Homework 3

Due: Monday, November 2, 2020

Question 1. QR Decomposition
Develop a numerical algorithm that implements the modified Gram-Schmidt orthogonalization procedure. Compare your algorithm to (i) the qrfactor.m code that we built in class (you can download it from the third lecture on QR on the website), and (ii) MATLAB's QR algorithm on a variety of matrices to see how well your algorithm works. Be sure to try it on a matrix that is ill-conditioned, i.e. cond (A) $\gg 1$.

We use two metrics to compare these algorithms, i.e. the error of restoring the original matrix (QR error) and the error of restoring the identity matrix (Orthogonality error). We define the errors as follows (implemented in `test.m`)

$$\text{QR error} = \frac{\|\mathbf{QR} - \mathbf{A}\|}{\|\mathbf{A}\|}$$

$$\text{Orthogonality error} = \|\mathbf{Q}^*\mathbf{Q} - \mathbf{I}\|$$

For a well-conditioned matrix, all algorithms perform similarly.

```
>> test(rand(5,3))
Condition number: 6.4174e+00

QR error
Matlab QR: 1.4875e-16
Modified GS: 6.9318e-17
Householder: 2.1091e-16

Orthogonality error
Matlab QR: 4.4513e-16
Modified GS: 4.1435e-16
Householder: 2.4243e-16
```

For an ill-conditioned matrix, the modified Gram-Schmidt fails to restore orthogonality.

```
>> test(hilb(20))
Condition number: 2.1065e+18

QR error
Matlab QR: 2.3340e-16
Modified GS: 4.3193e-17
Householder: 5.3150e-16

Orthogonality error
Matlab QR: 1.1697e-15
Modified GS: 9.9896e-01
Householder: 1.3425e-15
```

Question 2. Consider the polynomial

$$p(x) = (x-2)^9 = x^9 - 18x^8 + 144x^7 - 672x^6 + 2016x^5 - 4032x^4 + 5376x^3 - 4608x^2 + 2304x - 512$$

(a) Plot the polynomial $p(x)$ for $x \in [1.920, 2.080]$ for step-sizes of $\delta x = 0.001$ using the right-hand side of the expression above. (b) Plot the polynomial again over the same interval using the left-hand side of the expression, i.e. $(x-2)^9$.

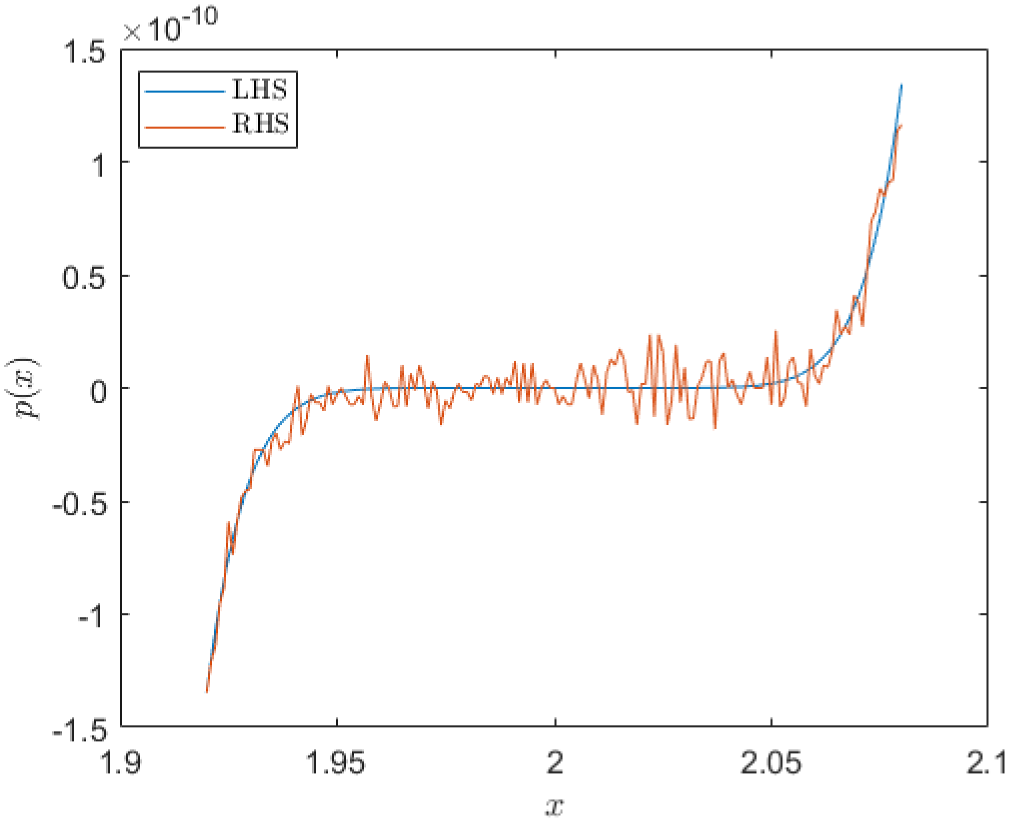

Figure 1: Plot of $p(x)$ for $x \in [1.920, 2.080]$ and $\delta x = 0.001$. RHS polynomial fluctuates due to numerical round-off of each term in the polynomial

Question 3. Consider the conditioning of a matrix.

(a) Construct a random matrix of size $m \times n$ where $m > n$, i.e. use $\mathbf{A} = \text{randn}(m, n)$. Study the condition number as a function of the size of the matrix (increase the m and n).

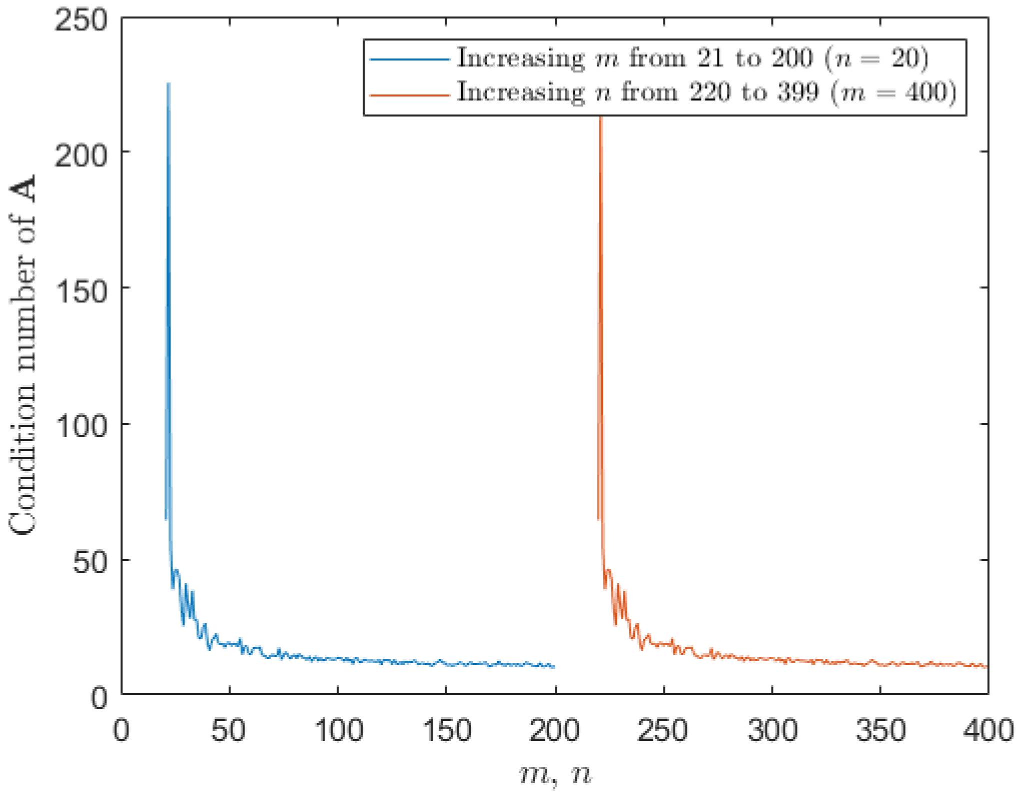

Figure 2: Condition number as a function of m and n. It is evident that increasing m and n reduces the condition number

(b) For a fixed m and n, copy the first column of $\mathbf{A}$ and append it as the $(n+1)$ th column of $\mathbf{A}$. What is the condition number and determinant of the matrix?

Appending the first column to the last produces a singular matrix, thus the condition number will be very large ($O(10^{16})$) and determinant will be close to 0. In theory, the condition number is infinity and determinant is exactly 0.

(c) Take the appended $(n+1)$ th column and add noise to it, i.e. $\mathbf{a}_{n+1} = \mathbf{a}_{n+1} + \epsilon\,\text{rand}(m, 1)$ and see what happens to the condition number as a function of ϵ.

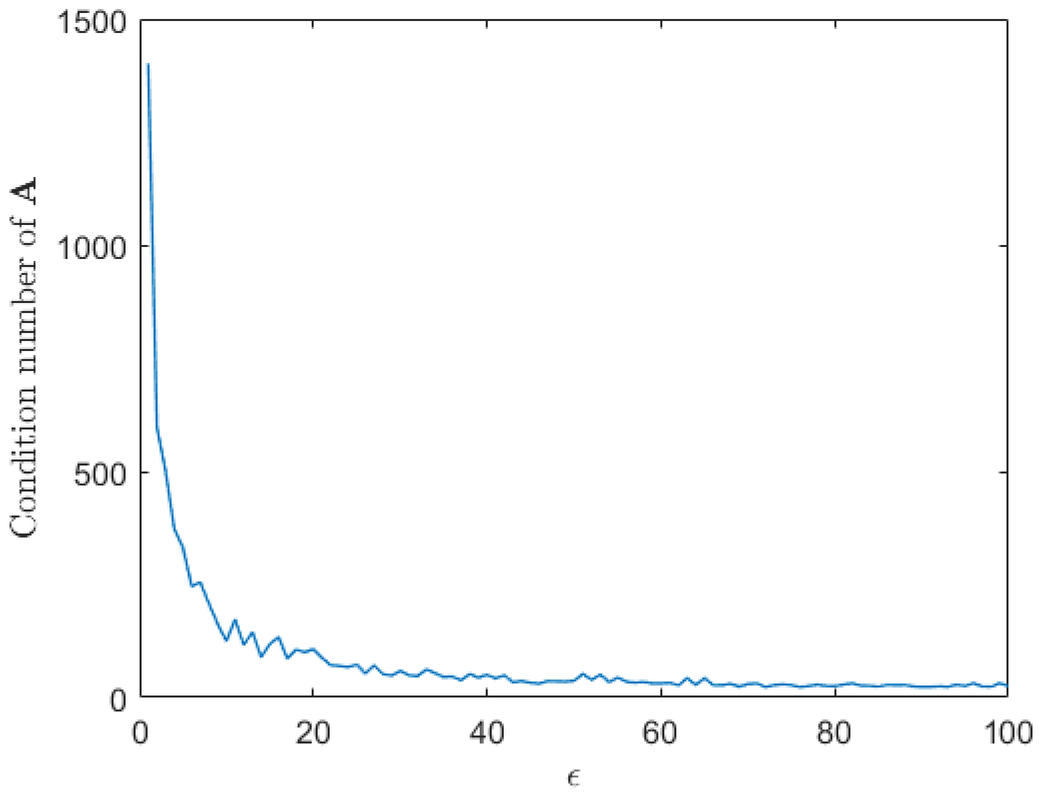

Figure 3: Condition number of a 40×20 matrix as a function of ϵ. When we add noise to a singular matrix, it becomes less singular, thus its conditional number decreases. When the noise gets too large, the matrix is randomized and its condition number fluctuates

Code

qrMGS.m

```
function [Q,R] = qrMGS(A)

[m,n] = size(A);
q=zeros(m,n);
a=A;
r=zeros(n,n);

for i=1:n
    r(i,i)=norm(a(:,i));
    q(:,i)=a(:,i)/r(i,i);
    for j=i+1:n
        r(i,j)=q(:,i)'*a(:,j);
        a(:,j)=a(:,j)-r(i,j)*q(:,i);
    end
end
R=-r; % Matlab's QR works in the negative direction
Q=-q;
```

qrHH.m

```
function [Q,R] = qrHH(A)

[m,n] = size(A);
Q=eye(m);
for k = 1:n
    % Find the HH reflector
    z = A(k:m,k);
    v = [ -sign(z(1))*norm(z) - z(1); -z(2:end) ];
    v = v / sqrt(v'*v);   % remove v'*v in den

    % Apply the HH reflection to each column of A and Q
    for j = 1:n
        A(k:m,j) = A(k:m,j) - v*( 2*(v'*A(k:m,j)) );
    end
    for j = 1:m
        Q(k:m,j) = Q(k:m,j) - v*( 2*(v'*Q(k:m,j)) );
    end

end

Q = Q';
Q = Q(:,1:n); % extract reduced Q
R = triu(A);  % extract triangularity
R = R(1:n,:);
```

test.m

```
function test(A)

I = eye(min(size(A)));

cn=cond(A);

% Matlab QR
[Q,R] = qr(A,0);
qr_mat = norm(Q*R-A)/norm(A);
or_mat = norm(Q'*Q-I);

% Modified GS
[Q,R] = qrMGS(A);
qr_mgs = norm(Q*R-A)/norm(A);
or_mgs = norm(Q'*Q-I);

% Householder
[Q,R] = qrHH(A);
qr_hh= norm(Q*R-A)/norm(A);
or_hh = norm(Q'*Q-I);

fprintf('Condition number: %10.4e\n',cn)
fprintf('\n')
fprintf('QR error\n')
fprintf('Matlab QR: %10.4e\n',qr_mat)
fprintf('Modified GS: %10.4e\n',qr_mgs)
fprintf('Householder: %10.4e\n',qr_hh)
fprintf('\n')
fprintf('Orthogonality error\n')
fprintf('Matlab QR: %10.4e\n',or_mat)
fprintf('Modified GS: %10.4e\n',or_mgs)
fprintf('Householder: %10.4e\n',or_hh)
```