MATLAB - 4

Q1

```
% Define two vectors
v1 = [1; 2; 3];
v2 = [2; 4; 6];
%Find the basis for orthogonal complement
A=[v1', v2']
```

```
A = 1×6
     1     2     3     2     4     6
```

```
N=null(A)
```

```
N = 6×5
   -0.2390   -0.3586   -0.2390   -0.4781   -0.7171
    0.9490   -0.0766   -0.0510   -0.1021   -0.1531
   -0.0766    0.8852   -0.0766   -0.1531   -0.2297
   -0.0510   -0.0766    0.9490   -0.1021   -0.1531
   -0.1021   -0.1531   -0.1021    0.7958   -0.3063
   -0.1531   -0.2297   -0.1531   -0.3063    0.5406
```

```
b1=N(:,1)
```

```
b1 = 6×1
   -0.2390
    0.9490
   -0.0766
   -0.0510
   -0.1021
   -0.1531
```

```
b2=N(:,2)
```

```
b2 = 6×1
   -0.3586
   -0.0766
    0.8852
   -0.0766
   -0.1531
   -0.2297
```

```
% Calculate the span of given vectors
span = linspace(-30, 30, 10);
[X, Y,Z] = meshgrid(span, span,span);
plane_points = v1 * X(:)' + v2 * Y(:)'
```

```
plane_points = 3×1000
  -90.0000  -76.6667  -63.3333  -50.0000  -36.6667  -23.3333  -10.0000    3.3333 ···
 -180.0000 -153.3333 -126.6667 -100.0000  -73.3333  -46.6667  -20.0000    6.6667
 -270.0000 -230.0000 -190.0000 -150.0000 -110.0000  -70.0000  -30.0000   10.0000
```

```
% Calculate the span of basis of orthogonal complement
oplane_points = b1 * X(:)' + b2 * Y(:)'
```

```
oplane_points = 6×1000
   17.9284   15.5380   13.1475   10.7571    8.3666    5.9761    3.5857    1.1952 ···
```

```
  -26.1718  -26.6823  -27.1927  -27.7031  -28.2135  -28.7239  -29.2344  -29.7448
  -24.2578  -18.3567  -12.4557   -6.5547   -0.6536    5.2474   11.1484   17.0495
    3.8282    3.3177    2.8073    2.2969    1.7865    1.2761    0.7656    0.2552
    7.6563    6.6355    5.6146    4.5938    3.5730    2.5521    1.5313    0.5104
   11.4845    9.9532    8.4220    6.8907    5.3594    3.8282    2.2969    0.7656
```

```
figure;
% Plot the span of given vectors
scatter3(plane_points(1,:), plane_points(2,:), plane_points(3,:), 'r',
'MarkerFaceColor', 'r');
hold on;
% Plot the span of basis of orthogonal complement
scatter3(oplane_points(1,:), oplane_points(2,:), oplane_points(3,:), 'k',
'MarkerFaceColor', 'k');
% Plot vectors
quiver3(0, 0, 0, v1(1), v1(2), v1(3), 'b');
quiver3(0, 0, 0, v2(1), v2(2), v2(3), 'b');

% Set labels and title
xlabel('x');
ylabel('y');
zlabel('z');
title('orthogonal complement');

% Set aspect ratio to be equal
axis equal;

% Show grid and hold off
grid on;
hold off;
```

Q2

```
% MATLAB Code for Gram-Schmidt Process in R^3
clc; clear; close all;

% Define three linearly independent vectors in R^3
A = [1 3 0; -1 3 0; 0 0 2]; % Example matrix with column vectors

% Number of vectors
[m, n] = size(A);

% Initialize orthonormal basis matrix
Q = zeros(m, n);

% Gram-Schmidt Process
for i = 1:n
   v = A(:, i); % Take the i-th column of A

   for j = 1:i-1
```

```
        v = v - (dot(Q(:, j), A(:, i)) / dot(Q(:, j), Q(:, j))) * Q(:, j);
    end

    Q(:, i) = v / norm(v); % Normalize the vector
end

% Display the orthonormal basis
disp('Orthonormal basis Q:');
```

```
Orthonormal basis Q:
```

```
disp(Q);
```

```
    0.7071    0.7071         0
   -0.7071    0.7071         0
         0         0    1.0000
```

```

% Verify orthogonality (Q' * Q should be identity)
disp('Verification (Q^T * Q):');
```

```
Verification (Q^T * Q):
```

```
disp(Q' * Q);
```

```
    1.0000         0         0
         0    1.0000         0
         0         0    1.0000
```

Q3

```
clc; clear; close all;

% Define the matrix A
A = [1 3 0; -1 3 0; 0 0 2];

% Get the size of A
[m, n] = size(A);

% Initialize Q (orthonormal) and R (upper triangular)
Q = zeros(m, n);
R = zeros(n, n);

% Gram-Schmidt Process for QR Decomposition
for i = 1:n
    v = A(:, i); % Take the i-th column of A

    for j = 1:i-1
        R(j, i) = dot(Q(:, j), A(:, i)); % Compute R(j, i)
        v = v - R(j, i) * Q(:, j); % Subtract projection
    end

```

```
    R(i, i) = norm(v); % Compute R(i, i)

    % Avoid division by zero (check for linear dependence)
    if R(i, i) < 1e-10
        disp('Linearly dependent column detected, skipping...');
        continue;
    end

    Q(:, i) = v / R(i, i); % Normalize to get Q(:, i)
end

% Display results
disp('Orthonormal matrix Q:');
```

```
Orthonormal matrix Q:
```

```
disp(Q);
```

```
    0.7071    0.7071         0
   -0.7071    0.7071         0
         0         0    1.0000
```

```
disp('Upper triangular matrix R:');
```

```
Upper triangular matrix R:
```

```
disp(R);
```

```
    1.4142         0         0
         0    4.2426         0
         0         0    2.0000
```

```
% Verification: A should be approximately equal to Q * R
disp('Verification (A ≈ Q * R):');
```

```
Verification (A ≈ Q * R):
```

```
disp(Q * R);
```

```
     1     3     0
    -1     3     0
     0     0     2
```

Q4

```
clc; clear; close all;

% Define the matrix A
A = [1 0 -1; 3 -1 -1; 2 -1 0];

% Get the size of A
[m, n] = size(A);
```

```
% Initialize Q (orthonormal) and R (upper triangular)
Q = zeros(m, n);
R = zeros(n, n);

% Gram-Schmidt Process for QR Decomposition
for i = 1:n
   v = A(:, i); % Take the i-th column of A

   for j = 1:i-1
       R(j, i) = dot(Q(:, j), A(:, i)); % Compute R(j, i)
       v = v - R(j, i) * Q(:, j); % Subtract projection
   end

   R(i, i) = norm(v); % Compute R(i, i)

   % Avoid division by zero (check for linear dependence)
   if R(i, i) < 1e-10
       disp('Linearly dependent column detected, skipping...');
       continue;
   end

   Q(:, i) = v / R(i, i); % Normalize to get Q(:, i)
end
```

```
Linearly dependent column detected, skipping...
```

```
% Display results
disp('Orthonormal matrix Q:');
```

```
Orthonormal matrix Q:
```

```
disp(Q);
```

```
    0.2673    0.7715         0
    0.8018    0.1543         0
    0.5345   -0.6172         0
```

```
disp('Upper triangular matrix R:');
```

```
Upper triangular matrix R:
```

```
disp(R);
```

```
    3.7417   -1.3363   -1.0690
         0    0.4629   -0.9258
         0         0    0.0000
```

```
if rank(A)==n
   [Q , R]=qr(A)
else
   disp('Orthonormal basis is not possible')
```

```
end
```

```
Orthonormal basis is not possible
```

```
% Verification: A should be approximately equal to Q * R
disp('Verification (A = Q * R):');
```

```
Verification (A = Q * R):
```

```
disp(Q * R);
```

```
    1.0000         0   -1.0000
    3.0000   -1.0000   -1.0000
    2.0000   -1.0000    0.0000
```

QR Decomposition

Q1

```
%define matrix
A=[1 0 0; 1 1 0; 1 1 1]
```

```
A = 3×3
     1     0     0
     1     1     0
     1     1     1
```

```
[Q,R]=qr(A)
```

```
Q = 3×3
   -0.5774    0.8165   -0.0000
   -0.5774   -0.4082   -0.7071
   -0.5774   -0.4082    0.7071
R = 3×3
   -1.7321   -1.1547   -0.5774
         0   -0.8165   -0.4082
         0         0    0.7071
```

```
[Q,R]=qr(A')
```

```
Q = 3×3
     1     0     0
     0     1     0
     0     0     1
R = 3×3
     1     1     1
     0     1     1
     0     0     1
```

```
[Q,R]=qr(inv(A))
```

```
Q = 3×3
   -0.7071   -0.4082    0.5774
    0.7071   -0.4082    0.5774
         0    0.8165    0.5774
R = 3×3
   -1.4142    0.7071         0
         0   -1.2247    0.8165
         0         0    0.5774
```

PRACTICE PROBLEMS

- Find the orthogonal complement of subspace spanned by {(1,2,3),(4,5,6)}

```
% Define two vectors
v1 = [1; 2; 3];
v2 = [4; 5; 6];
%Find the basis for orthogonal complement
A=[v1', v2']
```

```
A = 1×6
     1     2     3     4     5     6
```

```
N=null(A)
```

```
N = 6×5
   -0.2097   -0.3145   -0.4193   -0.5241   -0.6290
    0.9602   -0.0597   -0.0796   -0.0995   -0.1194
   -0.0597    0.9105   -0.1194   -0.1492   -0.1790
   -0.0796   -0.1194    0.8409   -0.1989   -0.2387
   -0.0995   -0.1492   -0.1989    0.7513   -0.2984
   -0.1194   -0.1790   -0.2387   -0.2984    0.6419
```

```
b1=N(:,1)
```

```
b1 = 6×1
   -0.2097
    0.9602
   -0.0597
   -0.0796
   -0.0995
   -0.1194
```

```
b2=N(:,2)
```

```
b2 = 6×1
   -0.3145
   -0.0597
    0.9105
   -0.1194
   -0.1492
   -0.1790
```

```
% Calculate the span of given vectors
span = linspace(-30, 30, 10);
[X, Y,Z] = meshgrid(span, span,span);
plane_points = v1 * X(:)' + v2 * Y(:)'
```

```
plane_points = 3×1000
 -150.0000 -123.3333  -96.6667  -70.0000  -43.3333  -16.6667   10.0000   36.6667 ···
 -210.0000 -176.6667 -143.3333 -110.0000  -76.6667  -43.3333  -10.0000   23.3333
 -270.0000 -230.0000 -190.0000 -150.0000 -110.0000  -70.0000  -30.0000   10.0000
```

```
% Calculate the span of basis of orthogonal complement
oplane_points = b1 * X(:)' + b2 * Y(:)'
```

```
oplane_points = 6×1000
   15.7243   13.6277   11.5311    9.4346    7.3380    5.2414    3.1449    1.0483 ···
  -27.0161  -27.4139  -27.8118  -28.2097  -28.6075  -29.0054  -29.4032  -29.8011
  -25.5241  -19.4543  -13.3844   -7.3145   -1.2446    4.8253   10.8952   16.9651
    5.9678    5.1721    4.3764    3.5807    2.7850    1.9893    1.1936    0.3979
    7.4598    6.4651    5.4705    4.4759    3.4812    2.4866    1.4920    0.4973
    8.9517    7.7582    6.5646    5.3710    4.1775    2.9839    1.7903    0.5968
```

```
figure;
% Plot the span of given vectors
scatter3(plane_points(1,:), plane_points(2,:), plane_points(3,:), 'r',
'MarkerFaceColor', 'r');
hold on;
% Plot the span of basis of orthogonal complement
scatter3(oplane_points(1,:), oplane_points(2,:), oplane_points(3,:), 'k',
'MarkerFaceColor', 'k');
% Plot vectors
quiver3(0, 0, 0, v1(1), v1(2), v1(3), 'b');
quiver3(0, 0, 0, v2(1), v2(2), v2(3), 'b');

% Set labels and title
xlabel('x');
ylabel('y');
zlabel('z');
title('orthogonal complement');

% Set aspect ratio to be equal
axis equal;

% Show grid and hold off
grid on;
hold off;
```

- Find the orthogonal complement of subspace spanned by {(1,2)}

```
%Define two vectors
v1 = [1; 2];
%Find the basis for orthogonal compliment
A = [v1']
```

```
A = 1×2
     1     2
```

```
N=null(A)
```

```
N = 2×1
   -0.8944
    0.4472
```

```
b1=N(:,1)
```

```
b1 = 2×1
   -0.8944
    0.4472
```

```
%Calculate the span of given vectors
span = linspace(-30, 30, 10);
[X, Y]= meshgrid(span, span);
plane_points = v1 * X(:)'
```

```
plane_points = 2×100
  -30.0000  -30.0000  -30.0000  -30.0000  -30.0000  -30.0000  -30.0000  -30.0000 ···
  -60.0000  -60.0000  -60.0000  -60.0000  -60.0000  -60.0000  -60.0000  -60.0000
```

```
%Calculate the span of basis of orthogonal compliment
oplane_points = b1 * X(:)'
```

```
oplane_points = 2×100
   26.8328   26.8328   26.8328   26.8328   26.8328   26.8328   26.8328   26.8328 ···
  -13.4164  -13.4164  -13.4164  -13.4164  -13.4164  -13.4164  -13.4164  -13.4164
```

```
figure;
set(gcf, 'Position', [100, 100, 900, 700]); % Adjust figure size
%Plot the span of given vectors
scatter(plane_points(1,:), plane_points(2,:), 'r.', 'MarkerFaceColor','r');
hold on;
%Plot the span of basis of orthogonal compliment
scatter(oplane_points(1,:), oplane_points(2,:), 'k.', 'MarkerFaceColor','k');
%Plot vectors
quiver(0, 0, v1(1), v1(2), 'b', 'LineWidth', 2, 'MaxHeadSize', 0.5);
quiver(0, 0, b1(1), b1(2), 'g', 'LineWidth', 2, 'MaxHeadSize', 0.5);
%Set labels and title
xlabel('X', 'FontSize', 14);
ylabel('Y', 'FontSize', 14);
title('orthogonal compliment', 'FontSize', 16);
% Set aspect ratio to be equal
axis equal;
```

```
%Show grid and hold off
grid on;
hold off;
```

- Apply Gram Schmidt Process to find Q and hence find the QR Decomposition of A

```
%Gram-Schmidt Process and QR Decomposition
clc; clear; close all;
%Define the matrix  A
A = [1 0 2; 0 1 1; 1 2 0];
% Get the size of A
[m, n]=size(A);
%Initialize Q (orthonormal) and R (upper triangular)
Q = zeros(m, n);
R = zeros(n, n);
%Gram-Schmidt Process for QR Decomposition
for i =1:n
   v = A(:,i); %Take the i-th column of a matrix
   for j = 1:i-1
       R(j, i) = dot(Q(:, j), A(:, i)); %Compute R(j, i)
        v = v - R(j, i) * Q(:, j); % Subtract projection
   end
   R(i,i) = norm(v); % Compute R(i, i)
   %Avoid division by zero (Check for linear dependence)
   if R(i, i) < 1e-10
       disp('Linearly dependent column detected, skipping...');
       continue;
   end
   Q(:, i) = v / R(i,i); %NOrmalize to get Q(:,i)
end
%Display results
disp('Orthonormal matrix Q:');
```

```
Orthonormal matrix Q:
```

```
disp(Q);
```

```
    0.7071   -0.5774    0.4082
         0    0.5774    0.8165
    0.7071    0.5774   -0.4082
```

```
disp('Upper triangular matrix R:');
```

```
Upper triangular matrix R:
```

```
disp(R);
```

```
    1.4142    1.4142    1.4142
         0    1.7321   -0.5774
         0         0    1.6330
```

```
if rank(A)==n
   [Q R] = qr(A)
else
   disp('Orthonormal basis is not possible')
end
```

```
Q = 3×3
   -0.7071    0.5774   -0.4082
         0   -0.5774   -0.8165
   -0.7071   -0.5774    0.4082
R = 3×3
   -1.4142   -1.4142   -1.4142
         0   -1.7321    0.5774
         0         0   -1.6330
```

```
 %Verification : A should be approximately equal to Q * R
 disp('Verification (A ??? Q * R): ');
```

```
Verification (A ??? Q * R):
```

```
 disp(Q * R);
```

```
    1.0000   -0.0000    2.0000
         0    1.0000    1.0000
    1.0000    2.0000   -0.0000
```