Problem 2

Table of Contents

I need to ensure: 1. Q is unitary 2. R is upper triangular 3. QR = A Compare results to matlabs qr function and comment on comparision Answer: Is QR factorization unique?

Generate 10 random complex matrices

```
clear all
close all

% pick the dimension of the matrices which are mxn
m = 2;
n = 4;
mat = zeros(m,n,10);
% generate random complex numbers in the interval (a,b) to fill matrices
a = 0;
b = 10;
for z = 1:10,
    r = a + (b-a).*rand(m*n,1); %real part
    c = a + (b-a).*rand(m*n,1); %complex part
    cn = r + 1i*c; %complex numbers to fill matrix with
    p = 1;
    for i = 1:m, %fill matrix here
        for j = 1:n,
            mat(i,j,z) = cn(p);
            p = p+1;
        end
    end
end
```

Factor each matrix using my qr function

```
Q_my = zeros(m,m,10);
R_my = zeros(m,n,10);
for i = 1:10,
    [Q_my(:,:,i),R_my(:,:,i)] = myqr(mat(:,:,i));
end
%check if Q is unitary, if R is Upper triangular, and if QR = mat
%Check if Q is unitary
check1 = zeros(m,m);
check2 = zeros(m,m);
for i = 1:10,
    check1(:,:) = eye(m) - Q_my(:,:,i)'*Q_my(:,:,i);
    check2(:,:) = eye(m) - Q_my(:,:,i)*Q_my(:,:,i)';
```

```
    if any(any(or(abs(check1) > 1e-6, abs(check2) > 1e-6))),
        disp(strcat('Q number: ', num2str(i), ' is not unitary'))
    end
    %Check if R is upper Triangular
    sum_tot = 0;
    if m > n,
        for j = 1:n,
            R = R_my(:,:,i);
            sum_tot = sum(R(j+1,j)) + sum_tot;
            if sum_tot > 1e-6,
                disp(strcat('R number: ', num2str(i), ' is not upper triangular'))
            end
        end
    else
        for j = 1:m-1,
            R = R_my(:,:,i);
            sum_tot = sum(R(j+1,j)) + sum_tot;
            if sum_tot > 1e-6,
                disp(strcat('R number: ', num2str(i), ' is not upper triangular'))
            end
        end
    end
    %Check if QR = mat
    if any(any(abs(Q_my(:,:,i)*R_my(:,:,i) - mat(:,:,i)) > 1e-6)),
        disp(strcat('Q',num2str(i),'*R',num2str(i),'!=mat',num2str(i)))
    end
end
```

Factor each matrix using matlab's qr function and compare

```
Q_mat = zeros(m,m,10);
R_mat = zeros(m,n,10);
for i = 1:10,
    [Q_mat(:,:,i),R_mat(:,:,i)] = qr(mat(:,:,i));
end

%Note that my answer is not equal to the built in matlab answer; however,
%it is equally valid. So the QR factorization is not unique. This is
%because there are multiple transformations to derive Q and R. I used the
%Householder transformation, while matlab likely uses a different one
%(maybe the Givens rotation). I have displayed one of the ten below to
%demonstrate my answers. Both the matlab algorithm and my algorithm
%successfully factor the matrix (i.e. QR = A for both algorithms).

Q_matlab = Q_mat(:,:,1)
Q_my_alg = Q_my(:,:,1)
R_matlab = R_mat(:,:,1)
R_my_alg = R_my(:,:,1)
QR_matlab = Q_mat(:,:,1)*R_mat(:,:,1)
QR_my_alg = Q_my(:,:,1)*R_my(:,:,1)
```

```
clear a b c check1 check2 cn i j m n p r R sum_tot z
```

```
Q_matlab =

  -0.2542 - 0.9628i  -0.0616 + 0.0677i
  -0.0886 - 0.0227i  -0.3168 - 0.9441i


Q_my_alg =

  -0.9958 - 0.0000i   0.0446 + 0.0799i
  -0.0446 + 0.0799i  -0.9958 - 0.0000i


R_matlab =

  -6.8222 + 0.0000i  -7.6172 + 1.3864i  -5.4292 + 7.2341i  -6.5914 + 6.6384i
   0.0000 + 0.0000i -10.3708 + 0.0000i  -3.5618 + 3.7019i  -2.5259 + 2.7888i


R_my_alg =

  -1.7412 - 6.5963i  -3.2846 - 7.0111i  -8.3802 - 3.4031i  -8.1008 - 4.6788i
  -0.0000 + 0.0000i  -3.2998 - 9.8319i  -4.6428 - 2.1989i  -3.4475 - 1.5073i


QR_matlab =

   1.7339 + 6.5686i   3.9094 + 6.2797i   8.3138 + 2.9198i   8.0336 + 4.3165i
   0.6047 + 0.1549i   3.9926 + 9.8406i   5.2688 + 1.6717i   4.1680 + 1.0622i


QR_my_alg =

   1.7339 + 6.5686i   3.9094 + 6.2797i   8.3138 + 2.9198i   8.0336 + 4.3165i
   0.6047 + 0.1549i   3.9926 + 9.8406i   5.2688 + 1.6717i   4.1680 + 1.0622i
```

Published with MATLAB® R2014b