MATH-6800
Vignesh Ramakrishnan
RIN: 662028006
Due: Thursday October 6, 2022

Problem Set 4

1. Consider the Householder reflector, $F = I - 2uu^*, u^*u = 1$. Determine the eigenvalues and eigenvectors, determinant and singular values of F.

   Since, uu^* is symmetrical, F is symmetrical as well.

$$\begin{aligned} F^*F &= \left(I - 2uu^*\right)^* \left(I - 2uu^*\right) \\ F^*F &= I - 2uu^* - 2uu^* + 4uu^*uu^*,\ u^*u = 1 \\ F^*F &= I \end{aligned}$$

   F is orthogonal and since $F = F^*$, its Hermitian. Therefore, it has real eigenvalues. If F has eigenvalues λ, then, F^*F has eigenvalues λ^2. Since, $F^*F = I$, $\lambda^2 = 1$. Therefore, it has singular values $\sigma = 1, 0$ and eigenvalues $\lambda = \pm 1$.

$$Fu = u - 2uu^*u = -u$$

   Therefore, u is definitely one of the eigenvectors and $\lambda = -1$ is surely one of its eigenvalues. Therefore the determinant should be of the form $\det|\lambda I - F| = (\lambda + 1)(\lambda \pm 1)^{m-1} = 0$.

2. General Householder reflector. Let $x, y \in \mathbb{C}^m$, with $m > 1$. Show explicitly (using algebra) that if $||x||_2 = ||y||_2$ then there is Householder reflector $F = I - 2uu^*$, $||u||_2 = 1$, such that $Fx = \alpha y$ where $\alpha = \pm\text{sign}\left(y^*x\right)$. Note: for $z \in \mathbb{C}$, with $z = re^{i\theta}$, $r, \theta \in \mathbb{R}$ and $r \geq 0$ then $\text{sign}\left(z\right) = e^{i\theta}$. (Hint: if $x \neq \alpha y$ consider $v = x - \alpha y$, $u = v/\left||v|\right|_2$.)

   We are trying to project a vector x on to another generic vector y. Hence, we want to prove, $Fx = \pm\text{sign}(y^*x)\frac{||x||_2}{||y||_2}y$. But since, we are told $||x||_2 = ||y||_2$ means we need $Fx = \pm\text{sign}(y^*x)y$. Let, $\alpha = \pm\text{sign}(y^*x)$. Then, if we choose $F = I - 2\frac{vv^*}{v^*v}$ and $v = -x + \alpha y$

$$\begin{aligned} Fx &= x - v\left(2\frac{v^*x}{v^*v}\right) \\ Fx &= x - \beta v = \alpha y \end{aligned}$$

   So, all we need to prove is $\beta = -1$ or in other words, $\frac{v^*x}{v^*v} = \frac{-1}{2}$.

$$\begin{aligned} v^*v &= \left(-x + \alpha y\right)^* \left(-x + \alpha y\right) \\ &= ||x||_2^2 - \alpha\left(x^*y\right) - \alpha^*\left(y^*x\right) + \left(\alpha^*\alpha\right)||y||_2^2 \\ &= 2\,||x||_2^2 - \left(y^*x\right)\left(\alpha + \alpha^*\right) \end{aligned}$$

$$\begin{aligned} v^*x &= \left(-x + \alpha y\right)^* x \\ &= -\,||x||_2^2 + \alpha\left(y^*x\right) \end{aligned}$$

   $\beta = -1$, if $\alpha = \alpha^*$. Now, we need to think about what α is. Lets consider two cases. If $\alpha \in \mathbb{R}$, then this is a trivial case as $\alpha = \alpha^*$, therefore, the Householder reflector F will make $Fx = \alpha y$ true. Since, $\alpha = \pm\text{sign}(y^*x) = \pm\frac{y^*x}{|y^*x|}$. If it is complex in nature, $\alpha + \alpha^*$ will only consist of its $\mathbb{R}$ component. I'm unsure how to complete this proof.

3. Write a Matlab function [W,R] = house(A) that computes an implicit representation of a full or reduced QR factorization for $A \in \mathbb{C}^{m\times n}$ with $m \geq n$ using Householder reflections. The output variables are a lower triangular matrix $W \in \mathbb{C}^{m\times n}$ whose columns are the Householder vectors v_k, and an upper triangular matrix $R \in \mathbb{C}^{n\times n}$.
Also write a Matlab function Q = formQ(W) that takes the matrix W from house and generates the full matrix $Q \in \mathbb{C}^{m\times m}$.

   (a) Test your programs on the Vandermonde matrix from problem set 3 with $m = 5$. Compare Q and R from the Matlab function [Q,R]=qr(A) to the output from house and formQ. Print A along with Q and R for each case along with $||A - QR||_2$, and $||Q^*Q - I||_2$ for each factorization.

   The code to compute Q, R from **house** and **formQ** functions are added in Listing 1, and Listing 2. The code to compare between MATLAB's QR factorization and the Householder factorization is attached in Listing 3 and Listing 4. The Householder algorithm performs well and is comparable to the MATLAB **qr** and it has been included in the PDF **Q3 solution** under the header **Part 3.a**.

```
function [W,R] = house(A)

[m,n] = size(A);
W = zeros(m,n);
R = A;

if m >= n
    for k=1:n
        x  = R(k:m,k);
        l  = length(x);
        e  = zeros(l,1);
        e(1) = 1;
        if x(1)>0
            vk = norm(x)*e + x;
        else
            vk = -norm(x)*e + x;
        end
        vk = vk/norm(vk);
        R(k:m,k:n) = R(k:m,k:n) - 2*vk*(vk'*R(k:m,k:n));
        W(k:m,k) = vk;
    end
else
    disp("Algorithm works only for m >= n");
end

end
```

Listing 1: code to compute R through **[W,R] = house(A)**

```
function Q = formQ(W)

[m,n] = size(W);
Q = eye(m);

if m >=n
    % Algorithm 10.3 - implicit calculation of Q using ei
    for i=1:m          % perform on each column of I
        for k=n:-1:1   % perform Q*ei to get back column of Q
```

```
            vk = W(k:m,k);
            Q(k:m,i) = Q(k:m,i) - 2*vk*(vk'*Q(k:m,i));
        end
    end
else
    disp("Algorithm works only for m >= n");
end

end
```

Listing 2: code to compute Q through **Q = formQ(W)**

```
function [] = QR_func(m)
A = zeros(m);
x = zeros(m,1);

for i=1:m
    x(i,1) = (i-1)/(m-1);
end

for i=1:m
    A(:,i) = x.^(i-1);
end

computeQR(A);

end
```

Listing 3: code to compare MATLAB **qr** and Householder algorithm

```
function [] = computeQR(A)

[m,n] = size(A);

[W, Rh] = house(A);
Qh = formQ(W);
[Qm,Rm] = mgs(A);
[Q, R] = qr(A);

for i=1:n
    if(R(i,i)<0)
        Q(:,i) = -1*Q(:,i);
        R(i,:) = -1*R(i,:);
    end
    if(Rh(i,i)<0)
        Qh(:,i) = -1*Qh(:,i);
        Rh(i,:) = -1*Rh(i,:);
    end
    if(Rm(i,i)<0)
        Qm(:,i) = -1*Qm(:,i);
        Rm(i,:) = -1*Rm(i,:);
    end
end

disp("MATLAB - norm2(A-QR) = ");
disp(norm(A-Q*R));

disp("Householder - norm2(A-QhRh) = ");
disp(norm(A-Qh*Rh));

disp("MGS - norm2(A-QmRm) = ");
disp(norm(A-Qm*Rm));

disp("norm2(Qh-Q) = ");
disp(norm(Qh-Q));

disp("norm2(Rh-R) = ");
disp(norm(Rh-R));

disp("norm2(Qm-Q) = ");
disp(norm(Qm-Q(:,1:n)));

disp("norm2(Rm-R) = ");
disp(norm(Rm-R(1:n,:)));

disp("Householder - norm2(Qh^*Qh - I)");
disp(norm(Qh'*Qh - eye(m)));

disp("MGS - norm2(Qm^*Qm - I)");
disp(norm(Qm'*Qm - eye(n)));

disp("MATLAB - norm2(Q^*Q - I)");
disp(norm(Q'*Q - eye(m)));

end
```

Listing 4: computing **QR** and printing comparison results

(b) (NLA 10.3) Let Z be the matrix

$$Z = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 7 \\ 4 & 2 & 3 \\ 4 & 2 & 2 \end{bmatrix}$$

Compute three reduced QR factorizations of Z in Matlab: by the Gram-Schmidt routine mgs, by the householder routines house and formQ and by Matlab's builtin command qr. Compare the three results and comment on the differences.

The **mgs** algorithm is attached in Listing 5. It is important to note that **mgs** algorithm written here produces a reduced QR factorization where $Q \in \mathbb{C}^{m\times n}$ and $R \in \mathbb{C}^{n\times n}$. Hence, the norms computed and having a lesser norm value when computing $||Q_m^* Q_m - I||_2$ or while computing $||Q_m - Q(:, 1:n)||_2$ is not really comparable because many other columns worth of norm is missing and not added. The Householder algorithm and MATLAB's **qr** performed equally well, with MATLAB's inbuilt **qr** performing slightly better. These two are easily comparable because full QR factorization is generated in both of these cases.

```
function [Qm, Rm] = mgs(A)

[m,n] = size(A);
Qm = zeros(m,n);
Rm = zeros(n);

```

```
V = A;
if m >= n
    for i=1:n
        Rm(i,i) = norm(V(:,i));
        Qm(:,i) = V(:,i)/Rm(i,i);
        for j=i+1:n
            Rm(i,j) = Qm(:,i)'*V(:,j);
            V(:,j) = V(:,j) - Rm(i,j)*Qm(:,i);
        end
    end
else
    disp("Algo works only if m >=n");
end

end
```

Listing 5: code to compute QR using Modified GS Algorithm

```
---- Problem 3.a ----
MATLAB - norm2(A-QR) =
   1.0144e-15

Householder - norm2(A-QhRh) =
   1.7708e-15

MGS - norm2(A-QmRm) =
     0

norm2(Qh-Q) =
   1.5924e-15

norm2(Rh-R) =
   1.0238e-15

norm2(Qm-Q) =
   2.6317e-14

norm2(Rm-R) =
   5.1851e-16

Householder - norm2(Qh^*Qh - I)
   9.7734e-16

MGS - norm2(Qm^*Qm - I)
   2.6493e-14

MATLAB - norm2(Q^*Q - I)
   7.7604e-16

---- Problem 3.b ----
MATLAB - norm2(A-QR) =
   1.9769e-15

Householder - norm2(A-QhRh) =
   2.3564e-15

MGS - norm2(A-QmRm) =
   1.3636e-15

norm2(Qh-Q) =
   1.1344e-15

norm2(Rh-R) =
   1.2982e-15

norm2(Qm-Q) =
   7.0976e-16

norm2(Rm-R) =
   6.6613e-16

Householder - norm2(Qh^*Qh - I)
   9.0619e-16

MGS - norm2(Qm^*Qm - I)
   7.1498e-16
```

```
MATLAB - norm2(Q^*Q - I)
   4.4314e-16

>>
```

4. Take $m = 50, n = 12$. Create a vector $t \in \mathbb{R}^m$ of equally spaced points for $t \in [0, 1]$. Create the $m \times n$ Vandermonde matrix using the points t to build the matrix associated with solving a least squares fit of a polynomial of degree $n - 1$ to the function $\cos 4t$ so that the right-hand-side is $b = \cos 4t$.
Solve the least squares problem in 7 ways:

   (a) by forming and solving the normal equations.

   (b) using a QR factorization and Classical Gram-Schmidt, **clgs**.

   (c) using a QR factorization and modified Gram-Schmidt, **mgs**.

   (d) using a QR factorization and the Householder triangularization function, **house** (Exercise 3.).

   (e) using a QR factorization and Matlab's **qr**.

   (f) using Matlab's backslash function: $x = A \setminus b$.

   (g) using the SVD, using Matlab's svd function.

   The script used to work on this problem is attached in Listing 6. The final results of the solution vectors $x_a, x_b, x_c, x_d, x_e, x_f, x_g$ are

```
clc
clear
%%

m = 50;
n = 12;

t = (0.02:0.02:1)';
b = cos(4*t);
A = zeros(m,n);

for i=1:n
    A(:,i) = t.^(i-1);
end

%% a - normal equations
xa = (A'*A)\(A'*b);
%% b - clgs QR factorization
[Qc,Rc] = clgs(A);
bprime = Qc'*b;

xb = zeros(n,1);
for i=n:-1:1
    if i==n
        xb(i) = bprime(i)/Rc(i,i);
    else
        rv = Rc(i,i+1:n);
        xv = xb(i+1:n,1);
        xb(i) = (bprime(i)-rv*xv)/Rc(i,i);
    end
end
%% c - mgs QR factorization
[Qm,Rm] = mgs(A);
bprime = Qm'*b;

xc = zeros(n,1);
```

```
for i=n:-1:1
    if i==n
        xc(i) = bprime(i)/Rm(i,i);
    else
        rv = Rm(i,i+1:n);
        xv = xc(i+1:n,1);
        xc(i) = (bprime(i)-rv*xv)/Rm(i,i);
    end
end
%% d - Householder QR factorization
[W,Rh] = house(A);
Qh = formQ(W);
bprime = Qh'*b;

xd = zeros(n,1);
for i=n:-1:1
    if i==n
        xd(i) = bprime(i)/Rh(i,i);
    else
        rv = Rh(i,i+1:n);
        xv = xd(i+1:n,1);
        xd(i) = (bprime(i)-rv*xv)/Rh(i,i);
    end
end
%% e - MATLAB QR factorization
[Q,R] = qr(A);
bprime = Q'*b;

xe = zeros(n,1);
for i=n:-1:1
    if i==n
        xe(i) = bprime(i)/R(i,i);
    else
        rv = R(i,i+1:n);
        xv = xe(i+1:n,1);
        xe(i) = (bprime(i)-rv*xv)/R(i,i);
    end
end
%% f - using MATLAB backslash
xf = A\b;
%% g - using MATLAB SVD
[U,S,V] = svd(A);
bprime = U'*b;
w = S\bprime;
xg = V*w;
%%
fprintf('Normal \t\t\t CLGS \t\t\t MGS \t\t\t HOUSE\n');
for i=1:n
    fprintf('%22.15e %22.15e %22.15e %22.15e\n',xa(i),xb(i),xc(i),xd(i))
end

    fprintf(' Matlab-QR \t\t Matlab-backslash \t\t SVD\n');
for i=1:n
    fprintf('%22.15e %22.15e %22.15e\n',xe(i),xf(i),xg(i))
end
```

Listing 6: code used to work on Q4

Normal Equations	CLGS	MGS
$\underline{9.999999109532806e-01}$	$1.000077647224682e+00$	$1.000000041170736e+00$
$\underline{6.754693828649009e-06}$	$-4.971932917314963e-03$	$-3.292423175880044e-06$
$-8.000167020767115e+00$	$-7.900928661862072e+00$	$-7.999912404059410e+00$
$1.956588310083076e-03$	$-9.134144352945499e-01$	$-1.117513335143340e-03$
$1.065384192005753e+01$	$1.529138263478292e+01$	$1.067469059127887e+01$
$5.150512336971136e-02$	$-1.402801189535030e+01$	$-3.520678913475670e-02$
$-5.822730048918321e+00$	$2.088674498037030e+01$	$-5.590912260411920e+00$
$2.338164329569813e-01$	$-3.171249102409233e+01$	$-1.722995276464837e-01$
$\underline{1.339502483110237e+00}$	$2.483857530331423e+01$	$1.802985799305330e+00$
$2.602954422066424e-01$	$-9.584384521068278e+00$	$-7.149614546700377e-02$
$-4.781462589853315e-01$	$1.445873119716408e+00$	$-3.429048488872990e-01$
$\underline{1.064750587932793e-01}$	$2.789734679148956e-02$	$8.253272870515629e-02$

Householder	MATLAB-QR	MATLAB-backslash
$\underline{1.000000018290838e+00}$	$1.000000018290851e+00$	$1.000000018290830e+00$
$\underline{-1.534554746056279e-06}$	$-1.534555301811576e-06$	$-1.534553796144743e-06$
$-7.999956768854704e+00$	$-7.999956768845637e+00$	$-7.999956768878433e+00$
$-5.837197769989676e-04$	$-5.837198522172034e-04$	$-5.837195081694597e-04$
$1.067108043299089e+01$	$1.067108043335647e+01$	$1.067108043128556e+01$
$-2.022528545774925e-02$	$-2.022528657665968e-02$	$-2.022527878420121e-02$
$-5.630891699577981e+00$	$-5.630891697338319e+00$	$-5.630891716486514e+00$
$-1.023642159034818e-01$	$-1.023642188756126e-01$	$-1.023641876466265e-01$
$\underline{1.723269198606801e+00}$	$1.723269201196124e+00$	$1.723269167674754e+00$
$-1.448810203183768e-02$	$-1.448810345307323e-02$	$-1.448808069556081e-02$
$-3.661218670059542e-01$	$-3.661218665613033e-01$	$-3.661218754177160e-01$
$\underline{8.663992350974073e-02}$	$8.663992344949385e-02$	$8.663992495468259e-02$

SVD
$1.000000018290850e+00$
$-1.534555290460902e-06$
$-7.999956768845543e+00$
$-5.837198536591567e-04$
$1.067108043336462e+01$
$-2.022528660314774e-02$
$-5.630891697284317e+00$
$-1.023642189454010e-01$
$1.723269201251188e+00$
$-1.448810347678534e-02$
$-3.661218665574340e-01$
$8.663992344973712e-02$

The normal equations do exhibit numerical instability. Householder QR, MATLAB-QR, MATLAB-Backlash and SVD techniques exhibited extremely similar error norm $||Ax-b||_2$ of $6.8208e-09$. The most error was observed when QR decomposition was done using the CLGS method. The norm of this error observed using Normal Equations was one order higher than the error norm observed using Householder QR, MATLAB-QR, MATLAB-Backlash and SVD techniques.
Comparing the underlined values between x_a and x_d, we can clearly see how we have lost so

many significant digits in x_a. This is because of the very poor conditioning of the matrix (A^*A). Computing the inverse of an almost singular matrix will lead to this problem.