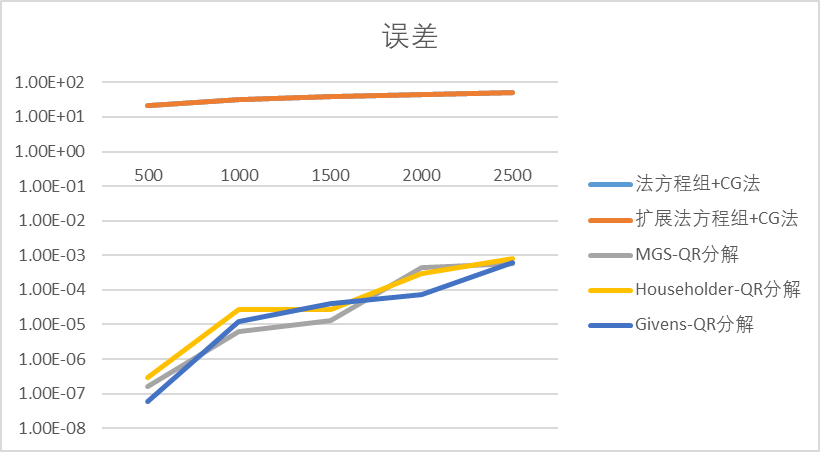
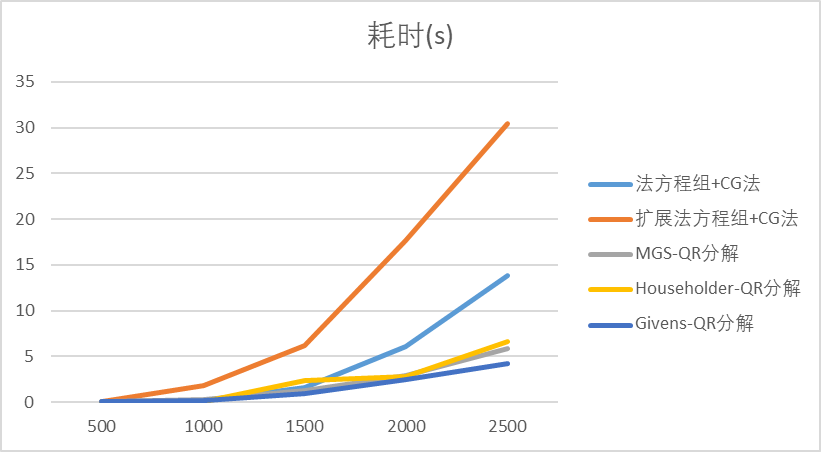
****

图8、9：五种方法求解最小二乘问题的耗时、误差对比

使用法方程组、扩展法方程组方法求解LS问题时，均使用CG算法，以期较好的耗时表现。误差为解与真解之差的2-范数。

如图所示，QR分解方法的耗时表现好于、误差表现远好于解法方程组的方法。此外，在耗时上，Givens略好于另外两种QR方法；而在误差上，三者的表现近似。注意两种法方程组方法的误差表现都极差，事实上，将系数矩阵B变为BTB会使其条件数增长为原来的平方。

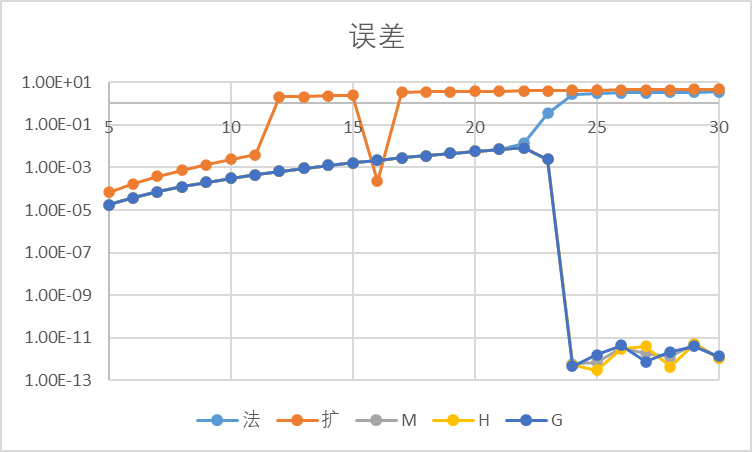


图10：五种方法求解最小二乘问题的误差对比（矩阵阶数较小时）

笔者尝试对低阶矩阵进行了相同的计算，此时两种方法的误差与QR方法在同一数量级[法方程组法可到23阶、扩展法方程组法可到11阶]，这排除是因为代码编写错误导致两种方法误差很大。但是还出现了一个奇怪的现象：在阶数≤23阶时，三种QR方法的误差表现反而一般（10-4量级，与法方程组法相同），在法方程组发失效的同时QR方法的误差表现突然变好（10-13量级）。如此“泾渭分明”的原因还有待进一步探究。

**第三题**

并未得到要求的实验结果。尝试使用对角阵Diag{2-k}、Hilbert矩阵和Vandermonde矩阵，并尝试不使用任何OpenBlas软件包中的指令，或将double精度改为float，CGS与MGS算法均有相同的表现。（以书中示例为例，两种方法得到的上三角阵，其对角元均能达到10-17~10-18量级。）使用Lapack软件包中的指令检查矩阵的条件数，为1017~1019量级，条件数很大。

原因不清楚。猜测可能是编译器进行了底层优化，但未找到相关文献。

**结语**

在本次数值实验中，笔者对MGS、Householder变换、Givens变换三种QR分解方法进行了对比，并比较了它们求解LS问题时与法方程组方法的差距。

在实验中出现了很多与理论不相符的现象：CGS并未体现出很差的直交性，GS、H、G的执行速度也与理论不同。此外，法方程组与QR方法求解LS问题的表现也出现了奇怪的现象。这都有极大的进一步探究的空间。

**实验代码**

由于篇幅限制，代码及原始数据不在实验报告中列出，可以在笔者github仓库中查看。网址为：<https://github.com/lk758tmy/NA2-Codes>。

**参考文献**

[1] 《数值代数》讲义. 张强

[2] 数值计算方法-下册. 林成森. 科学出版社. 2005-1第二版