深 圳 大 学 实 验 报 告

实验课程名称：最优化方法

实验项目名称：矩阵 QR 分解

学院：计算机与软件学院　专业：软件工程（腾班）

报告人：黄亮铭　学号：2022155028　班级：腾班

同组人：

指导教师：李炎然

实验时间：2024 年 11 月 08 日 – 2024 年 12 月 05 日

实验报告提交时间：2024 年 12 月 05 日

教务处制

实验报告包含内容

一、 实验目的与要求

1. 熟练掌握 QR 分解 Gram－Schmidt 方法；
2. 掌握 Householder 方法；
3. 能够判断矩阵是否可逆，并求出其逆矩阵。

二、 问题

读取附件MatrixA.mat文件中的矩阵A，利用Gram－Schmidt(GS)算法对A进行QR分解，GS的Matlab代码如1所示。

(1) 验证GS是否能稳定进行QR分解矩阵A，其Q矩阵是否正交？

```
[m,n] = size(A);        % 读取矩阵A的大小
Q = zeros(m,n);         % 初始Q矩阵
R = zeros(n,n);         % 初始R矩阵

for k=1:n
    R(1:k-1,k) = Q(:,1:k-1)'*A(:,k);      % 实现什么
    v= A(:,k) - Q(:,1:k-1)*R(1:k-1,k);    % ?
    R(k,k) = norm(v);                     % ?
    Q(:,k) = v/R(k,k);                    % ?
end
```

图1. Gram－Schmidt算法的Matlab代码

(2) 实现Householder方法QR分解代码，并验证其对矩阵A分解是否稳定？

(3) 读取附件MatrixB.mat文件的方矩阵B，判断其是否可逆？如果可逆，求其逆矩阵。

三、模型建立及求解

3.1 问题 1

QR 分解是一种将矩阵 A 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的方法，即$A = QR$，形式如下所示。

$$A = (a_1 \quad \dots \quad a_n) = (q_1 \quad \dots \quad q_n) \times \begin{bmatrix} R_{11} & R_{12} & \cdots & \cdots & R_{1\,n} \\ 0 & R_{22} & \cdots & R_{2\,n-1} & R_{2\,n} \\ \vdots & 0 & \ddots & \cdots & \vdots \\ 0 & \vdots & \vdots & R_{n-1\,n-1} & R_{n-1\,n} \\ 0 & 0 & \cdots & \cdots & R_{nn} \end{bmatrix}$$

3.1.1Gram–Schmidt 算法思路

Gram–Schmidt 算法（后称 GS 算法）QR 分解是一种将矩阵 A 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的方法，即$A = QR$。这种方法通过正交化和单位化一组线性无关的向量来实现。

以下是 GS 算法进行 QR 分解的基本步骤：

① 输入数据：矩阵A，$A \in R^{m\times n}$；

② 初始化：新建零矩阵Q、R，其中$Q \in R^{m\times n}$，$A \in R^{n\times n}$；$q_1 = a_1/\|a_1\|_2$。

③ 正交化：$q_i' = a_i - (q_1^T a_i)q_1 \dots - (q_{i-1}^T a_i)q_{i-1}$；

④ 判断是否线性相关：如果$q_i' = 0$，说明线性相关，提前退出迭代；如果$q_i' \neq 0$，说明线性无关，迭代继续。

⑤ 单位化：$q\_i = q_i'/\|q_i'\|_2$。

⑥ 重复步骤③、④和⑤，直到$i = n$。

在实际的算法实现中，我们通常会将初始化步骤中的$q_1 = a_1/\|a_1\|_2$放入到迭代过程中。此外，在迭代中判断是否线性相关时，$q_i' = 0$总是为假。因为在计算机中，0 会使用很小的数字代替，所以，我们在实际的算法中会比较q_i'和eps的大小，其中eps是一个很小的浮点数。

3.1.2GS 算法的稳定性

测试 GS 算法进行 QR 分解的稳定性即测试矩阵 Q 的每一列的正交性是否得到保持。已知：如果矩阵 Q 正交，则对于矩阵 Q 中的每一列均有如下性质：

$$q_i^T q_j = 0, i \neq j$$

$$q_i^T q_j = 1, i = j$$

利用上述性质，我们可以通过公式$Error = \max\limits_{1\le i<k} \left|q_i^T q_k\right|, k \in [2, n]$来测试$q_k$和前面的列的正交性偏差。

3.1.3 测试结果

使用 3.1.1 中的算法求得矩阵 Q 之后，利用 3.1.2 中的公式计算矩阵 Q 的每一列的正交性偏差，得到向量$Error$，然后将$Error$可视化，得到如下图所示的结果。

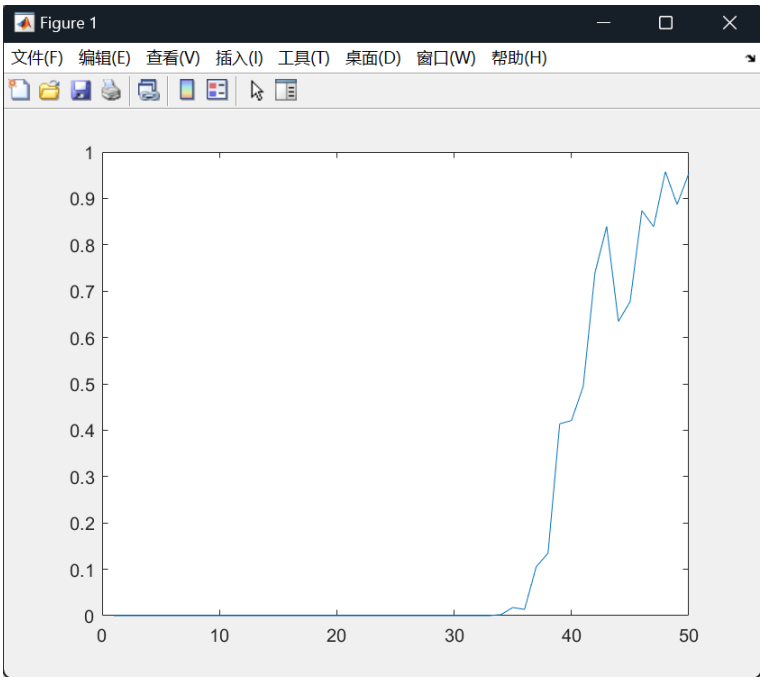

图 1GS 算法 QR 分解的累计误差

3.1.4GS 算法小结

的结果说明随着 k 值增大，正交性偏差也逐渐加大，即矩阵 Q 逐渐失去正交的性质，稳定性降低。造成这种结果的原因是浮点数的舍入误差。当矩阵的维数较小的时候，累计误差较小，可以忽略不计，GS 算法进行 QR 分解的稳定性较好。但是当矩阵的维数较大的时候，累计误差无法忽略，对结果造成比较大的影响。

综上所述，当维数较小（由实验得知以维数 35 为界）时，使用 GS 算法进行 QR 分解的稳定性较好，矩阵 Q 的正交性得到保持；当维数较大时，该算法的稳定性较差，无法保证矩阵 Q 的正交性质。

3.1.5Modified Gram–Schmidt 算法思路/稳定性测试

由上文的分析可知，GS 算法进行 QR 分解时由于浮点数舍入（计算精度）的问题，其计算误差会随着矩阵的维数的增加而累积，矩阵维数越高其稳定性越差。

一种可能减小计算误差的方法如下：在计算当前向量的时候，不仅考虑前面的向量，还需要考虑后面的向量，即 3.1.1 的步骤③正交化需要进一步减去后续的向量。除此之外，Modified Gram–Schmidt 算法（后称 MGS 算法）的思路与 GS 算法的思路基本保持一致。

MGS 算法的稳定性的测试方法与 GS 算法的稳定性的测试方法完全一致，因此这里不再重复。

3.1.6 测试结果

使用 3.1.5 中的算法求得矩阵 Q 之后，利用 3.1.2 中的公式计算矩阵 Q 的每

一列的正交性偏差，得到向量$Error$，然后将$Error$可视化，得到如下图所示的结果。

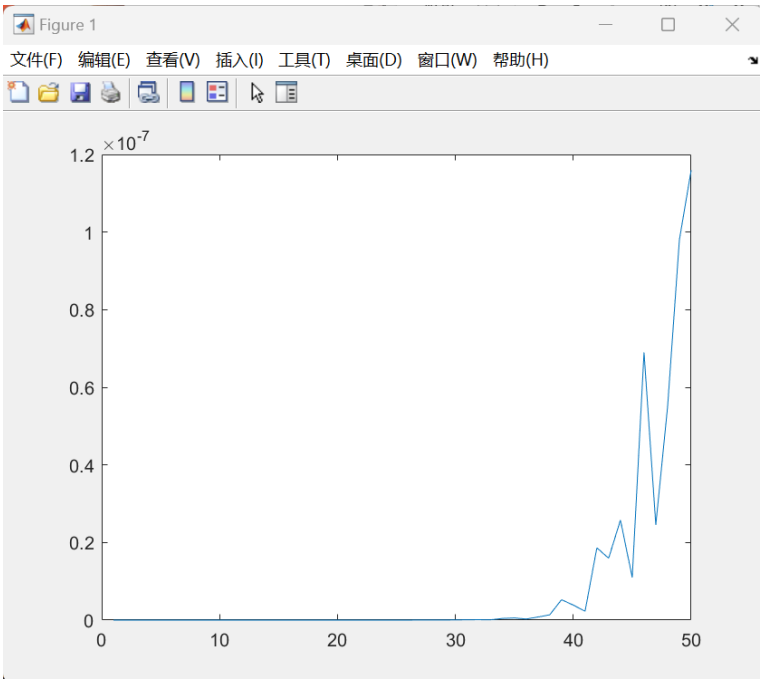

图 2MGS 算法 QR 分解的累计误差

算法和 MGS 算法对比

由两种算法的累计误差对比可知，GS 算法和 MGS 算法的计算误差都会随着矩阵的维数增加而呈现增加的趋势，且增加的速度逐渐变快。但是 MGS 算法考虑了全局，因此累计误差仍然控制在可忽略的范围内；而 GS 算法只考虑了局部，累计误差无法忽略（如下图红框中所示）。

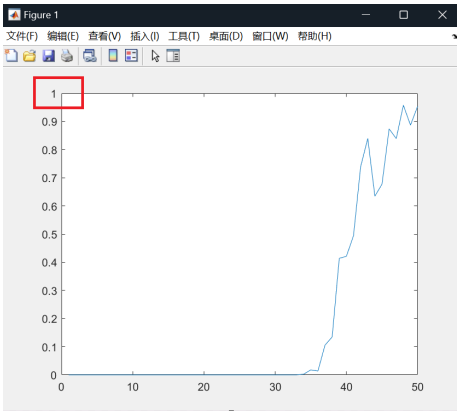

图 3a GS 算法 QR 分解的累计误差

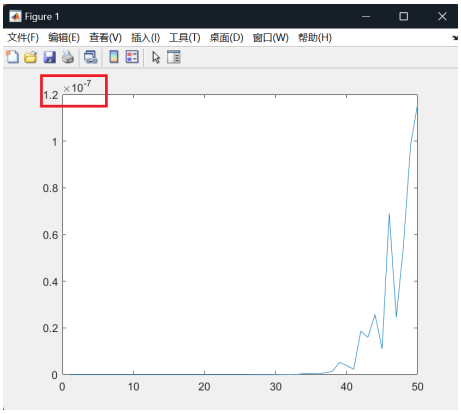

图 3b MGS 算法 QR 分解的累计误差

3.2 问题 2

3.2.1Householder 算法思路

算法 QR 分解是一种通过构造反射算子，利用镜面反射的原理

不断对原矩阵进行三角化实现 QR 分解的方法。与 GS 算法相比，Householder 对舍入误差更有鲁棒性。

$$A = [Q \quad \tilde{Q}]\begin{bmatrix} R \\ 0 \end{bmatrix}$$

$$[Q \quad \tilde{Q}] = H_1 H_2 \cdots H_n$$

图 4QR 因数分解一般形式

Householder 对矩阵 A 进行三角化的一个样例如下图所示。

$$H_n H_{n-1} \cdots H_1 A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

$$A = \begin{bmatrix} X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \\ X & X & X & X & X & X & X & X \end{bmatrix} \xrightarrow{H_1 A} A_1 = \begin{bmatrix} X & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \end{bmatrix} \xrightarrow{H_2 A_1} A_2 = \begin{bmatrix} X & X & X & X & X & X & X & X \\ 0 & X & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \\ 0 & 0 & X & X & X & X & X & X \end{bmatrix}$$

图 5Householder 三角化样例

Householder 算法进行 QR 分解的基本步骤。

输入数据：输入数据：矩阵A，$A \in R^{m \times n}$；

② 初始化：新增单位矩阵Q，$Q \in R^{m \times m}$；

③ 循环体内容 1：令$y = A_{k:m,k} \in R^{m-k+1}$，计算向量$v_k$，其中$v_k = \frac{1}{\|w\|_2} w$，$w = y + sign(y_1)\|y\|_2 e_1$；

循环体内容 2：将$A_{k:m,k:n} \in R^{(m-k+1)\times(n-k+1)}$与反射算子$I - 2v_k v_k^T$相乘；

循环体内容 3：计算H_k，$H_k = \begin{bmatrix} I & 0 \\ 0 & I - 2v_k v_k^T \end{bmatrix}$；

⑥ 循环体内容 4：计算Q，$Q = Q \times H_k$；

⑦ 循环步骤③、④、⑤和⑥，直到$k = m$。

⑧ 将矩阵A赋值给矩阵R。

在代码实现的时候，我们可以直接使用 matlab 库函数中的 qr 分解函数是实现，因为其 qr 分解底层原理是基于 Householder 算法的。这里我们既使用自己代码实现的 Householder 算法进行 QR 分解，也使用 matlab 库函数的 qr 函数进行 QR 分解，然后将两者进行对比。

3.2.2Householder 算法的稳定性测试

测试方法与 3.1.2 的相同，请参考 3.1.2GS 算法的稳定一章。

3.2.3 测试结果

使用 3.2.1 中的算法求得矩阵 Q 之后，利用 3.1.2 中的公式计算矩阵 Q 的每一列的正交性偏差，得到向量$Error$，然后将$Error$可视化，得到如下图所示的结

果.

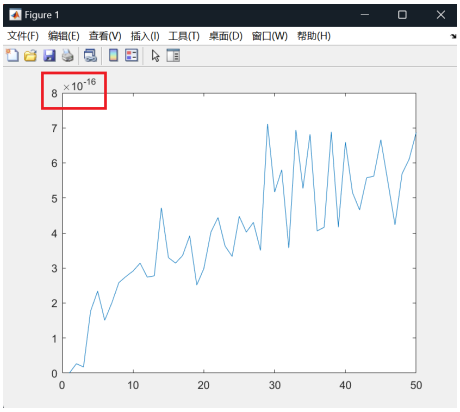

图 6a 自行实现代码 QR 分解的累计误差

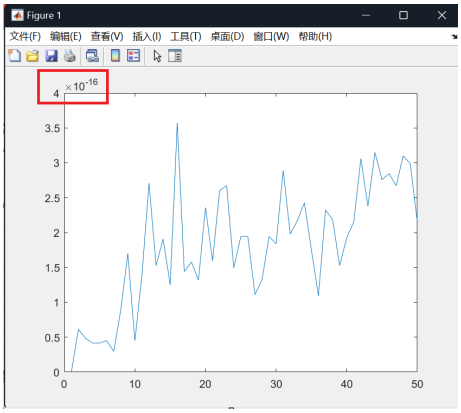

图 6b matlab 库函数 QR 分解的累计误差

3.2.4 小结

观察上述累计误差图，可以发现：当维数小于 30 时，我写的代码实现 QR 分解和 matlab 库函数实现 QR 分解的累计误差比较接近，当维数继续增大时，两者之间的差距开始增大，我实现的 QR 分解的累计误差逐渐增大，而 matlab 库函数实现的 QR 分解的累计误差仍然稳定在一定的范围。两者仍然在同一个数量级上。

两者的累计误差均远远小于 GS 算法和 MGS 算法进行 QR 分解的累计误差，说明 Householder 算法对计算精度问题具有更好的鲁棒性，进行 QR 分解的稳定性更高。

3.3 问题 3

3.3.1 判断矩阵是否可逆

常见的判断矩阵是否可逆的方法如下所示。

矩阵的秩等于矩阵的阶数，矩阵可逆。

矩阵的行列式不等于零，矩阵可逆。

3) 将矩阵视为线性方程组的系数矩阵，如果该方程组有唯一解，则矩阵可逆。

4) 若存在矩阵 B，使得$AB = BA = I$，则矩阵 A 可逆。

在本次实验中，我使用方法 1 判断矩阵是否可逆，即判断矩阵的秩是否等于矩阵的阶数。

3.3.2 逆矩阵求解

由$A = QR$和Q为正交矩阵我们可以得到：$A^{-1} = (QR)^{-1} = R^{-1}Q^{-1} = R^{-1}Q^{T}$。也就是说，我们可以先对矩阵进行 QR 分解操作，然后对分解后的矩阵 R 求逆，对矩阵 Q 求转置，然后两者进行矩阵相乘的操作即可得到原矩阵的逆。

逆矩阵求解步骤总结如下：

①首先利用矩阵的秩判断矩阵是否可逆，如果不可逆输出提示，然后退出；如果可逆则进行下一步；

②对矩阵进行 QR 分解；
③对 R 求逆，对 Q 求转置；
④将$R^{-1} \times Q^T$作为结果输出。

3.3.3 求解结果验证

使用 matlab 库函数 inv 函数进行矩阵求逆，然后将求得的结果与 3.3.2 步骤中获得的结果进行逐元素比较，如果误差不大则说明求解正确；如果误差太大则说明求解错误。

因为 B 的维数太大，无法完全显示，因此只能截图很小一部分。但是，经过逐元素比较，已经证明 3.3.2 步骤中的求解方法的正确性。

```
     0.06591     0.015891       0.1862      0.14058
    0.030609     0.017671     0.076341    -0.019422
   -0.035553    -0.062064       0.1858    -0.021621
   -0.096529     -0.18406    -0.16355     0.041425
    0.039572    -0.055272    -0.067276     0.042467
     0.12568      0.15249     0.085574     -0.11826
    0.030814      0.16954     -0.21266    -0.055031
    -0.12208   -0.0087951      0.02075     -0.21147

QR求解正确
```

图 7 求解结果验证

四、总结（可含个人心得体会）

1) 在本次实验中，我深入学习并实践了矩阵 QR 分解的两种主要方法：Gram－Schmidt 方法和 Householder 方法。
2) 通过编写代码实现 Gram－Schmidt 方法进行 QR 分解，我了解了如何通过正交化和单位化向量将矩阵分解为一个正交矩阵 Q 和一个上三角矩阵 R。
3) 通过编写代码实现 Householder 方法进行 QR 分解，我了解了如何构造反射算子,利用镜面反射的原理不断对原矩阵进行三角化实现 QR 分解。
4) 在第三部分的矩阵有关逆的问题中，我收获了多种判断矩阵可逆性的方法。同时，我也明白了如何通过 QR 分解求矩阵的逆。
5) Gram－Schmidt 方法由于浮点数的舍入误差，随着矩阵维数的增加，累积误差逐渐增大，导致正交性偏差也随之增加。Modify Gram－Schmidt 方法在一定程度上解决了这个问题，但是还是避免不了在超高维下累计误差太大的问题。
6) Householder 方法显示出了更好的稳定性和对舍入误差的鲁棒性。

指导教师批阅意见： 成绩评定： 指导教师签字：李炎然 2024 年 12 月 7 日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。