

Fall 2016 CSE/CS 383 HW#3 Po-Cheng Pan (UT EID: pp22828)

1. Let $A \in \mathbb{R}^{m \times n}$ be full rank and let $\{g_j\}_{j=1}^n$ be the orthonormal vectors from classical Gram-Schmidt orthogonalization of the columns of A

Let $P_j = g_j g_j^*$

$$w = \left[\prod_{j=1}^{i-1} (I - P_j) \right] A(:, i), \quad g_i = \frac{w}{\|w\|_2}, \quad i=1, \dots, m$$

From the textbook, we know the classical Gram-Schmidt as below:

$$g_1 = \frac{A(:,1)}{\|A(:,1)\|_2} \quad g_2 = \frac{A(:,2) - g_1^* A(:,1) g_1}{\|A(:,2) - g_1^* A(:,1) g_1\|_2} \quad \dots \quad g_n = \frac{A(:,n) - \sum_{i=1}^{n-1} g_i^* A(:,i) g_i}{\|A(:,n) - \sum_{i=1}^{n-1} g_i^* A(:,i) g_i\|_2}$$

From (8.1) & (8.3)

g_i can be also expressed as $g_1 = \frac{A(:,1)}{\|A(:,1)\|_2} \quad g_2 = \frac{P'_2 A(:,2)}{\|P'_2 A(:,2)\|_2} \quad \dots \quad g_n = \frac{P'_n A(:,n)}{\|P'_n A(:,n)\|_2}$ where $P'_j = I - Q_{j-1} Q_{j-1}^*$

$$Q_{j-1} = [g_1 | g_2 | \dots | g_{j-1}]$$

Then consider modified Gram-Schmidt algorithm, $P_j = g_j g_j^*$

$$i=1: \prod_{j=1}^0 (I - P_j) = I \Rightarrow \frac{w_1}{\|w_1\|_2} = \frac{A(:,1)}{\|A(:,1)\|_2}$$

$$i=2: \prod_{j=1}^1 (I - P_j) = I - P_1 = I - g_1 g_1^* = I - Q_1 Q_1^* = P'_2 \Rightarrow \frac{w_2}{\|w_2\|_2} = \frac{P'_2 A(:,2)}{\|P'_2 A(:,2)\|_2} = g_2$$

We suppose when $i=k$, $\prod_{j=1}^{k-1} (I - P_j) = P'_k \Rightarrow \frac{w_k}{\|w_k\|_2} = g_k$

Consider $i=k+1$

$$\begin{aligned} \prod_{j=1}^k (I - P_j) &= \prod_{j=1}^{k-1} (I - P_j) \cdot (I - P_k) \\ &= P'_k \cdot (I - g_k g_k^*) \\ &= (I - Q_k Q_k^*) (I - g_k g_k^*) \\ &= I - Q_k Q_k^* - g_k g_k^* + Q_k Q_k^* g_k g_k^* \quad (\because g_1, g_2, \dots, g_k \text{ all are orthonormal} \\ &\quad \therefore Q_k Q_k^* g_k g_k^* = 0) \\ &= I - Q_k Q_k^* - g_k g_k^* \\ &= I - (Q_k Q_k^* + g_k g_k^*) \quad (\because Q_k Q_k^* \& g_k g_k^* \text{ both are } \mathbb{R}^{n \times n} \\ &\quad \therefore \text{We can combine } g_k \text{ into } Q_{k+1} \text{ as} \\ &\quad Q_{k+1} = [g_1 | g_2 | \dots | g_k] \text{ s.t. } Q_{k+1} Q_{k+1}^* = Q_k Q_k^* + g_k g_k^*) \\ &= I - Q_{k+1} Q_{k+1}^* \\ &= P'_{k+1} \Rightarrow \frac{w_{k+1}}{\|w_{k+1}\|_2} = g_{k+1} \end{aligned}$$

By induction, we prove $g_i = \frac{w_i}{\|w_i\|_2}, i=1, \dots, m$. Hence, both classical and modified algorithm are mathematically equivalent.

Homework 3

Fall 2016, CSE/CS 383, Linear Algebra

Coding Parts:

2. Write a single Matlab function `[Q,R] = gramschmidt(A,flag)` that computes a reduced QR factorization $A = QR$ of an $m \times n$ full rank matrix A with $m \geq n$ using the classical Gram Schmidt orthogonalization when `flag == true` and the modified Gram Schmidt orthogonalization when `flag == false` or `flag` is left unspecified. Q should be an $m \times n$ matrix and R an $n \times n$ matrix.

Ans:

Code:

```
1 %%gramschmidt
2 function [Q, R] = gramschmidt(A, flag)
3 if ~exist('flag', 'var')
4     flag = false;
5 end
6 [m, n] = size(A);
7 Q = zeros(m, n);
8 R = zeros(n, n);
9 % Classical Gram-Schmidt
10 if (flag == true)
11     for j = 1:n
12         v = A(:, j);
13         for i = 1:(j-1)
14             R(i,j) = Q(:,i)'*A(:,j);
15             v = v - R(i, j)*Q(:,i);
16         end
17         R(j,j) = norm(v, 2);
18         Q(:, j) = v/R(j,j);
19     end
20 % Modified Gram-Schmidt
21 else
22     for i = 1:n
23         R(i, i) = norm(A(:,i),2);
24         Q(:, i) = A(:,i)/R(i,i);
25         for j = i+1:n
26             R(i,j) = Q(:,i)'*A(:,j);
27             A(:, j) = A(:, j) - R(i, j)*Q(:,i);
28         end
29     end
30 end
31
```

3. Suggest a test (or tests) to (quantitatively) check the accuracy of your QR factorization. Apply the test(s) to QR factorizations obtained by the two Gram Schmidt variants and Matlab's $[Q,R]=qr(A)$ to the following matrices:
 $A = \text{gallery}('randsvd',100,kappa);$ for $kappa=1,1E3,1E6, 1E9$. Discuss your results.

Ans:

I consider three different directions to check the accuracy of my QR factorization :
 Accuracy, Orthogonality and whether R is upper triangular or not.

For these three directions, I have following definitions to evaluate them.

Accuracy : $\text{error} = \|QR - A\|_2 / \|A\|_2$

Orthogonality: $\|Q^*Q - I\|_2 / \|I\|_2$

R is upper triangular or not : Using the Matlab function : `istriu`

In these tests, test matrices are $A = \text{gallery}('randsvd',100,kappa);$ for $kappa=1,1E3,1E6, 1E9$. ($kappa$ is the value of the condition number.) Besides, to be general, I ran each test for 100 times and then calculated their averages. The test results are as following three tables.

Accuracy:

	Matlab	Classical	Modified
1	1.8851E-15	5.9620E-16	6.1756E-16
1E+03	8.0518E-16	2.2419E-16	2.2264E-16
1E+06	7.1646E-16	1.7984E-16	1.7920E-16
1E+09	7.0958E-16	1.6370E-16	1.6333E-16

Orthogonality:

	Matlab	Classical	Modified
1	2.4466E-15	1.0949E-15	1.0256E-15
1E+03	2.4262E-15	7.8979E-12	6.2346E-14
1E+06	2.3803E-15	6.2346E-05	3.4670E-11
1E+09	2.4021E-15	1.1393E+01	2.6848E-08

Upper Triangular:

	Matlab	Classical	Modified
1	The R matrices of all tests are still upper triangular matrices		
1E+03			
1E+06			
1E+09			

Discussion:

From the test results, we can find that

1. As the condition number increases, the accuracy of classical and modified QR method is better than the Matlab intrinsic QR function. And the errors of modified QR method are a little lesser than the errors of simplified QR method when the condition number is large
2. However, as the condition number increases, the orthogonality of Matlab intrinsic function is much better than other two methods. In addition, the orthogonality of modified method is better than the classical method.
3. The R matrices created by all methods all satisfy the conditions of upper triangular matrices.

Test Code: (P.S. All my codes are also uploaded to Canvas)

```

1  %% Main for Question 2
2  clc; clear; close all;
3
4  repeattime = 100;
5  kappa = [1, 1e3, 1e6, 1e9];
6  error = zeros(3, size(kappa, 2));
7  orthogonality = zeros(3, size(kappa, 2));
8  isUpperTriangular = zeros(3, size(kappa, 2));
9
10 for j = 1 : repeattime
11     for i = 1 : size(kappa, 2)
12
13         A = gallery('randsvd', 100, kappa(i));
14         [Q_matlab, R_matlab] = qr(A);
15
16         error(1, i) = error(1, i) + norm((A - Q_matlab * R_matlab)) / norm(A);
17         orthogonality(1, i) = orthogonality(1, i) + norm(Q_matlab * Q_matlab' - eye(size(Q_matlab))) / norm(eye(size(Q_matlab)));
18         isUpperTriangular(1, i) = isUpperTriangular(1, i) + istriu(R_matlab);
19
20         [Q_classical, R_classical] = gramschmidt(A, true);
21
22         error(2, i) = error(2, i) + norm((A - Q_classical * R_classical)) / norm(A);
23         orthogonality(2, i) = orthogonality(2, i) + norm(Q_classical * Q_classical' - eye(size(Q_classical))) / norm(eye(size(Q_classical)));
24         isUpperTriangular(2, i) = isUpperTriangular(2, i) + istriu(R_classical);
25
26         [Q_modified, R_modified] = gramschmidt(A);
27
28         error(3, i) = error(3, i) + norm((A - Q_modified * R_modified)) / norm(A);
29         orthogonality(3, i) = orthogonality(3, i) + norm(Q_modified * Q_modified' - eye(size(Q_modified))) / norm(eye(size(Q_modified)));
30         isUpperTriangular(3, i) = isUpperTriangular(3, i) + istriu(R_modified);
31
32     end
33 end
34
35 error = error ./ repeattime;
36 orthogonality = orthogonality ./ repeattime;
37 isUpperTriangular = isUpperTriangular ./ repeattime;

```