数值试验报告

袁一杨　141110101

2016-10-27

1　实验一

1.1　基本内容

随机构造一个可逆方阵，利用不同方法给出它的QR分解。观测所得列向量的正交性、CPU时间和所谓的向后稳定行。

1.1.1　CGS：

$$\mu_{11} = \|a_1\|$$
$$q_1 = a_1/\mu_{11}$$
$$\mu_{12} = q_1^\top a_2$$
$$\mu_{22} = \|a_2 - \mu_{12}q_1\|_2$$
$$q_2 = (a_2 - \mu_{12}q_1)/\mu_{12}$$

$$j = 3, \ldots, r \begin{cases} \mu_{ij} = q_i^\top a_j, i = 1, 2, \ldots, j-1 \\ \mu_{jj} = \|a_j - \sum_{i=1}^{j-1} \mu_{ij} q_i\|_2 \\ q_j = (a_j - \sum_{i=1}^{j-1} \mu_{ij} q_i)/\mu_{jj} \end{cases}$$

$$\mu_{ij} = q_i^\top a_j, i = 1, 2, \ldots, r, j = r+1, \ldots, n$$

1.1.2　MGS：

$$A = [a_1^{(1)}, \ldots, a_n^{(1)}]$$

$$\mu_{11} = \|a_1\|$$
$$q_1 = a_1/\mu_{11}$$
$$\mu_{1j} = q_1^\top a_j, j = 2, \ldots, n$$
$$a_j^{(2)} = a_j^{(1)} - \mu_{1j} q_1$$
$$\mu_{22} = \|a_2^{(2)}\|_2$$
$$q_2 = a_2^{(2)}/\mu_{22}$$
$$\mu_{2j} = q_2^\top a_j^{(2)}$$
$$a_j^{(3)} = a_j^{(2)} - \mu_{2j} q_2$$
$$\vdots$$

1.1.3 Householder：

取

$$x = a_1^{(1)}, y = \alpha_1 e_1$$

, 其中

$$\alpha_1 = -sign(a_{11})\sigma_1, \sigma_1 = (\sum_{i=1}^{m} a_{i1}^2)^{\frac{1}{2}}$$

令

$$u_1 = a_1^{(1)} - \alpha_1 e_1, H_1 = I_m - b_1^{-1} u_1 u_1^\top$$

其中

$$b1 = \frac{1}{2}\|u_1\|_2^2 = \alpha_1^2 - \alpha_1 a_{11}$$

$$H_1 A = \begin{bmatrix} H_1\vec{a_1} = \alpha_1 e_1 & H_1\vec{a_2} & \ldots & H_1\vec{a_n} \end{bmatrix} = \begin{bmatrix} \alpha_1 & * & \ldots & * \\ 0 & & & \\ \vdots & & \widetilde{A_2} & \\ 0 & & & \end{bmatrix}$$

令

$$\widetilde{H_2} = I_{m-1} - b_2^{-1} u_2 u_2^\top$$

则

$$H_2 = \begin{bmatrix} =1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \widetilde{H_2} & \\ 0 & & & \end{bmatrix}$$

$$H_2H_1A = \begin{bmatrix} \alpha_1 & * & * & \dots & * \\ & \alpha_2 & * & \dots & * \\ & 0 & & \widetilde{A_3} & \\ & \vdots & & & \\ & 0 & & & \end{bmatrix}$$

最后得到

$$H_sH_{s-1}H_{s-2}\dots H_2H_1A = \begin{bmatrix} \alpha_1 & * & \dots & * & * & \dots & * \\ & \alpha_2 & \dots & * & * & \dots & * \\ & & \ddots & * & * & \dots & * \\ & & & \alpha_r & * & \dots & * \\ & 0_{(m-r)\times r} & & & & 0 & \end{bmatrix}$$

1.1.4 Givens：

记 $c = \cos\theta$和 $s = \sin\theta$，利用正交矩阵

$$\begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & \ddots & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}$$

将每一个 $\widetilde{A_i}$第一列的 $i+1:n$的分量变换为零。

若 $x_j = 0$，则 $c = 1, s = 0$;

若 $|x_j| \geq |x_i|$，通常取 $s > 0$，即 $t = \frac{x_i}{x_j}, s = \frac{1}{\sqrt{1+t^2}}, c = st$

若 $|x_j| < |x_i|$，通常取 $c > 0$，即 $t = \frac{x_j}{x_i}, c = \frac{1}{\sqrt{1+t^2}}, s = ct$

最后将矩阵化为

$$\begin{bmatrix} \alpha_1 & * & \dots & * & * & \dots & * \\ & \alpha_2 & \dots & * & * & \dots & * \\ & & \ddots & * & * & \dots & * \\ & & & \alpha_r & * & \dots & * \\ & 0_{(m-r)\times r} & & & & 0 & \end{bmatrix}$$

1.2 数据

取 $n = 50$,hilbert矩阵，循环500次求均值。

方法	列向量正交性	CPU时间	向后稳定行
CGS	42.1685	0.0159	$8.4265e^{-17}$
MGS	1.0000	0.0044	$8.5454e^{-17}$
Householder	$1.5428e^{-15}$	0.0381	$5.7529e^{-09}$
Givens	$1.5350e^{-15}$	0.0058	$4.2057e^{-16}$

1.3 结论

从运行的结果，可以得出以下一些结论：

1. 可以看出Householder和Givens的正交性要比CGS和MGS好很多。
2. 尽管CGS和MGS在计算上只是换了一下运算的顺序，但是从结果可以看出，CGS的正交性明显很差，MGS的正交性要好很多。
3. 从正交性、向后稳定行，以及运算时间上来看，MGS算法要优于CGS算法。
4. MGS和Givens的计算起来明显要快很多。MGS算得快，有一部分原因来自它主要采用内积运算，比矩阵乘法要快。Givens算的快很大程度上是由于本实验中的矩阵为稀疏矩阵。

2 实验二

2.1 基本内容

实现本章的两张图。

2.1.1 对角线比较：

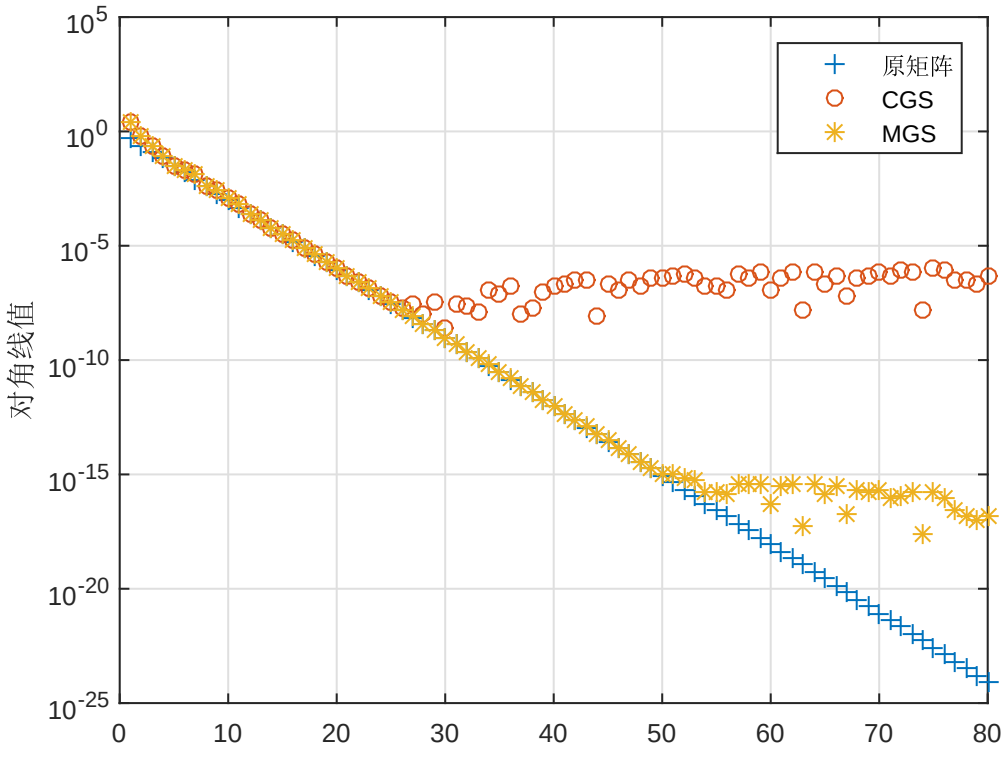
CGS和MGS比较

2.1.2 图片压缩：

step 1 load clown.mat
step 2 [U,S,V]=svd(X)
step 3 colormap('gray')
step 4 image(U(:,1:k)*S(1:k,1:k)*V(:,1:k)

2.2 数据

2.2.1 对角线比较



矩阵 $Udiag(2^{-k})_{k=1}^{80}V^{\top}$的对角线元素比较

2.2.2 图片压缩

见图1

2.3 结论

1. 若计算是极为精确的，CGS和MGS分解的对角线应该和原方程组完全重合。

2. 在图中，CGS的对角线元素在近似于机器精度开根号的位置不再变化，说明更加精确的对角线元素已经算不出来了。而MGS的对角线则走的更远，说明MGS算法更加稳定更加精确。

2.3.1 图片压缩

通过和原图比较，发现 $k \leq 30$时当图片的清晰程度变化较明显，当取到60个奇异值的时候，图片的清晰度就已经和原图差不多了。而拆分将图片进行奇异值分解之后有200多个奇异值。因此，可以看出，利用奇异值进行图片压缩可以很好地减少图片的储存空间。

3 实验三

3.1 基本内容

考虑列满秩的最小二乘问题$A_{n\times(n-1)}x_{n-1} = b_n$，其中系数矩阵和右端项分别为

$$A_{n\times(n-1)} = T_n(1:n, 1:n-1) = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \\ & & & & -1 \end{bmatrix}$$

$$b_n = (1 + \frac{1}{n}, \frac{2}{n}, \frac{3}{n}, \ldots, \frac{n-2}{n}, 1 + \frac{n-1}{n}, 0)^\top$$

对应的最小二乘解为 $x_{LS} = (1, 1, 1, \ldots, 1, 1)^\top$.分别用 (a)法方程组； (b)MGS方法； (c)Householder法； (d)Givens法，四种方法求解这个最小二乘问题

令n从 5增加到 30，绘图比较上述四种算法的计算工作量（可用CPU时间表示）和计算精度与n的关系。

3.2 实验目的

通过将法方程组方法，MGS法、Householder法，Givens法比较，同时对三种方法的效率有一个直观的了解。

3.3 数据及分析

见图2、3、4、5

3.4 结论

1. 由图2，可以发现，随着矩阵阶数由 5 ∼ 30不断增加，四种方法的的误差都在不断增加，其中Householder的误差增长速度最快，法方程组法其次，MGS和Givens看起来相对稳定。

2. 我们将法方程组，MGS，Givens方法单独进行 5 100阶的误差统计，如图4、5和3。很明显可以看出，法方程组法和MGS 的误差增长的越来越快，而Givens法最后的误差会趋于稳定，为 $1e^{-12}$量级，法方程组的误差为 $1e^{-10}$级，MGS的为 $1e^{-11}$量级并且由趋势来看还会不断增加。因为法方程组求解的过程中用到了cholesky分解，该方法中涉及开根号、除法运算，因此误差较大，随着矩阵阶数的不断增加计算次数增加，再加上解法方程组的计算量本身就很大误差会不断地积累，导致增加得越来越快。

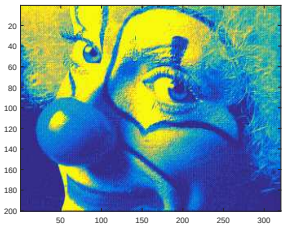
3. Householder的误差增长最快，由于在Householder的计算过程中，不断地使用开根号和除法运算。

4. 计算工作量方面，随着矩阵阶数的不断增大，CPU运行时间也在不断增加。Givens和法方程组的趋势大致相同（除去一些波动），MGS法的时间最少。虽然Householder和Givens在计算次数的阶数方面优于剩下两者，但是在编程运算的时候，可能要考虑到一些语句所花费的时间，比如Householder和Givens算法中的实际执行过程中会有一些判断语句，也会算入时间。另外，Matlab本身对于矩阵运算会有一些优化，这也导致了最终呈现出来的CPU时间和计算量不相符。

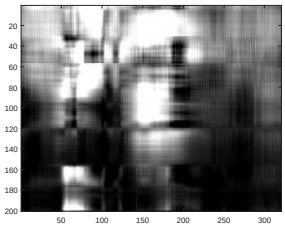
4 附录

1. cpu运行时间有波动。CPU的运行时间受很多因素的影响，因此本次实验的程序运行皆保证计算机在一个相对稳定的情况下进行的。

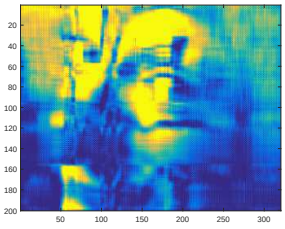
2. 矩阵比较特殊化，为稀疏矩阵。尽管理论上Householder算法的复杂度要小于Givens算法，但是在本次试验中，矩阵为稀疏矩阵，这就使得Givens算法在误差以及工作量上表现的都很好。



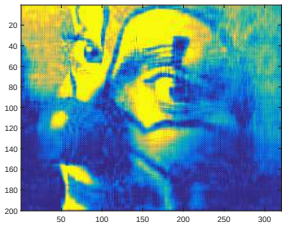
(a) 原图



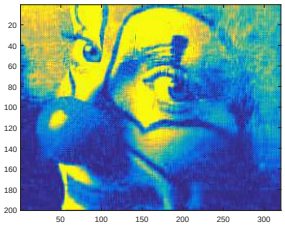
(b) $k = 5$



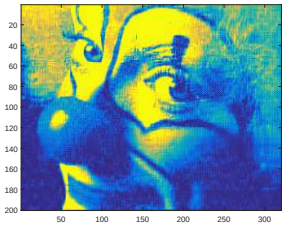
(c) $k = 10$



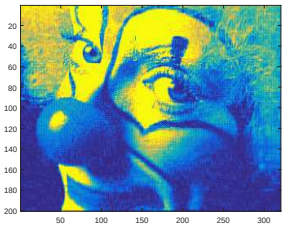
(d) $k = 20$



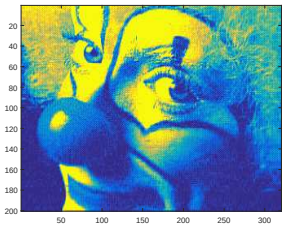
(e) $k = 30$



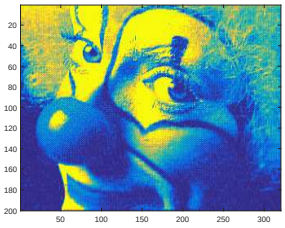
(f) $k = 35$



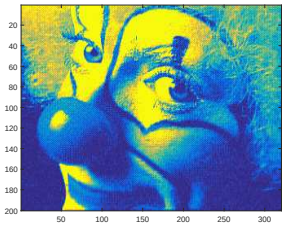
(g) $k = 50$



(h) $k = 60$

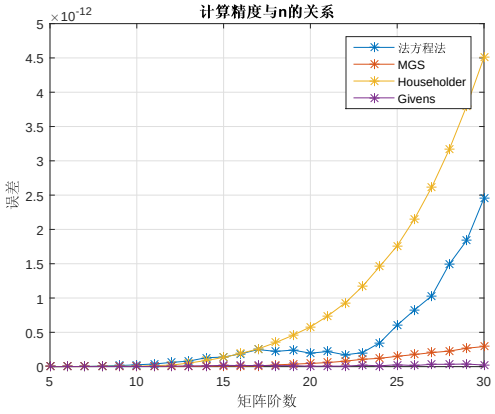


(i) $k = 70$

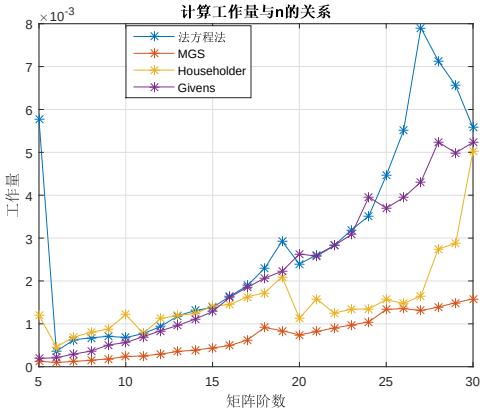


(j) $k = 80$

Figure 1: 奇异值分解运用于图像压缩



(a) 精确度



(b) 工作量

Figure 2: 工作量和精确度随矩阵阶数的变化

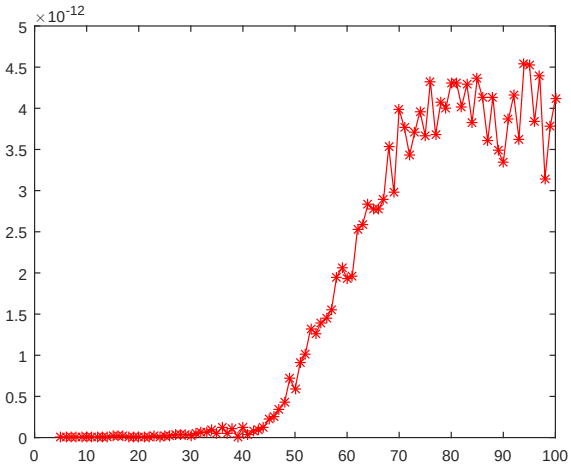


Figure 3: Givens误差变化

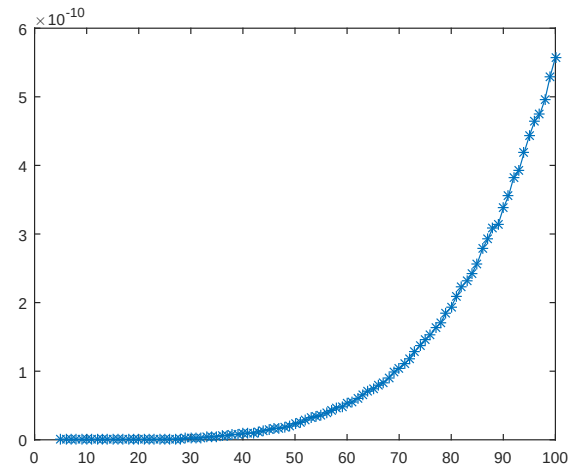


Figure 4: 法方程组法误差变化

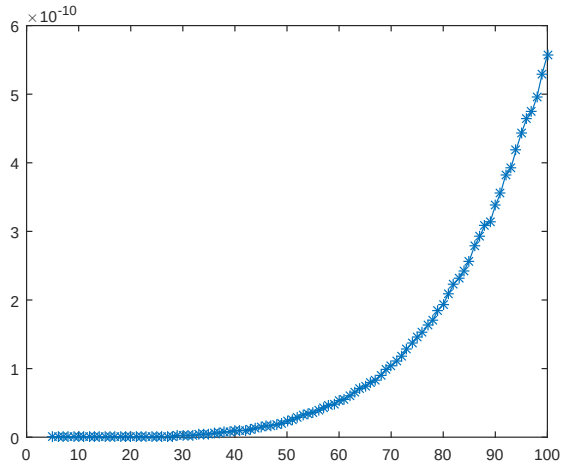


Figure 5: MGS误差变化