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

一、解决问题思路

本次实验实现过程的思维逻辑如图 2 所示，老师审批辛苦了💦！

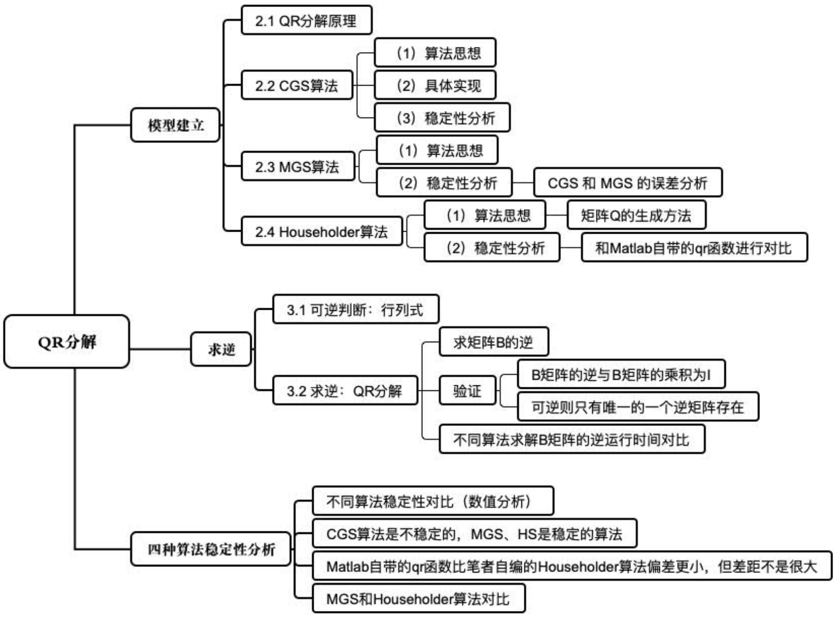


图 2 QR 实验实现过程思维逻辑图

二、模型建立与分析

2.1 QR 分解原理

QR 分解即将矩阵分解成一个正交矩阵与一个上三角矩阵的积，经常被用来求解线性最小二乘法问题。其定义如下：

一个矩阵$A \in \mathbb{R}^{m\times n}$,$m \geq n$且列向量之间线性无关，可以被分解成 $A = QR$, 其中, $Q \in \mathbb{R}^{m\times m}$ 是正交矩阵（$Q^{\top}Q = I$）$R \equiv \begin{bmatrix}\hat{R}\\0\end{bmatrix} \in \mathbb{R}^{m\times n}$，$\hat{R} \in \mathbb{R}^{n\times n}$ 是上三角矩阵

QR 分解的实际计算有很多方法，每一种方法都有其优点和不足。本实验笔者将尝试使用 Gram-Schmidt 正交化（CGS+MGS）和 Householder（自编 HS+Matlab qr）方法进行 QR 分解，并对比分析不同方法的稳定性。

2.2 CGS 算法（Classic Gram–Schmidt Orthogonalization）

（1）算法思想

表 1：Classic Gram-Schmidt 算法流程

$r_{11} = \|a_1\|_2$

$q_1 = a_1/r_{11}$

for $j = 2$ to n **do**

  $q_j = a_j$

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

    $r_{ij} = q_i^T a_j$

    $q_j = q_j - r_{ij}q_i$

```
        end for
        r_jj = ||q_j||_2
        q_j = q_j / r_jj
end for
```

由表 1CGS 算法可知：

$$a_1 = r_{11}q_1,\ a_j = r_{1j}q_1 + r_{2j}q_2 + \cdots + r_{jj}q_j = [q_1, q_2, \dots, q_j]\begin{bmatrix} r_{1j} \\ r_{rj} \\ \vdots \\ r_{jj} \end{bmatrix}, j = 2,3, \dots, n$$

记 $Q = [q_1, q_2, \dots, q_n], R = [r_{ij}]_{n\times n}$，其中 $r_{ij} = \begin{cases} q_i^{\mathrm{T}} a_j, & \text{for } i \le j \\ 0, & \text{for } i > j \end{cases}$

于是 Gram-Schmidt 过程可表示为

$$[a_1, a_2, \dots, a_n] = [q_1, q_2, \dots, q_n]\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & r_{n-1,n} \\ & & & r_{nn} \end{bmatrix}, \text{即 } A = QR$$

（2）具体实现

实验模板（图 1）已经给出了 CGS 算法的完整代码，先对循环内的四行代码功能进行解释，如表 2 所示。但是该代码省略了当 j=1 时的情况，之所以能够缩减，是因为矩阵(1:0,:)是一个空矩阵。

表 2:CGS 算法代码解析

1：生成了 $r_{ij} = q_i^T a_j\ (r_{1j} = q_1^T a_j, r_{2j} = q_2^T a_j, \dots, r_{j-1,j} = q_{j-1}^T a_j)$
2：生成了$q_j = a_j - r_{ij}q_i$
3：q_j的 2 范数作为了 R 对角线元素r_{jj}
4：对q_j进行单位化

（3）稳定性分析（题目 1:验证 GS 是否能稳定进行 QR 分解矩阵 A，Q 矩阵是否正交）

通过该算法求得矩阵 Q，为了验证 Q 矩阵的正交性，可以计算q_j与前面列正交性的偏差（偏差 = $\max_{1\le i<j}|q_i^T q_j|, j = 2, \dots, n$），结果如图 3 所示。实验结果说明了 CGS 方法并不能稳定地进行 QR 分解，当$j > 36$时，q_j与前面列不再具有严格的正交关系，因此 Q 矩阵的正交性也就不存在了。

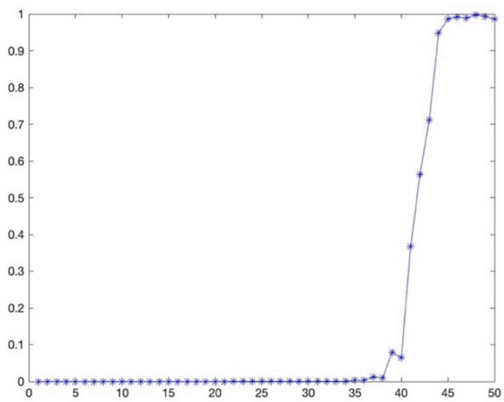


图 3 CGS:Q 矩阵的正交性偏差

值得注意的是，观察 CGS 算法过程,可以发现,唯一可能在理论上出问题的情况就是,出现某个$r_{jj}=0$,导致在算法（表 X）第 10 行（$q_j=q_j/r_{jj}$）出现分母为 0 的情况。因此只要$A\in\mathbb{R}^{m\times n}$是满秩的,且每个$r_{jj}>0$,那么 QR 分解的结果就是唯一的。

经典 GS 算法给笔者的第一印象是十分完美，并为算法的创造者钦佩之至。但是计算机总是个不完美的发明，体现不了数学理论的和谐，由于计算机浮点数存储的舍入误差，经典的 GS 算法对舍入误差很敏感,容易导致生成的基q_j的正交性随着迭代越来越弱,因此笔者在本实验中引入 MGS 算法。

2.3 MGS 算法（Modified Gram–Schmidt Orthogonalizatio）

（1）算法思想

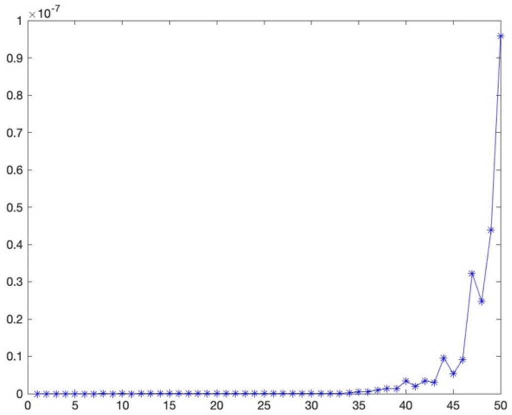
改进的 GS 算法（MGS）核心思想是：在每个q_j生成后,直接把 A 剩下的列(表 3 算法第 10 行)都去掉q_j的成分(表 3 算法的第 11、12 行)。由于只是改变了计算的顺序,所以理论上计算结果是一样的。

表 3：Modified Gram-Schmidt 算法实现 QR 分解

1: Set $R=[r_{ij}]=0_{n\times n}$ (the $n\times n$ zero matrix)
2: **if** $a_1=0$ **then**
3:     $q_1=0$
4: **else**
5:     $r_{11}=\|a_1\|_2$
6:     $q_1=a_1/\|a_1\|_2$
7: **end if**
8: **for** j= 2 to n **do**
9:     $q_j=a_j$
10:     **for** $i=1$ to j−1 **do**
11:         $r_{ij}=q_i^{\top}q_j$ % MGS 和 CGS 的区别
12:         $q_j=q_j-r_{ij}q_i$
13:     **end for**
14:     **if** $q_j\neq 0$ **then**
15:         $r_{jj}=\|q_j\|_2$
16:         $q_j=q_j/r_{jj}$
17:     **end if**
18: **end for**

（2）稳定性分析

通过 MGS 算法求得矩阵 Q，同样绘制q_j与前面列之间的正交性的偏差（如图 4 所示），可以看到 Q 矩阵正交性偏差极小，偏差的数量级为10^{-7}，可以忽略不计，可以认为 MGS 对矩阵 A 的分解稳定。笔者通过将 CGS 和 MGS 算法得到的 Q 矩阵正交性偏差放到一起，从图 5 可以直观地看到这两种算法的稳定性差距。



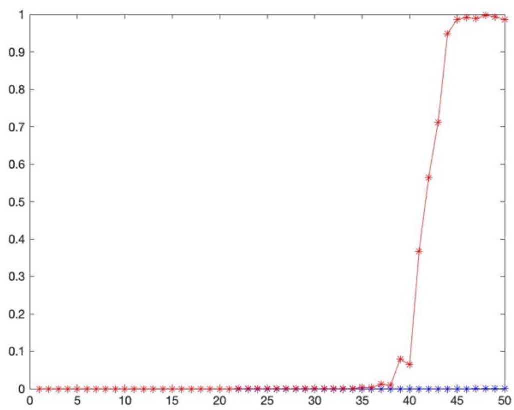


图 4 MGS:Q 矩阵的正交性偏差　　　　　图 5 CGS（红）和 MGS（蓝）偏差对比

（1）可以看到，改进之后稳定性会大幅提升，从实际计算步骤上来看：CGS 算法和 MGS 算法区别在于：CGS 算法中，每次迭代新的一列q_i,计算每个$r_{ij}=q_i^Tq_j$都是用的同一个q_j,而 MGS 算法计算r_{ij}的时候用的q_j是已经减去前面$\mathrm{j}-1$个基的分量之后的q_j。

（2）这样做的好处是：误差的传递是局部的。比如计算q_1是精确的,计算q_2出现误差,即q_2在q_1上存在一个微小分量，按照 CGS 算法,接下来要分别计算q_3在q_1和q_2的分量,最终$q_1^Tq_3\neq 0, q_2^Tq_3\neq 0$;而 MGS 算法则先计算$q_3$在$q_1$上的分量,去除掉这个分量之后成为$q_3'$,再计算并去除$q_3'$在$q_2$上的分量得到最终的$q_3''$,此时如果计算是精确的,那么至少可以保证$q_3''\perp q_2$。

（3）从另一个角度分析，根据(Rice J R ,1966)[1]中对 CGS 和 MGS 的误差分析。

由于计算机舍入误差的存在，$(q_i,q_j)=\epsilon_{ij}, i,j\leq k-1$；

$a_k=\sum_{i=1}^{k-1}\mathrm{R_{ik}}q_i+\eta$, η 是与标准正交基 q 正交的一个向量，$(\eta,q_i)=0$；

$q_k'=\eta-\sum_{j=1}^{k-1}\left(\sum_{i\neq j}R_{ik}\epsilon_{ij}\right)q_j$；

$q_k=\frac{q_k'}{||q_k'||}$，如果 η 较小，则q_k主要由$q_i, i=1,\dots,k-1$的线性组合构成，则 $\epsilon_{k,k-1}$更加接近 1；而对于 MGS，在每计算出一列 q后，将A中剩余的各列向量,分别减去其在q上的投影：

$$a_i^{(k)}=a_i^{(k-1)}-q_k\left(q_k^Ta_i^{(k-1)}\right), i=k+1,\dots,n$$

$\left(a_i^{(k)},q_k\right)=\left(a_i^{(k-1)},qk\right)-\left(a_i^{(k-1)},q_k\right)(q_k,q_k)=0$，即 $\epsilon_{k,k-1}=0$，而在 GS 中，这一项逐渐更可能接近 1。

2.4 Householder 算法

（1）算法思想

定理:设$x=[x_1,x_2,\dots,x_n]^\top\in\mathbb{R}^n$是一个非零向量,则存在 Householder 矩阵$H(v)$使得$H(v)x=\alpha e_1$,其中$\alpha=\|x\|_2$(或$\alpha=-\|x\|_2$),$e_1=[1,0,\dots,0]^\top\in\mathbb{R}^n$。

通过上述定理可知，通过 Householder 变换可以将任何一个非零变量$x\in\mathbb{R}^n$转化成$\|x\|_2 e_1$,即除第一个元素外,其它都为零。通过 Householder 变换来实现矩阵的QR分解的算法

[1] Rice J R . Experiments on Gram-Schmidt orthogonalization[J]. Mathematics of Computation, 1966, 20(94):325-328.

流程如表 4 所示：

表 4：基于 Householder 变换的 QR 分解

Set $Q = I_{m\times m}$

for $k = 1$ to n **do**

$\quad x = A(k{:}m, k)$

$\quad [\beta, v_k] =$ **House** $(\boldsymbol{x})$ $\quad \%\tilde{H}_k = I - \beta v_k v_k^T$

$\quad A(k{:}m, k{:}n) = (I_{m-k+1} - \beta v_k v_k^{\mathsf{T}})A(k{:}m, k{:}n)$

$\quad\quad\quad = A(k{:}m, k{:}n) - \beta v_k (v_k^{\mathsf{T}} A(k{:}m, k{:}n))$

$\quad Q(:, k{:}m) = Q(:, k{:}m)(I_{m-k+1} - \beta v_k v_k^{\mathsf{T}})$

$\quad\quad\quad = Q(:, k{:}m) - \beta(Q(:, k{:}m)v_k)v_k^{\mathsf{T}}$

end for

上述表 4 的算法只是关于利用 Householder 变换来实现 QR 分解的一个简单描述,并没有考虑运算量问题。在实际计算时,笔者保留了所有的 Householder 向量。由于$\tilde{H}_k$所对应的 Householder 向量v_k的长度为$m-k+1$,因此先把v_k单位化,使得v_k的第一元素为 1,这样就只要存储$v_k(2{:}end)$,共$m-k$个元素。这样就可以把所有的 Householder 向量存放在A的严格下三角部分，而A的上三角部分仍然存放R，若在计算Q时采用向后累积的方法,总的运算量大约为$4m^2n - 2mn^2 + \frac{2}{3}n^3$.若$m = n$,则运算量大约为$\frac{8}{3}n^3$。

这里笔者对此算法矩阵 Q 的生成方法进行了进一步的探讨：

(1) 如果不需要生成Q,则基于 Householder 变换的 QR 分解的总运算量大约为$2mn^2 - 2n^3/3$.

(2) 如果保留了每一步的 Householder 向量,则Q也可以通过下面的向后累积方法实现:

$$\begin{cases} Q = I_n \\ Q = QH_k,\ k = 1,2,\dots,n-1 \end{cases}$$

这样做的好处是一开始Q会比较稀疏,随着迭代的进行,Q才会慢慢变满。采用这种方法计算Q的运算量大约为$4m^2n - 4mn^2 + 4n^3/3$

(3) 如果将Q写成下面的形式，

$$Q = I + WY^{\mathsf{T}}$$

则可以采用分块形式来计算W和Y,虽然运算量会稍有增长,但大多数运算是矩阵乘法,因此可以尽可能多地采用 3 级 BLAS 运算,效率可能会更高(G. H. Golub,2013)[2]

（2）稳定性分析（题目 2:实现 Householder 代码，并验证其对矩阵 A 分解是否稳定）

通过 Householder 算法求得矩阵 Q，同样绘制q_j与前面列之间的正交性的偏差（如图 6 所示），可以看到正交性偏差极小，偏差的数量级为10^{-15}，可以忽略不计，验证了 HS 算法对矩阵 A 的分解稳定。

[2] G. H. Golub and C. F. Van Loan, Matrix Computations, The 4th Editon, The Johns Hopkins University Press, Baltimore, MD, 2013. 20, 37, 83, 87, 89, 105, 121, 152, 153

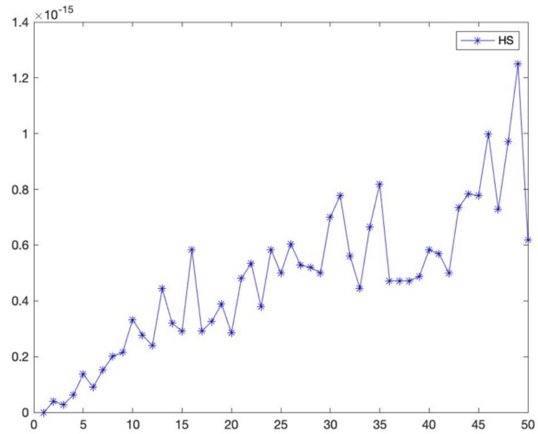


图 6 Householder:Q 矩阵的正交性偏差

笔者进一步将自编的 Householder 算法和 Matlab 自带的 qr 函数进行对比，使用两种方法对矩阵 A 和矩阵 B 进行 QR 分解，同样绘制q_j与前面列之间的正交性的偏差（如图 7 所示），可以看到 Matlab 自带的 qr 函数比笔者自编的 Householder 算法偏差更小，但差距不是很大。通过对查找资料，笔者发现这很大可能是因为 Matlab 的 qr 函数通过对矩阵 Q 的生成方法进行了优化（上一小节已对此进行了解释），提升了性能，这也是笔者将来可以改进的方向。

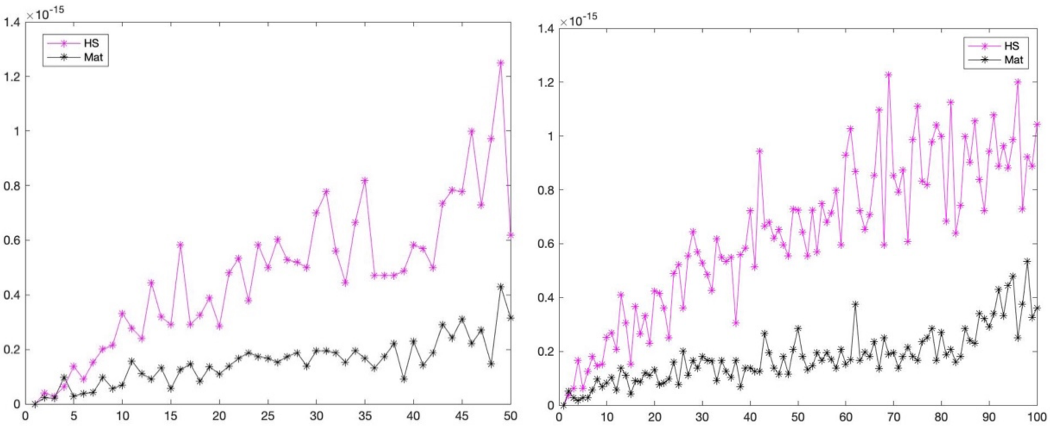


图 7 自编 HS 和 qr 函数正交性偏差对比（紫:自编,黑:Matlab）（左:A 矩阵,右:B 矩阵）

三、求逆（题目：读取方矩阵 B，判断其是否可逆？如果可逆，求其逆矩阵）

3.1 可逆判断：行列式

对于可逆判断，在学习 QR 分解之前，可以通过求矩阵 B 的行列式进行判断，若为 0，则不可逆；若不为 0, 则为可逆。通过 Matlab 的 det 函数求得矩阵的 B 的行列式为 0.9048，非 0，说明 B 的列向量线性无关，可逆，非奇异。

3.2 求逆：QR 分解

矩阵 B 非奇异，则 B 矩阵就可进行 QR 分解，其逆可以表示为$B^{-1}=(QR)^{-1}=R^{-1}Q^T$，利用右式，笔者这里使用 Householder 算法来求 B 矩阵的逆。

设$B=[b_1,b_2,\dots,b_n]$,则$Rb_1=Q^Te_1,Rb_2=Q^Te_2,\dots,Rb_n=Q^Te_n$。其中，R、Q 已通过 Householder 算法求得，又因为R为上三角矩阵,所以可以用回代法循环求解出所有的b_n,合并起来就是矩阵 B 的逆。

通过 Householder 算法对矩阵 B 进行 QR 分解的偏差（10^{-15}）如图 8 所示，得到矩阵 B 的逆如图 9 所示（截取部分，完整矩阵请见附件 invB.xlsx）

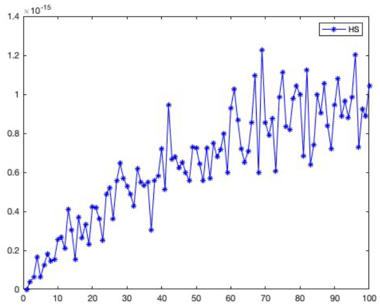
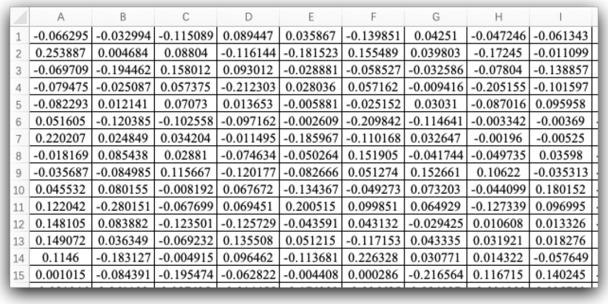


图 8 HS：Q 矩阵的正交性偏差（B 矩阵）



	A	B	C	D	E	F	G	H	I
1	-0.066295	-0.032994	-0.115089	0.089447	0.035867	-0.139851	0.04251	-0.047246	-0.061343
2	0.253887	0.004684	0.08804	-0.116144	-0.181523	0.155489	0.039803	-0.17245	-0.011099
3	-0.069709	-0.194462	0.158012	0.093012	-0.028881	-0.058527	-0.032586	-0.07804	-0.138857
4	-0.079475	-0.025087	0.057375	-0.212303	0.028036	0.057162	-0.009416	-0.205155	-0.101597
5	-0.082293	0.012141	0.07073	0.013653	-0.005881	-0.025152	0.03031	-0.087016	0.095958
6	0.051605	-0.120385	-0.102558	-0.097162	-0.002609	-0.209842	-0.114641	-0.003342	-0.00369
7	0.220207	0.024849	0.034204	-0.011495	-0.185967	-0.110168	0.032647	-0.00196	-0.00525
8	-0.018169	0.085438	0.02881	-0.074634	-0.050264	0.151905	-0.041744	-0.049735	0.03598
9	-0.035687	-0.084985	0.115667	-0.120177	-0.082666	0.051274	0.152661	0.10622	-0.035313
10	0.045532	0.080155	-0.008192	0.067672	-0.134367	-0.049273	0.073203	-0.044099	0.180152
11	0.122042	-0.280151	-0.067699	0.069451	0.200515	0.099851	0.064929	-0.127339	0.096995
12	0.148105	0.083882	-0.123501	-0.125729	-0.043591	0.043132	-0.029425	0.010608	0.013326
13	0.149072	0.036349	-0.069232	0.135508	0.051215	-0.117153	0.043335	0.031921	0.018276
14	0.1146	-0.183127	-0.004915	0.096462	-0.113681	0.226328	0.030771	0.014322	-0.057649
15	0.001015	-0.084391	-0.195474	-0.062822	-0.004408	0.000286	-0.216564	0.116715	0.140245

图 9 矩阵 B 的逆（部分）

接着，笔者进一步进行了验证，通过基于 Householder 算法的计算，所求得的 B 矩阵的逆与 B 矩阵的乘积为 I，说明算法稳定性较好。

从理论上来说，B 矩阵若可逆则只有唯一的一个逆矩阵存在，因此笔者尝试将该算法求的 B 的逆矩阵与 Matlab 算法算得的 B 的逆矩阵（命令为：InvB = inv(B'*B)*B'）做差,通过命令 det(InvA-X)求的 ans=0，再一次进行了验证。

为了再次对比 4 种不同算法 QR 分解稳定性的差异，笔者使用不同的算法来求解 B 矩阵的逆并计算运行时间(由于单次运行时间过短，笔者对每一种方法运行了 10 次并取均值)，得到的结果如表 5 所示。可以看到 Matlab 直接求解运行速度最快，可能是因为笔者在求逆的时候为了照顾完备性，使用了 backsub 回代法函数导致了性能稍微降低有关，但整体上，不同算法的运行时间差距不是很大。

表 5：不同算法求解 B 矩阵的逆运行时间对比

算法	CGS	MGS	HS	Matlab
运行时间	0.0402	0.0394	0.0361	0.0211

四、四种算法稳定性分析

笔者最后再一次对四种 QR 分解方法求解的正交性偏差进行总结对比，使用 A 矩阵作为实验对象，不同算法的稳定性如下表 6 所示。表 6 可帮助我们从数值角度进行分析，接下来笔者结合可视化图表进行一些总结。

表 6:不同算法稳定性对比（$\max_{1\le i<j}\left|q_i^T q_j\right|, j = 2, \ldots, n$）

n	CGS	MGS	HS	Matlab_qr
2	4.77E-18	8.24E-17	3.90E-17	3.90E-17
10	2.18E-13	5.98E-15	3.33E-16	3.33E-16
15	1.50E-11	2.27E-14	2.91E-16	2.91E-16
20	3.57E-09	2.93E-13	2.84E-16	2.84E-16
25	1.27E-07	7.19E-12	5.00E-16	5.00E-16
30	6.99E-05	2.85E-11	7.01E-16	7.01E-16
35	0.00375005	4.55E-10	8.19E-16	8.19E-16
40	0.06427329	3.41E-09	5.83E-16	5.83E-16

45	0.9866304	5.34E-09	7.77E-16	7.77E-16
50	0.98644042	9.58E-08	6.18E-16	6.18E-16

（1）将 4 种算法的偏差进行对比（图 10），说明 CGS 算法是不稳定的，MGS、HS 是稳定的算法

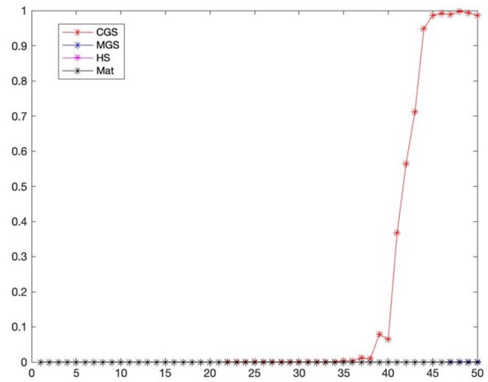


图 10 4 种算法的偏差对比

（2）Matlab 自带的 qr 函数比笔者自编的 Householder 算法偏差更小，但差距不是很大。（上述图 7 已进行分析）

（3）基于 Householder 的 QR 分解都具有很好的数值稳定性,基于 MGS 的 QR 分解也是向后稳定的（C. C. Paige,2006）[3]那么两种算法哪一种更优呢？笔者将进行探讨：

首先分别使用 MGS 和 HS 算法对 A 矩阵和 B 矩阵进行 QR 分解，得到的结果如图 11 所示。实验结果表明，对 n=50 的矩阵 A 进行 QR 分解，HS 算法表现更优；对 n=100 的矩阵 B 进行 QR 分解，MGS 的表现更优。

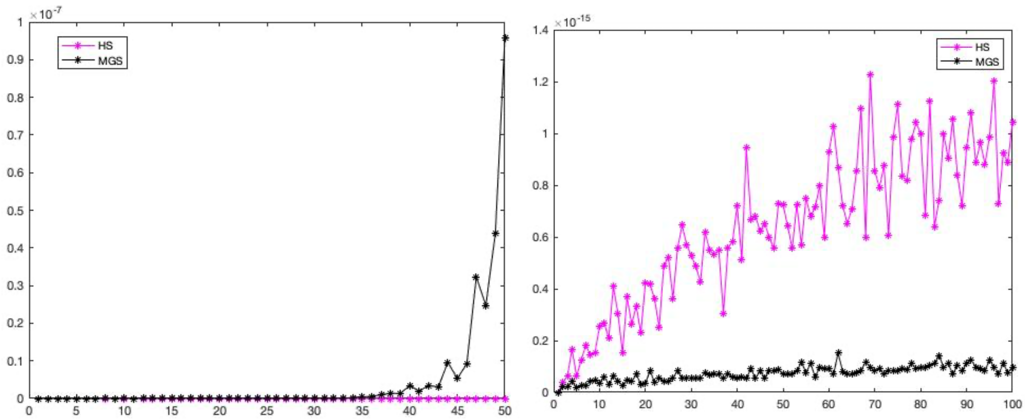


图 11 MGS 和 Householder 算法对比（左：矩阵 A；右：矩阵 B）

通过查阅资料，发现当需要计算矩阵Q时,基于 MGS 的QR分解的运算量相对较少，因此当A的列向量具有很好的线性无关性时,可以使用 MGS 来计算QR分解。但需要注意的是,MGS 得到的Q不是方阵,除非A是方阵。

由于舍入误差的原因,最后得到的矩阵Q会带有一定的误差,可能会导致Q失去正交性。

[3] C. C. Paige, M. Rozloˇz ník and Z. Strakoˇs, Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS-GMRES, SIAM Journal on Matrix Analysis and Applications, (28) 2006, 264-284. 85

(Björck,1967)[4]证明了,通过 MGS 计算的矩阵Q满足

$$Q^{\top}Q = I + E_{MGS} \text{ 其中} \|E_{MGS}\|_2 \approx \varepsilon_u \kappa_2(A).$$

而 Householder 变换计算的矩阵Q满足

$$Q^{\top}Q = I + E_H \text{ 其中 } \|E_H\|_2 \approx \varepsilon_u.$$

因此,如果正交性至关重要,则当A的列向量接近线性相关时,最好使用 Householder 变换。

四、小结（可含个人心得体会）

4.1 实验总结

笔者在上述已经对本次实验进行了很详细的总结，这里主要再从理论角度对 Gram-Schmidt 算法和 Householder 算法进行总结对比，如下表 7 所示：

表 7 Gram-Schmidt 算法和 Householder 算法对比

	算法复杂度	优点	缺点
GS	$2mn^2$	（1）适合小矩阵计算； （2）每次迭代都生成一个正交基，可以随时停止计算（HS 只能在迭代结束之后才可以计算全部正交基，中间产生了大量的半成品。）	（1） 不适合稀疏矩阵； （2）在有限精度的病态矩阵中会导致大量误差； （3）循环过程中必须保存整个矩阵，导致内存开销大。
HS	$4m^2n - 2mn^2 + \frac{2}{3}n^3$	（1）不显式计算 Q 矩阵，只保存若干个反射矩阵 H，适合某些仅需要 R 的任务； （2） 最适合稠密矩阵。	（1）迭代的过程不产生可用的正交基，必须等到全部迭代完成，无法中断； （2）对于稀疏矩阵会产生大量无效计算。

如表 7 所示，在算法复杂度方面，Householder 算法更占优势，但是实际工程上，需要根据任务要求和矩阵稠密度灵活分析。例如需要使用 Q 矩阵的标准正交基，则 GS 算法更适合。在算法稳定性方面，两种算法都不适合应用于稀疏矩阵。由于计算机用浮点数格式存储数，当其绝对值很小时，其值会被认为 0。所以如果一个矩阵中的数值过小，或者说该矩阵的行列式的值过小，则会使得该两种方法的应用都失去稳定性。

4.2 个人体会

这次实验考察的内容很多，笔者对 QR 分解有了更深的认识，对于算法的性能和稳定性也有了重新的认识。此外，Matlab 编程水平也进一步提高。

整个实验过程花费了很多时间，也查阅了很多资料和文献，这种钻研理解、实验验证的过程真的需要很多耐心和时间，但是当我敲下结尾这段快结束了的字，这种成就感和探索过后有收获真的很好，也许这就是所谓学习的快乐，只希望恨不得有更多的时间…

[4] Åke Björck, Solving linear least square problems by Gram-Schmidt orthogonalization, BIT, 7 (1967), 1–21. 85