MathTools HW1

```
%% 2024-9-13
%Question 1
```

Tetsing

```
%a)Get projection of V onto U
u = [0;1];
v = [2;3];
question1a = projection(u,v);
disp('Returned vector for 1a:');
```

```
Returned vector for 1a:
```

```
disp(question1a);
```

```
     0
     3
```

```
%b)Get component V that is orthogonal to U
question1b = ortho(u,v);
disp('Returned vector for 1b:');
```

```
Returned vector for 1b:
```

```
disp(question1b);
```

```
     2
     0
```

```
%c)Returns distance
question1c = distance(u,v);
disp('Returned distance:');
```

```
Returned distance:
```

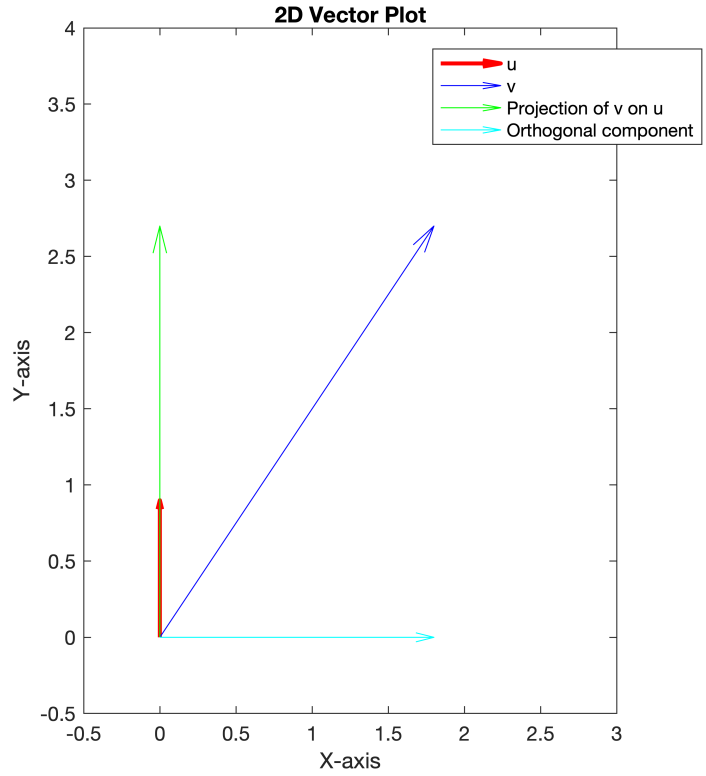
```
disp(question1c);
```

```
     2
```

Plotting

```
% figure;
% quiver(0, 0, u(1), u(2), 'r'); hold on; % Vector u
% quiver(0, 0, v(1), v(2), 'b');          % Vector v
% question1a = projection(u,v)
% quiver(0, 0, question1a(1), question1a(2), 'g');    % Projection of v onto u
% question1b = ortho(u,v)
% quiver(question1a(1), question1a(2), question1b(1), question1b(2), 'k');  % Orthogona
% axis equal;
% grid on;
% legend('u', 'v', 'Projection of v on u', 'Orthogonal component');
```

```
figure;
quiver(0, 0, u(1), u(2), 'r','LineWidth',2); hold on;% Vector u
quiver(0, 0, v(1), v(2), 'b'); hold on;% Vector v
quiver(0, 0, question1a(1), question1a(2), 'g'); hold on;% Projection
quiver(0, 0, question1b(1), question1b(2), 'c'); hold on;% Orthogonal
axis equal;
xlabel('X-axis'); ylabel('Y-axis');
title('2D Vector Plot');
xlim([-0.5, max(v(1)) + 1]); ylim([-0.5, max(v(2)) + 1]);
legend('u', 'v', 'Projection of v on u', 'Orthogonal component', Location='best');
hold off;
```




```
function question1a = projection(u,v)
question1a = (sum(v.*u)/sum(u.*u))*u;
end

function question1b = ortho(u,v)
question1a = projection(u,v);
question1b = v-question1a;
end

function question1c = distance(u,v)
question1b = ortho(u,v);
question1c = sqrt(sum(question1b .^ 2));
end
```

MathTools HW 1

Question 2

a) It's NOT a linear system. Since it doesn't follow the rule for both the additivity and homogeneity properties for a linear system. If the input is [0], no linear system matrix M can produce a non-zero output.

Q2.

Linear system { ① superposition
② additivity
③ homogeneity

b) $\because \begin{bmatrix} 2 \\ 3 \end{bmatrix} \cdot M = 9 = 2m_1 + 3m_2$

$\begin{bmatrix} -1 \\ 2 \end{bmatrix} \cdot M = 4 = -m_1 + 2m_2$

Output scalar. so M must be a row vector).

$M = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$

$\therefore m_1 + 5m_2 = 13$

$m_1 = 13 - 5m_2$

$\therefore -13 + 5m_2 + 2m_2 = 4$

$7m_2 = 17$

$m_2 = \frac{17}{7}$

$m_1 = 13 - 5 \times \frac{17}{7} = \frac{6}{7}$

$\therefore M = \begin{bmatrix} \frac{6}{7}, & \frac{17}{7} \end{bmatrix}$

$\therefore$ The system can be linear.

M is unique

c). $\because \begin{bmatrix} -1 \\ 2 \end{bmatrix} M = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$

$\begin{bmatrix} 3 \\ 1 \end{bmatrix} M = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$

$\begin{bmatrix} 0 \\ 7 \end{bmatrix} M = \begin{bmatrix} -2 \\ 7 \end{bmatrix}$

$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$\therefore \begin{cases} -a + 2b = 0 \\ -c + 2d = 2 \end{cases}$ ①

$\begin{cases} 3a + b = -1 \\ 3c + d = 1 \end{cases}$ ②

$\begin{cases} 0a + 7b = -2 \\ 0c + 7d = 7 \end{cases}$ ③

$\therefore b = -\frac{2}{7}, \quad d = 1$

in ①, $-a - \frac{4}{7} = 0 \Rightarrow a = -\frac{4}{7}$

$-c + 2 = 2 \Rightarrow c = 0$

in ②, $3a - \frac{2}{7} = -1 \Rightarrow a = -\frac{5}{21}$

$3c + 1 = 1 \Rightarrow c = 0$

Thus, a in ② $\neq a$ in ①.

So, it's not a linear system

d). $\begin{bmatrix} 3 \\ 2 \end{bmatrix} M = \begin{bmatrix} -4 \\ 1 \end{bmatrix}$ $\quad M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$\begin{bmatrix} -6 \\ -4 \end{bmatrix} M = \begin{bmatrix} 8 \\ -2 \end{bmatrix}$

① $\begin{cases} 3a + 2b = -4 \\ 3c + 2d = 1 \end{cases}$

② $\begin{cases} -6a - 4b = 8 \\ -6c - 4d = -2 \end{cases}$

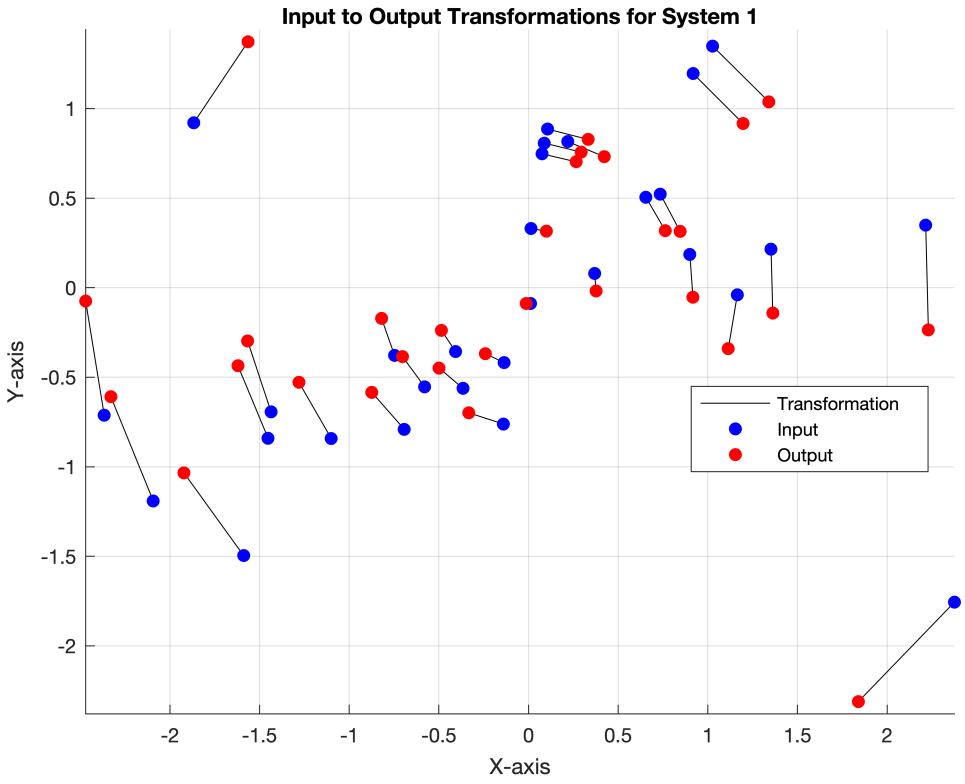
$\Rightarrow \quad a = \dfrac{-4 - 2b}{3} \qquad c = \dfrac{1 - 2d}{3}$

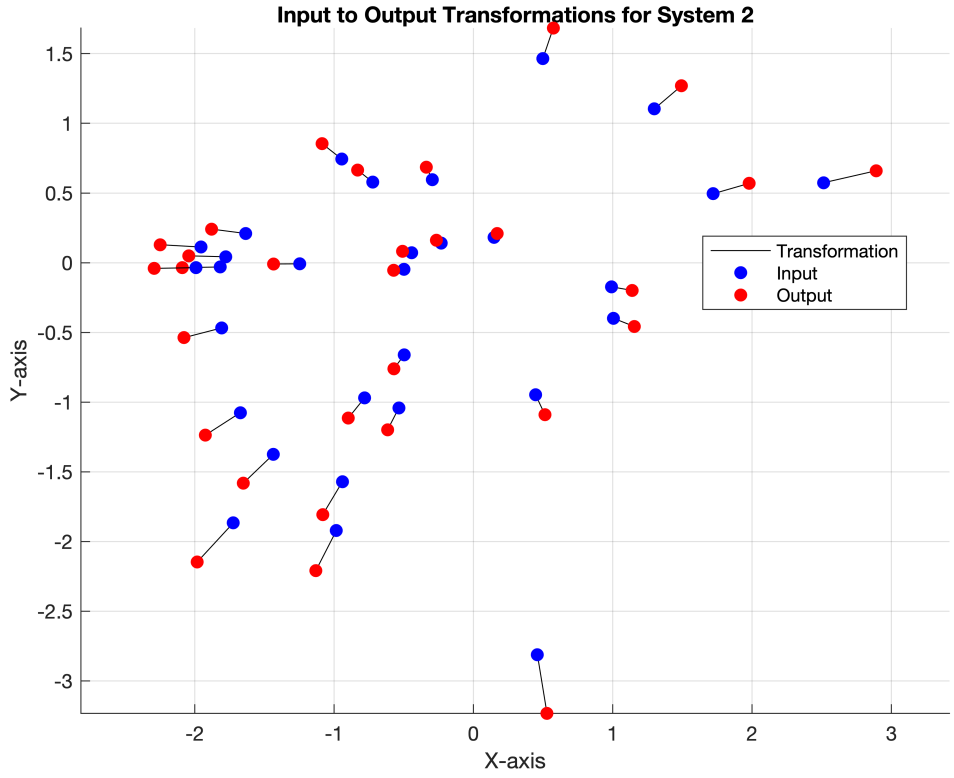
$\therefore$ The system can be linear.
since the input and output are scaled with the same size.
But M is not unique, there will be indefinite solutions, if they satisfy the dependent relationship between a, b, c and d.

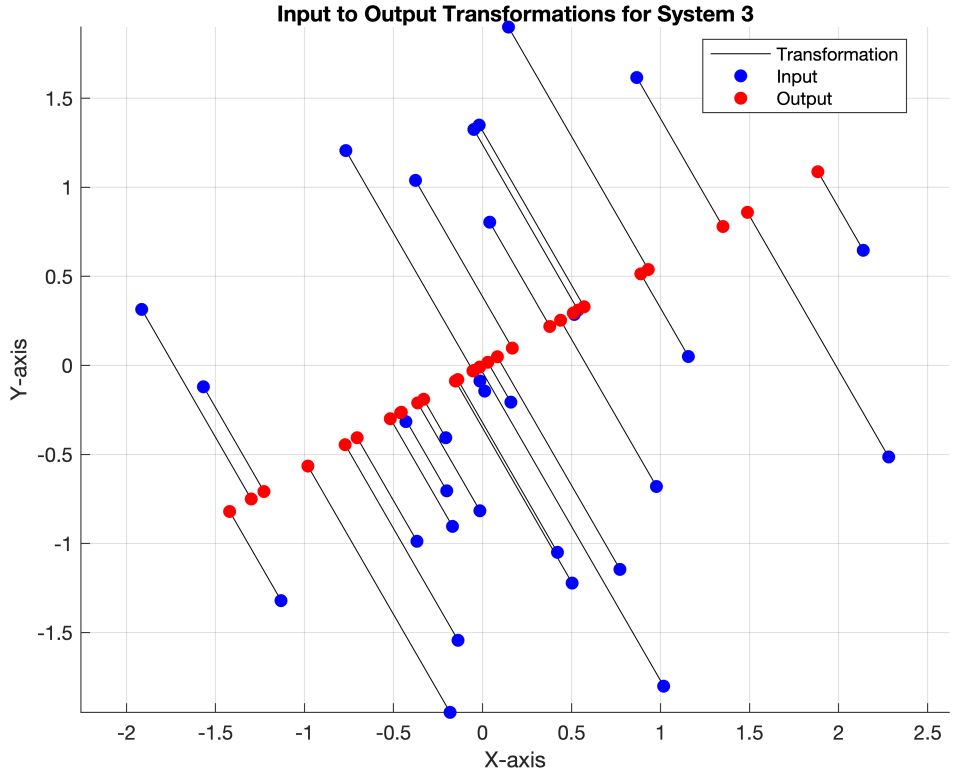
MathTools HW1

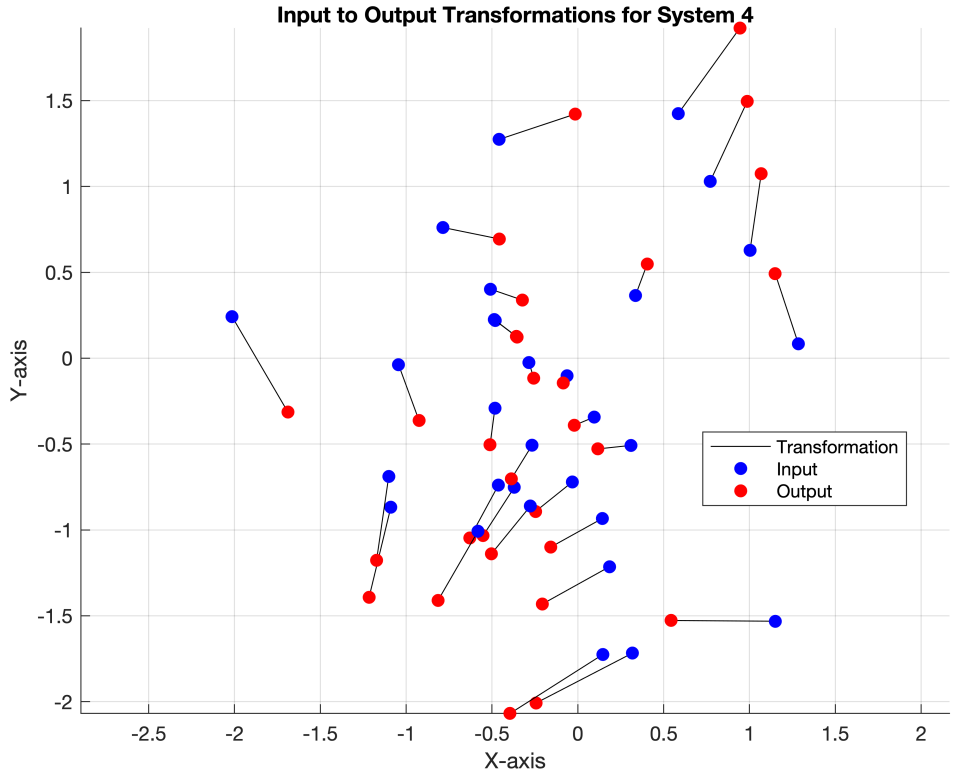
```
%% 2024-9-13
% Question 3
```

```
%a)

systems = {@sys1, @sys2, @sys3, @sys4};
sys_names = {'System 1', 'System 2', 'System 3', 'System 4'};

%Looping throught each system
for idx = 1:length(systems)
    system = systems{idx};
    system_name = sys_names{idx};

    %Generating 30 random 2D input vectors
    inputs = randn(30,2);

    %Empty holder for output
    outputs = zeros(30,2);

    % Compute outputs for each input
    for i = 1:30
        input_vector = inputs(i, :)';
        output_vector = system(input_vector);
        outputs(i, :) = output_vector';
    end

    figure; hold on;

    for i = 1:30
        input_point = inputs(i,:);
        output_point = outputs(i,:);

        plot([input_point(1), output_point(1)], [input_point(2), output_point(2)], 'k-
        plot(input_point(1), input_point(2), 'bo', 'MarkerFaceColor', 'b');
        plot(output_point(1), output_point(2), 'ro', 'MarkerFaceColor', 'r');
    end

    %Labeling
    xlabel('X-axis');
    ylabel('Y-axis');
    title(['Input to Output Transformations for ', system_name]);
    legend('Transformation', 'Input', 'Output', 'Location', 'Best');

    grid on;
    axis equal;
    hold off;

end
```


Input to Output Transformations for System 1
Transformation
Input
Output
X-axis
Y-axis
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2


Input to Output Transformations for System 2
Transformation
Input
Output
X-axis
Y-axis
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-2
-1
0
1
2
3






```
%b)
systems = {@sys1, @sys2, @sys3, @sys4};
```

```
sys_names = {'System 1', 'System 2', 'System 3', 'System 4'};
Ms = {};

% Loop over each system
for idc = 1:length(systems)
    system = systems{idc};
    system_name = sys_names{idc};

    fprintf('Analyzing %s\n', system_name);

    %Set some random simple impulses
    e1 = [1; 0];
    e2 = [0; 1];

    % Obatin each system's response to impulses
    m1 = system(e1);
    m2 = system(e2);

    % Construct M
    M = [m1, m2];
    Ms{idc} = M;
    fprintf('Matrix M for %s:\n', system_name);
    disp(M);

    %SVD of M
    [U, S, V] = svd(M);

    fprintf('Singular Values for %s:\n', system_name);
    disp(diag(S));

    fprintf('Matrix U for %s:\n', system_name);
    disp(U);
    fprintf('Matrix Vt for %s:\n', system_name);
    disp(V');

%     fprintf('Explanation for %s:\n', system_name);
%     if strcmp(system_name, 'System 1')
%         fprintf(['According to the SVD, the system (system 1) is first rotating inpu
%             'but perserving its magnitude, and then rotated an additional 75-degrees
%     elseif strcmp(system_name, 'System 2')
%         fprintf(['For this system (system 2), there is no rotation or reflection, bu
%             'equally scaled along both axes by 1.15. And there is no rotation or ref
%             'it remains in the same orientation.']);
%     elseif strcmp(system_name, 'System 3')
%         fprintf(['Firstly, the system 3 rotates input vectors by 150-degrees counter
%             'Finally, it rotates the vector by another 150-degree counterclockwise a
%     elseif strcmp(system_name, 'System 4')
%         fprintf(['The system 4 firstly rotated the vector by 60-degree and refelcts
%             'compresses it along another axis. Finally, it rotated results from s by
%     end %if

    fprintf('---\n\n');
end
```

```
Analyzing System 1
Matrix M for System 1:
    0.9659    0.2588
   -0.2588    0.9659
Singular Values for System 1:
    1.0000
    1.0000
Matrix U for System 1:
    0.2588   -0.9659
    0.9659    0.2588
Matrix Vt for System 1:
     0     1
    -1     0
---
Analyzing System 2
Matrix M for System 2:
    1.1500         0
         0    1.1500
Singular Values for System 2:
    1.1500
    1.1500
Matrix U for System 2:
     1     0
     0     1
Matrix Vt for System 2:
     1     0
     0     1
---
Analyzing System 3
Matrix M for System 3:
    0.7500    0.4330
    0.4330    0.2500
Singular Values for System 3:
    1.0000
    0.0000
Matrix U for System 3:
   -0.8660   -0.5000
   -0.5000    0.8660
Matrix Vt for System 3:
   -0.8660   -0.5000
    0.5000   -0.8660
---
Analyzing System 4
Matrix M for System 4:
    0.8750    0.3031
    0.3031    1.2250
Singular Values for System 4:
    1.4000
    0.7000
Matrix U for System 4:
    0.5000    0.8660
    0.8660   -0.5000
Matrix Vt for System 4:
    0.5000    0.8660
    0.8660   -0.5000
---
```

Part B Explanations:

System 1: According to the SVD, the system (system 1) is first rotating input vectors by 90-degrees counterclockwise, but perserving its magnitude, and then rotated an additional 75-degrees counterclockwise (after no scaling).

System2: For this system (system 2), there is no rotation or reflection, but the input vector has been equally scaled along both axes by 1.15. And there is no rotation or reflection after scaling, it remains in the same orientation.

System3: Firstly, the system 3 rotates input vectors by 150-degrees counterclockwise, then it projects the rotated vector x onto the first axis. Finally, it rotates the vector by another 150-degree counterclockwise and the reflected.

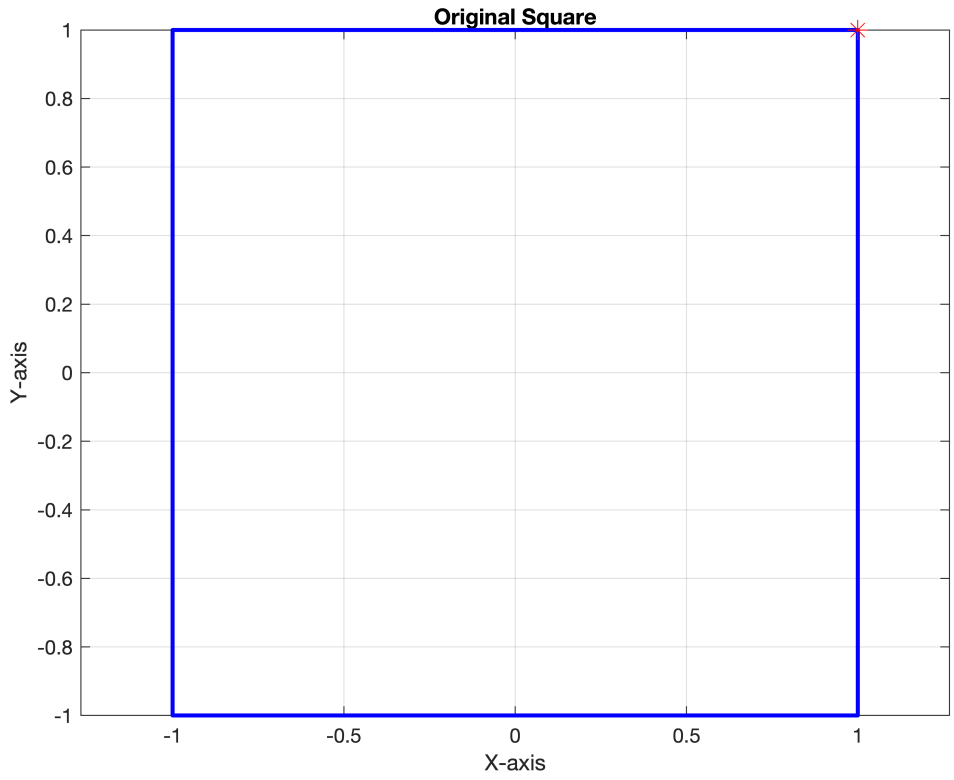
System4: The system 4 firstly rotated the vector by 60-degree and refelcts it across a line. Then, it stretched the vector along one axis and compresses it along another axis. Finally, it rotated results from s by another 60-degree and reflected it again.

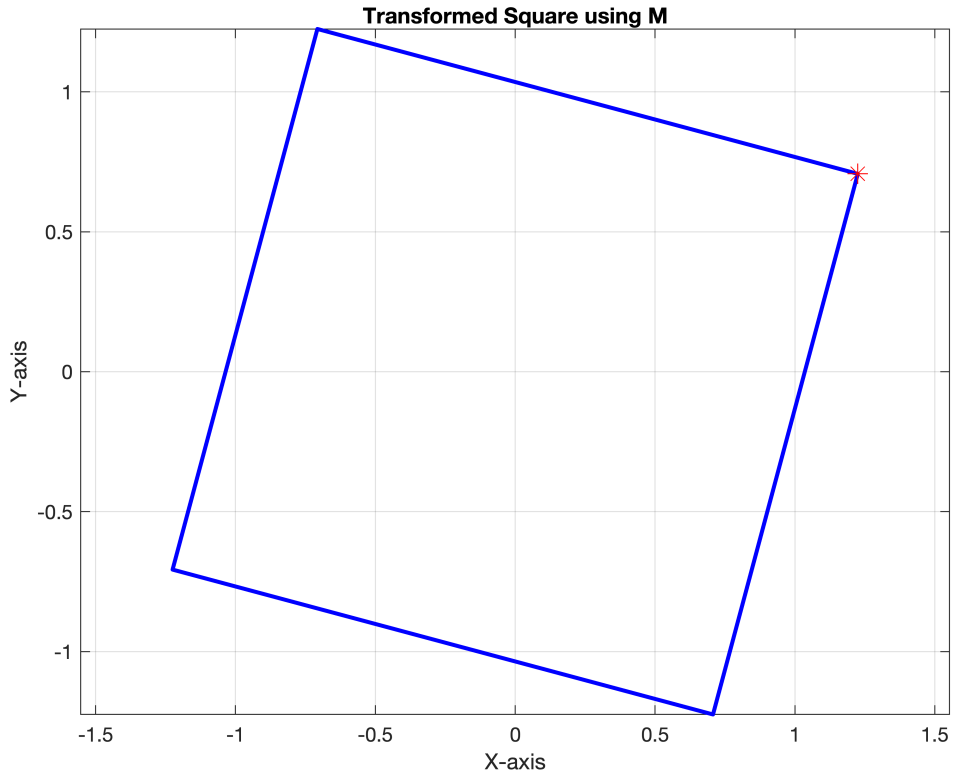
```
%c)

for idz = 1:length(systems)
    system = systems{idz};
    system_name = sys_names{idz};

    fprintf('Plotting square passing through %s\n', system_name);

% Generate the data matrix P
P = [  0,  1,  1,  1,  0, -1, -1, -1;
       1,  1,  0, -1, -1, -1,  0,  1 ];

P_closed = [P, P(:,1)];

% Plot the original square
figure;
plot(P_closed(1, :), P_closed(2, :), 'b-', 'LineWidth', 2);
hold on;
plot(P(1, 2), P(2, 2), 'r*', 'MarkerSize', 10);
xlabel('X-axis');
ylabel('Y-axis');
title('Original Square');
axis equal;
grid on;
hold off;

M = Ms{idz};

% Apply M to P
P_transformed = M * P;

% Plot the transformed square
P_transformed_closed = [P_transformed, P_transformed(:,1)];
figure;
plot(P_transformed_closed(1, :), P_transformed_closed(2, :), 'b-', 'LineWidth', 2);
```

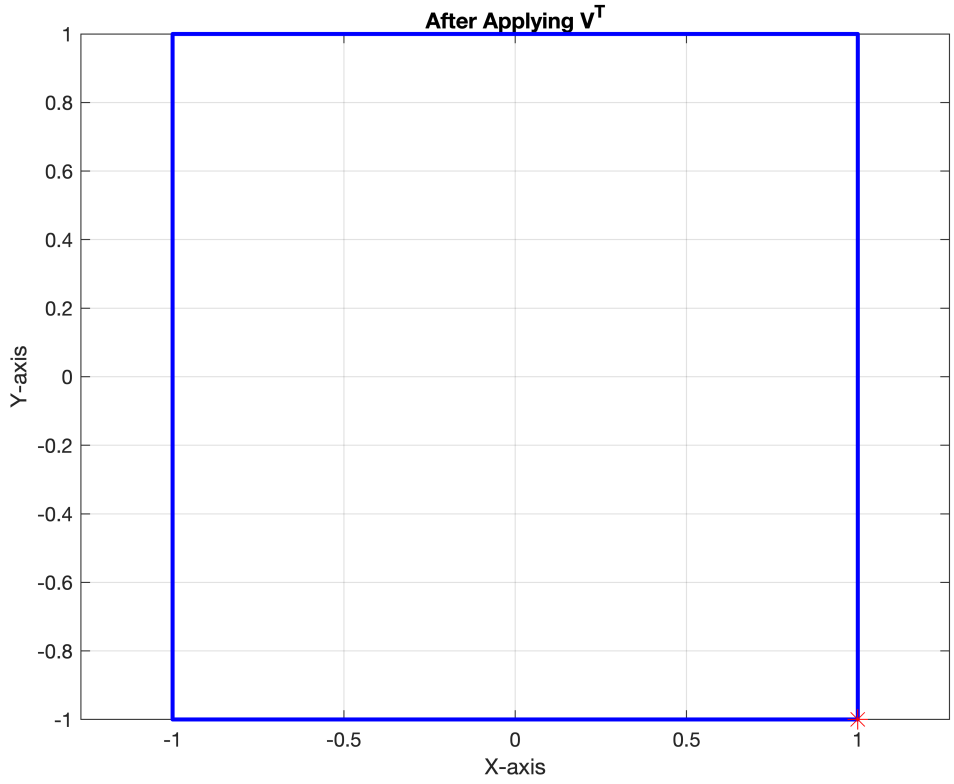
```
hold on;
plot(P_transformed(1, 2), P_transformed(2, 2), 'r*', 'MarkerSize', 10);
xlabel('X-axis');
ylabel('Y-axis');
title('Transformed Square using M');
axis equal;
grid on;
hold off;

% Compute the SVD of M
[U, S, V] = svd(M);

% Apply the transformations step by step
P_step1 = V' * P;
P_step2 = S * P_step1;
P_step3 = U * P_step2;

% Verify that P_step3 equals P_transformed
assert(norm(P_step3 - P_transformed, 'fro') < 1e-10, 'Mismatch in transformations');

% Plot after applying V^T
P_step1_closed = [P_step1, P_step1(:,1)];
figure;
plot(P_step1_closed(1, :), P_step1_closed(2, :), 'b-', 'LineWidth', 2);
hold on;
plot(P_step1(1, 2), P_step1(2, 2), 'r*', 'MarkerSize', 10);
xlabel('X-axis');
ylabel('Y-axis');
title('After Applying V^T');
axis equal;
grid on;
hold off;

% Plot after applying S
P_step2_closed = [P_step2, P_step2(:,1)];
figure;
plot(P_step2_closed(1, :), P_step2_closed(2, :), 'b-', 'LineWidth', 2);
hold on;
plot(P_step2(1, 2), P_step2(2, 2), 'r*', 'MarkerSize', 10);
xlabel('X-axis');
ylabel('Y-axis');
title('After Applying S');
axis equal;
grid on;
hold off;

% Plot after applying U
P_step3_closed = [P_step3, P_step3(:,1)];
figure;
plot(P_step3_closed(1, :), P_step3_closed(2, :), 'b-', 'LineWidth', 2);
hold on;
plot(P_step3(1, 2), P_step3(2, 2), 'r*', 'MarkerSize', 10);
xlabel('X-axis');
ylabel('Y-axis');
```

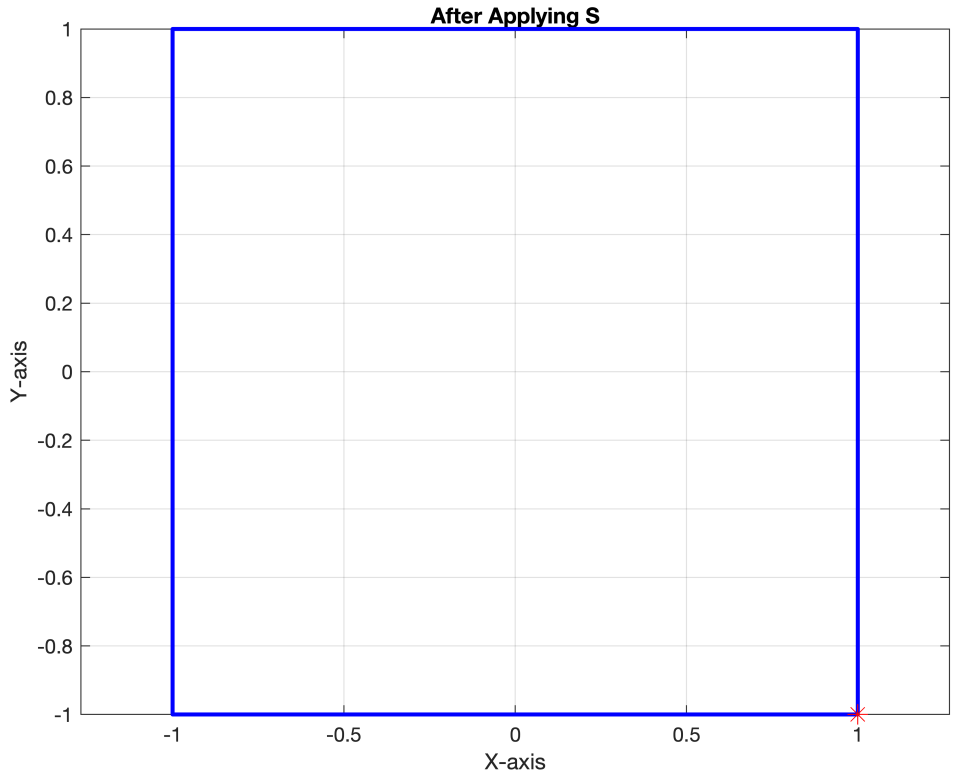
```
title('After Applying U (Final Transformation)');
axis equal;
grid on;
hold off;
end
```

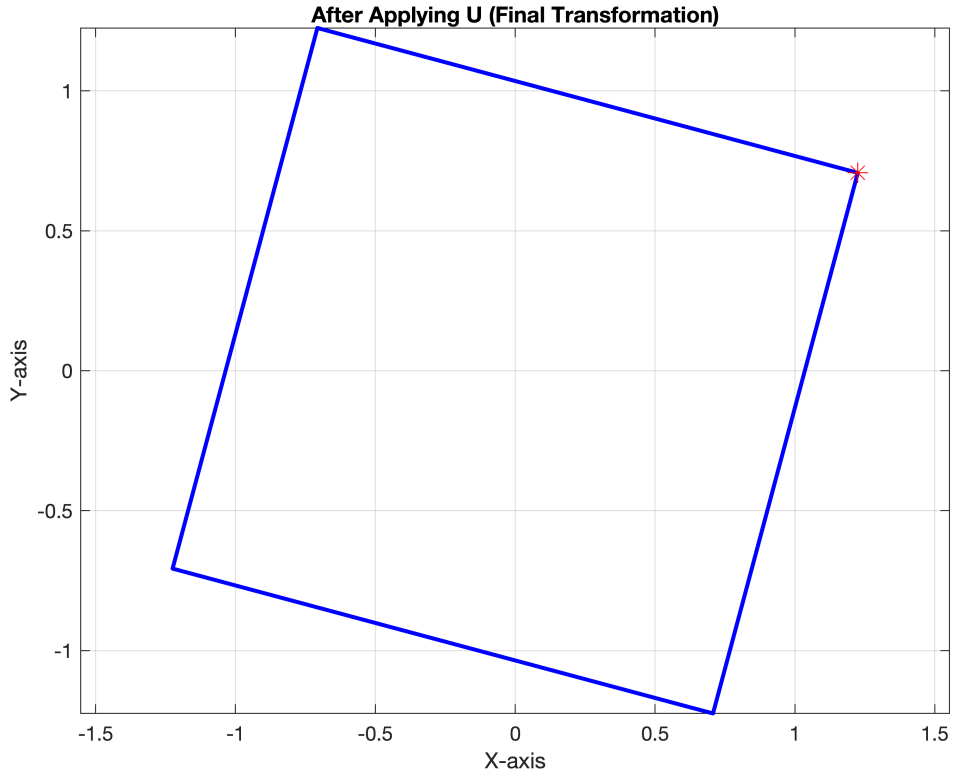
```
Plotting square passing through System 1
```



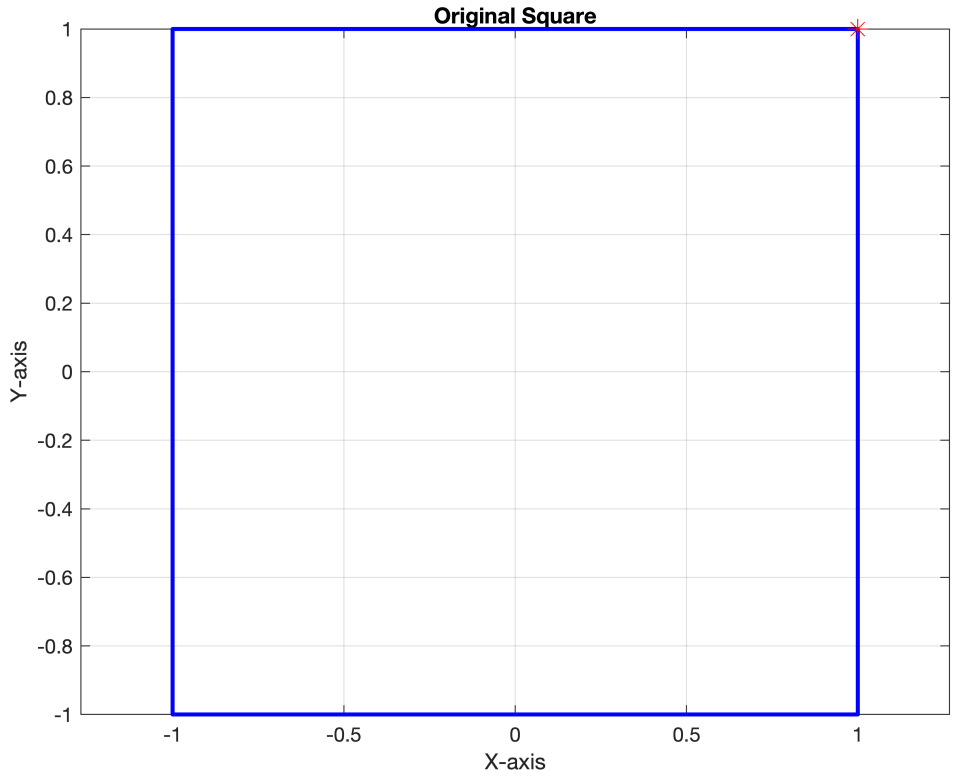

Transformed Square using M
Y-axis
X-axis
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5

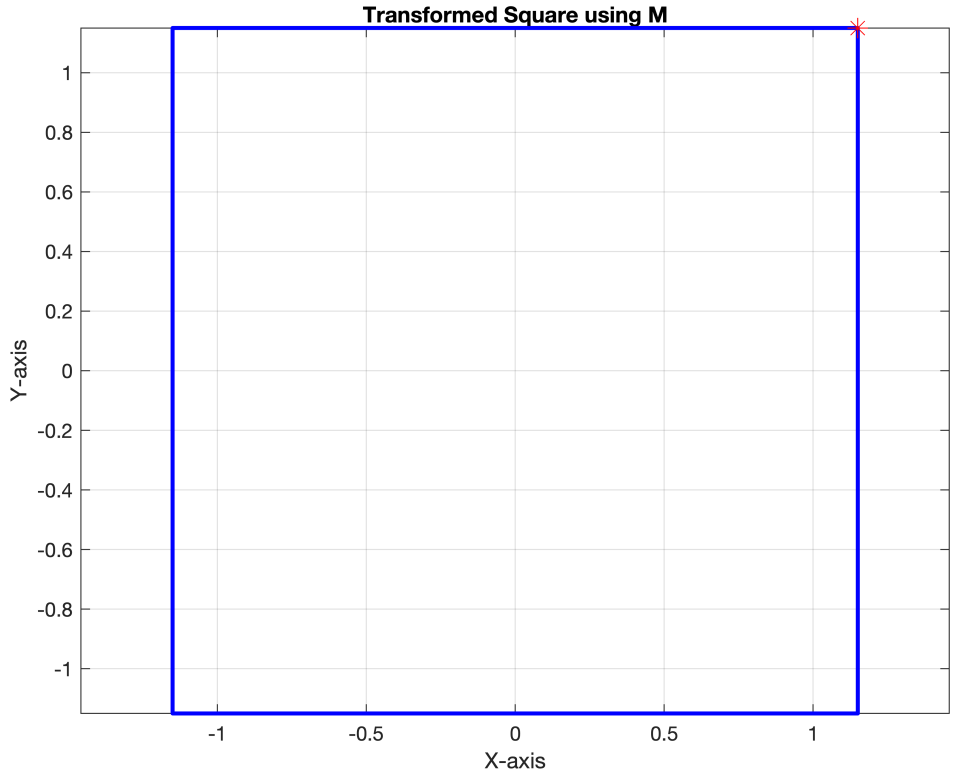

After Applying V^T
Y-axis
X-axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

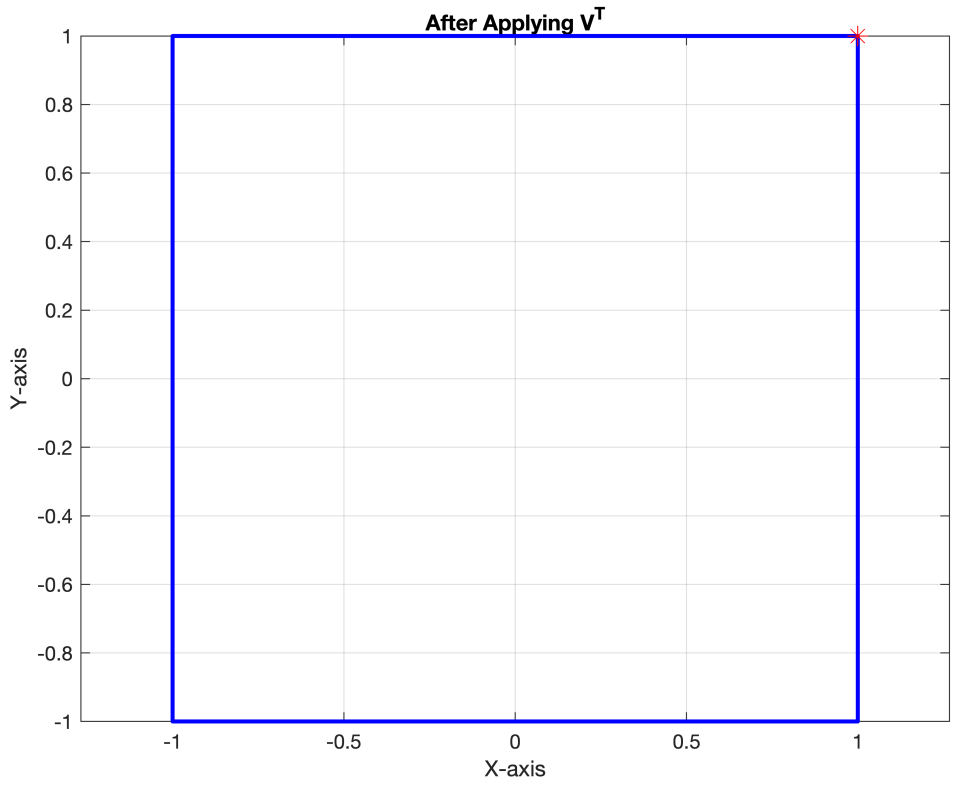


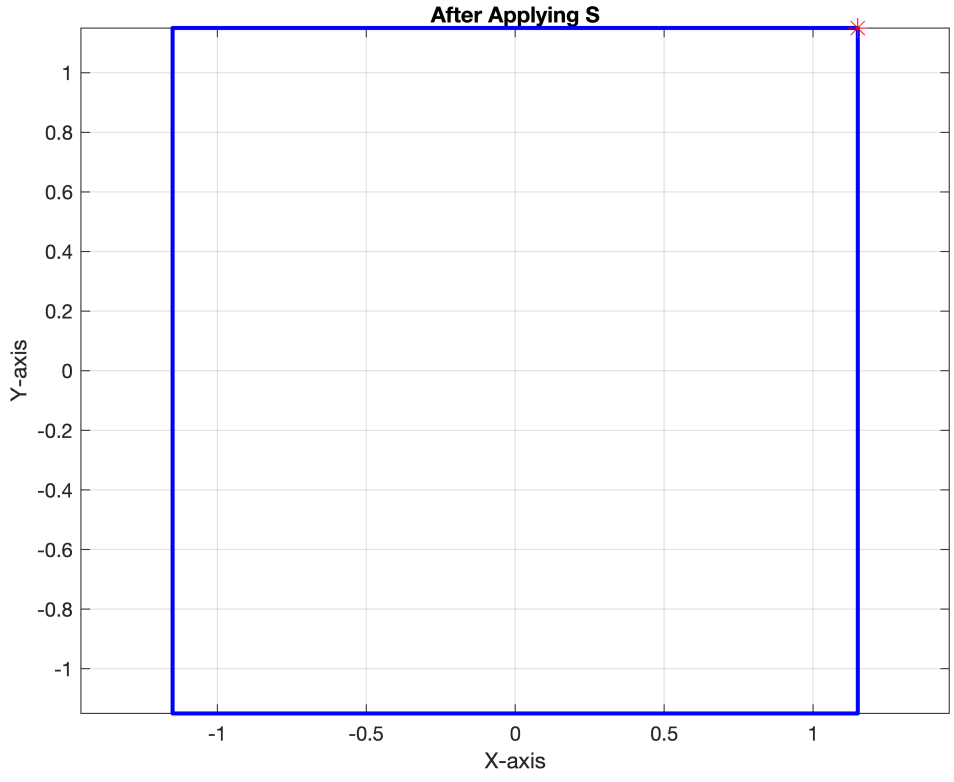


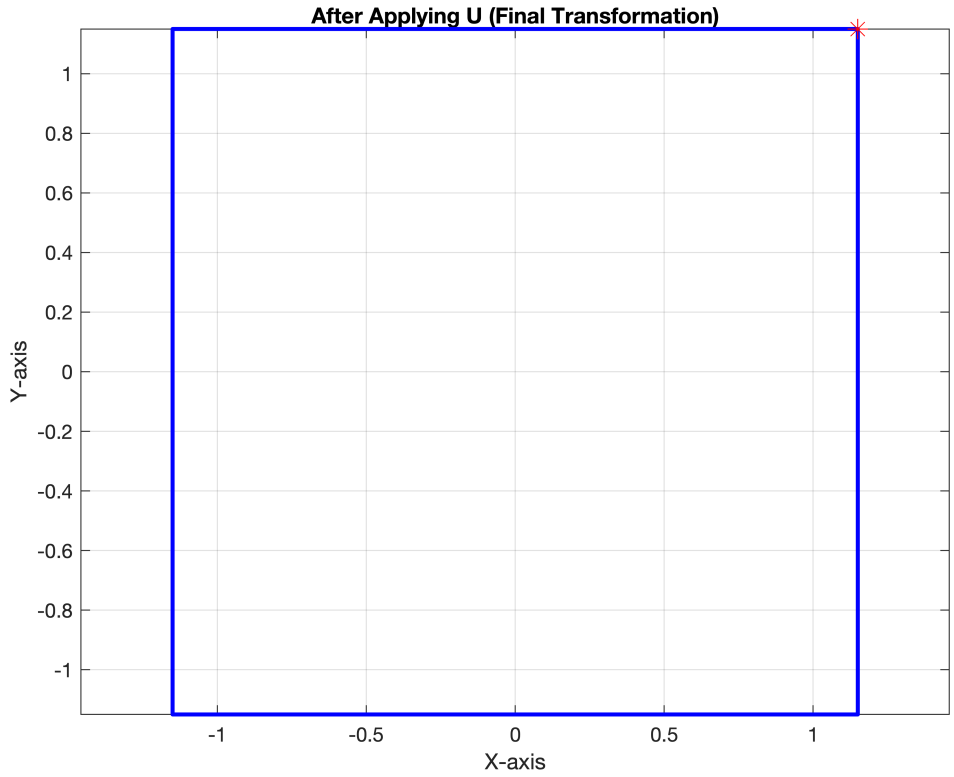

```
Plotting square passing through System 2
```


Original Square
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Y-axis
-1
-0.5
0
0.5
1
X-axis

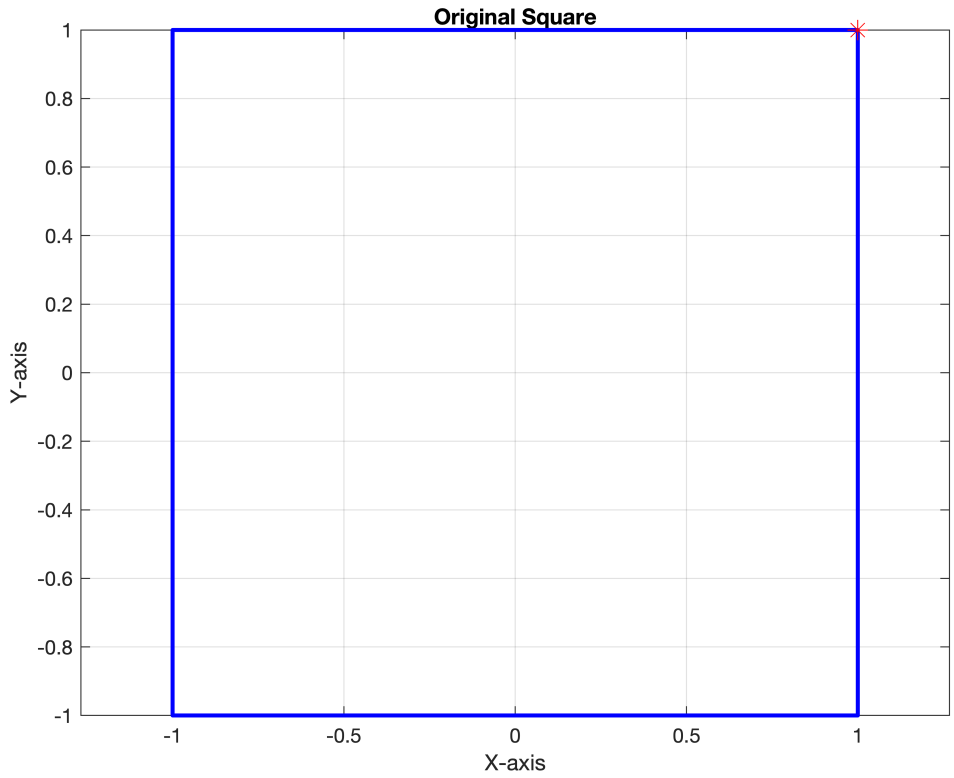

Transformed Square using M
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Y-axis
-1
-0.5
0
0.5
1
X-axis

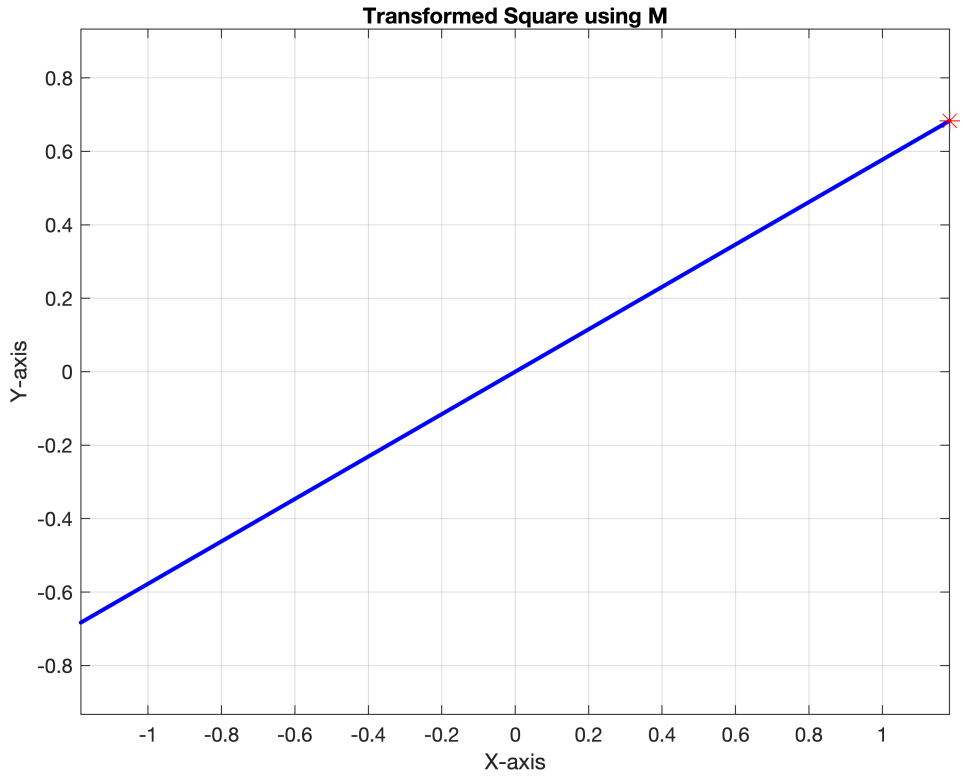

After Applying V^T
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Y-axis
-1
-0.5
0
0.5
1
X-axis

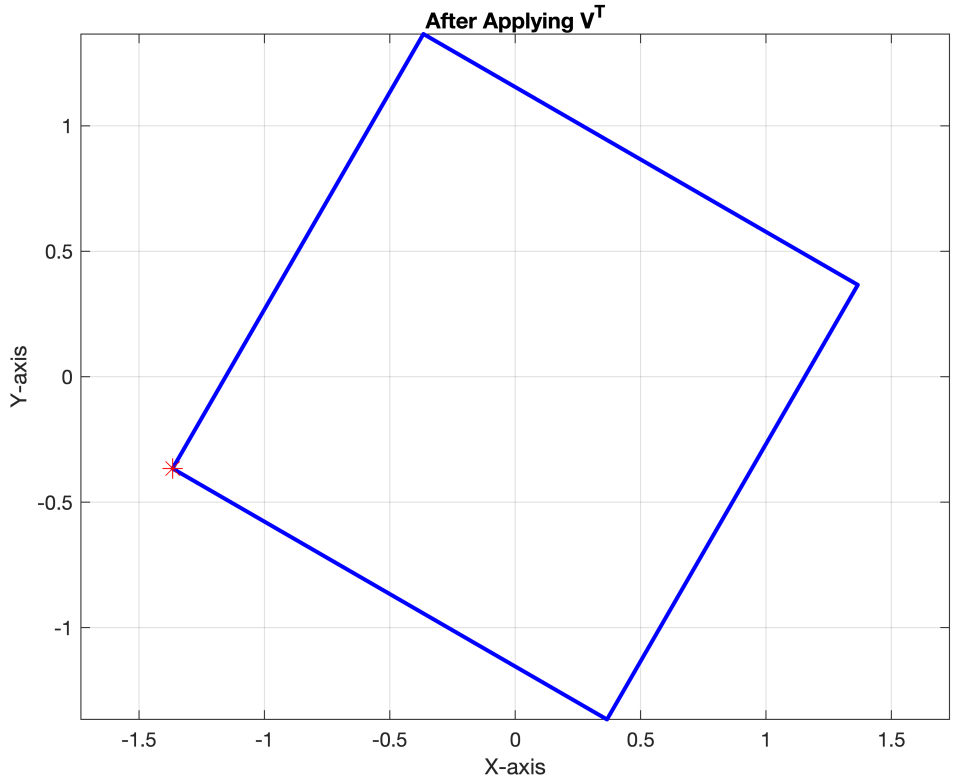

After Applying S
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Y-axis
-1
-0.5
0
0.5
1
X-axis

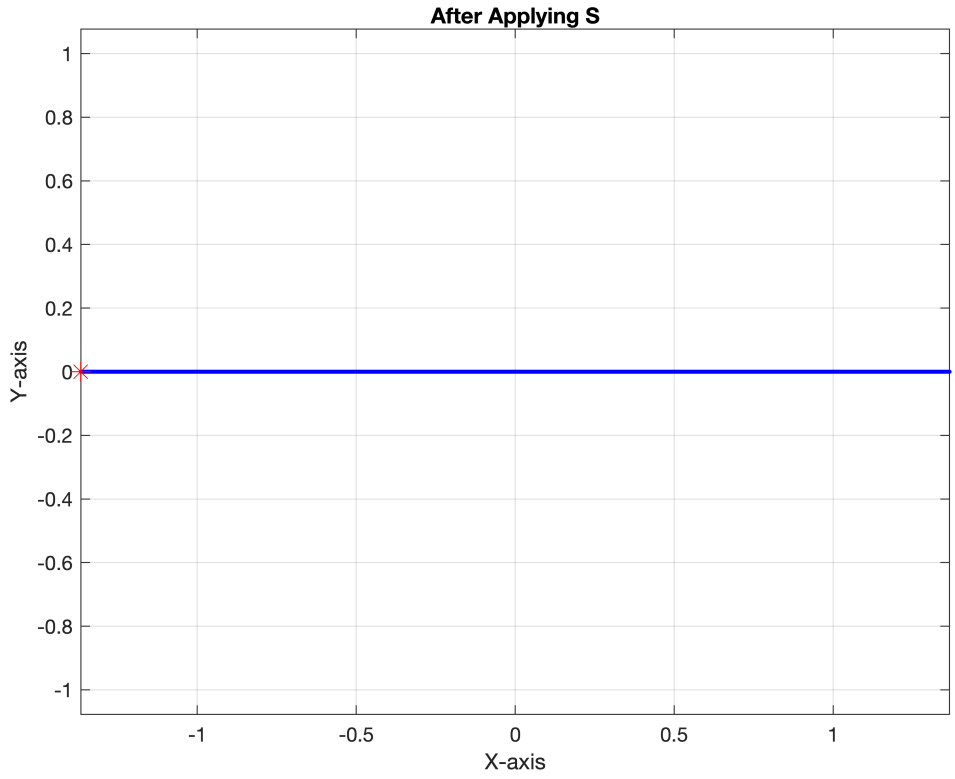


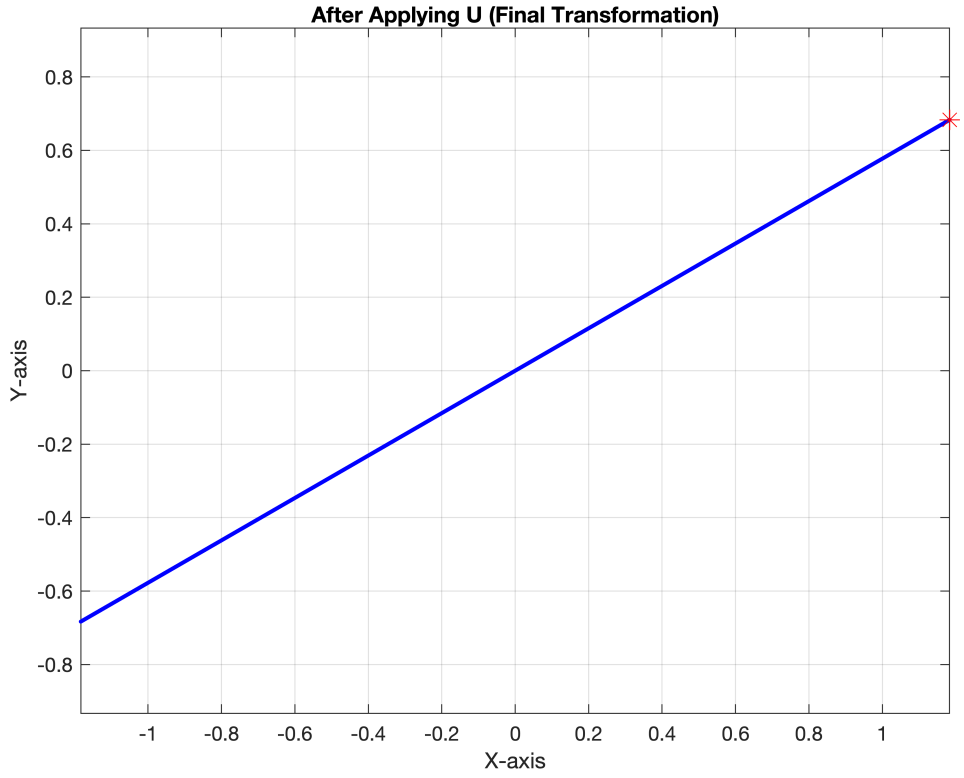

```
Plotting square passing through System 3
```



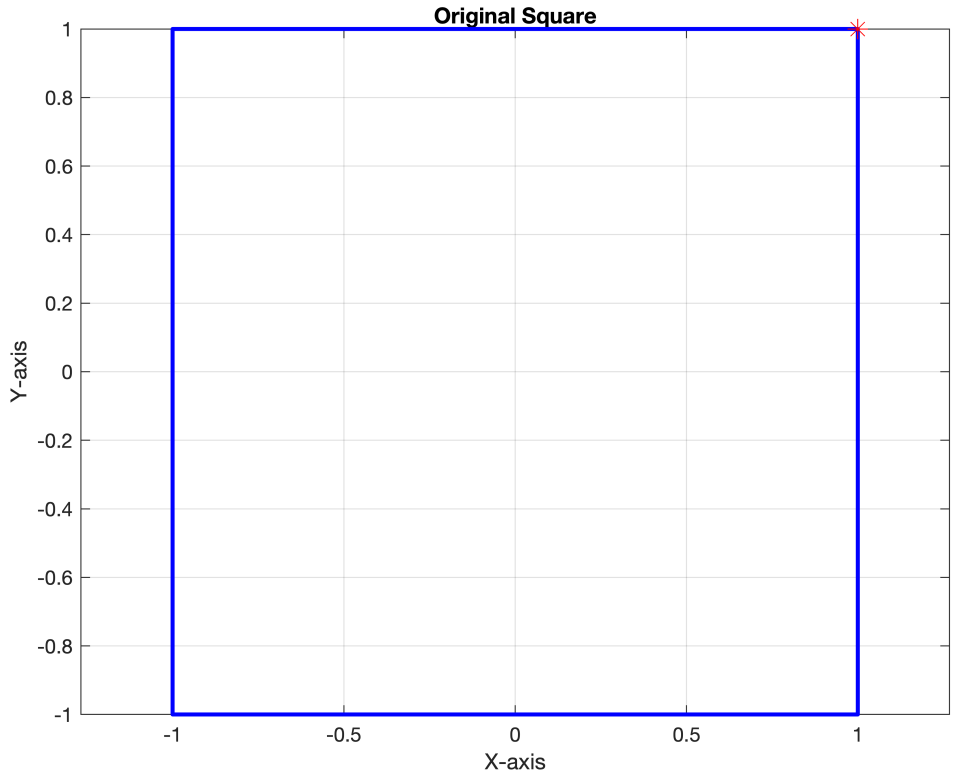

Transformed Square using M
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
Y-axis
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
X-axis

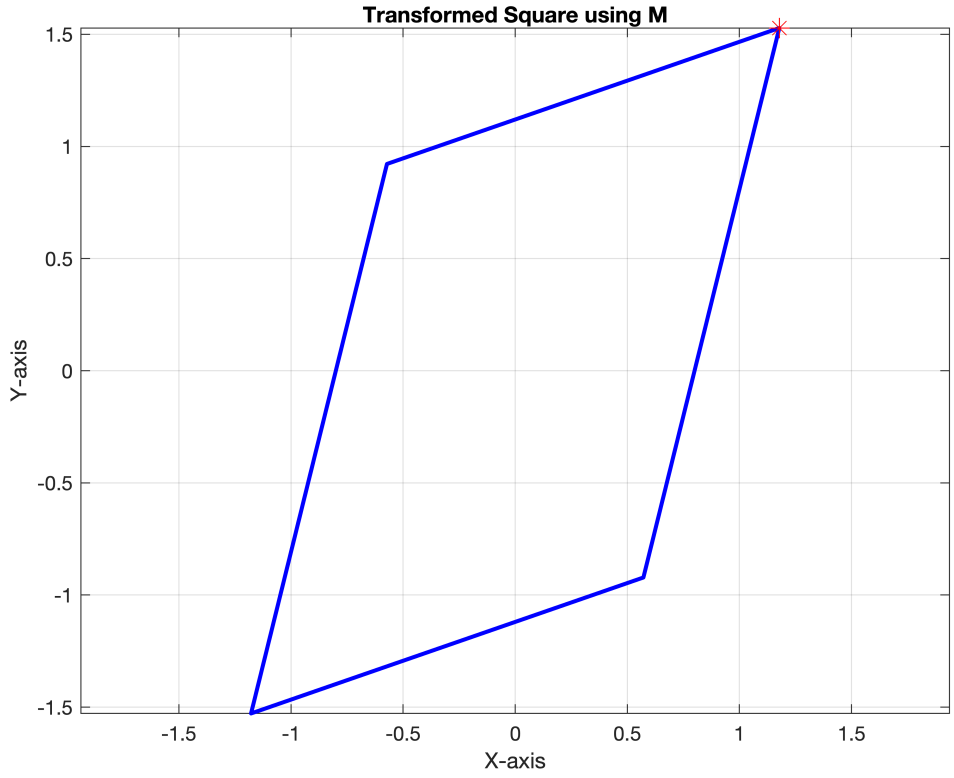

After Applying V^T
1
0.5
0
-0.5
-1
Y-axis
-1.5
-1
-0.5
0
0.5
1
1.5
X-axis

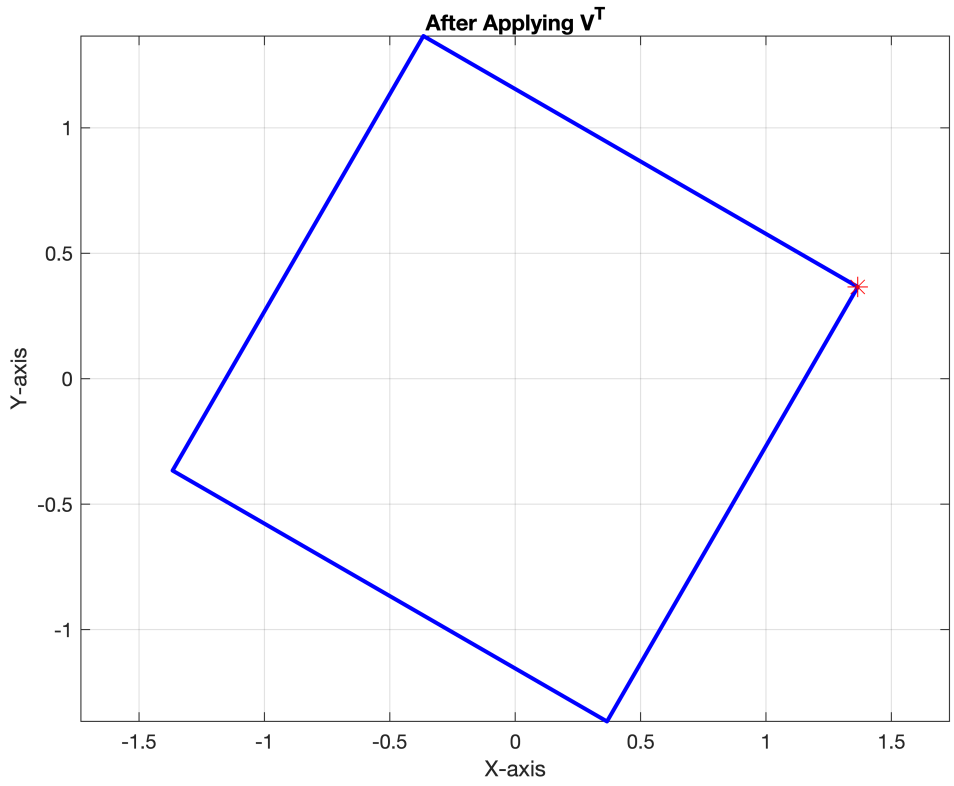


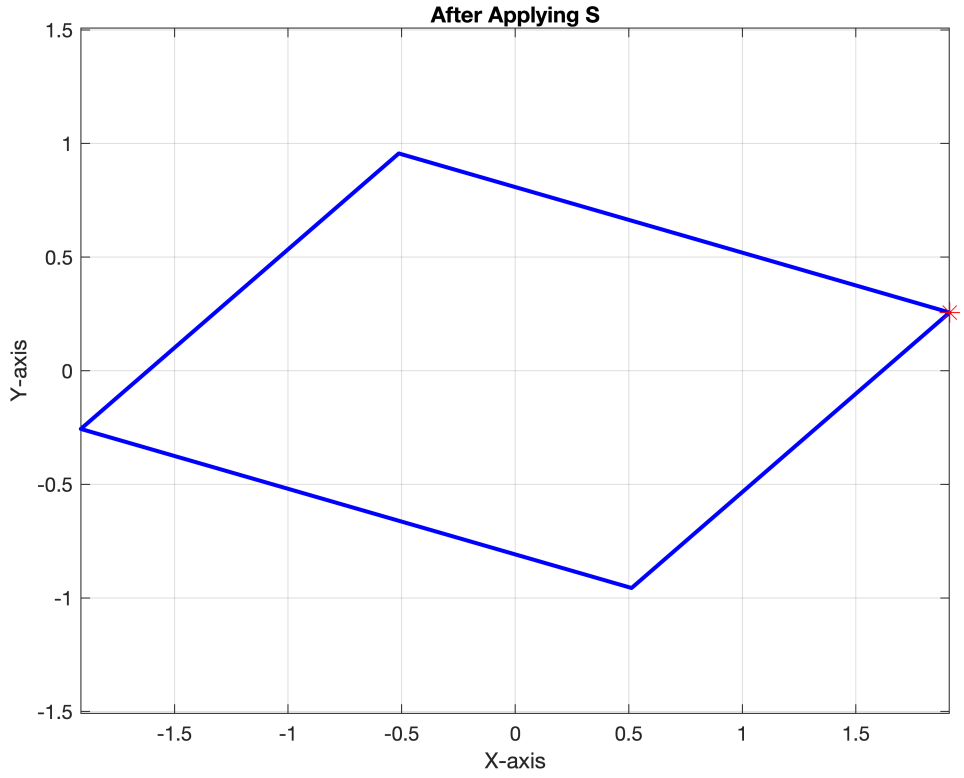


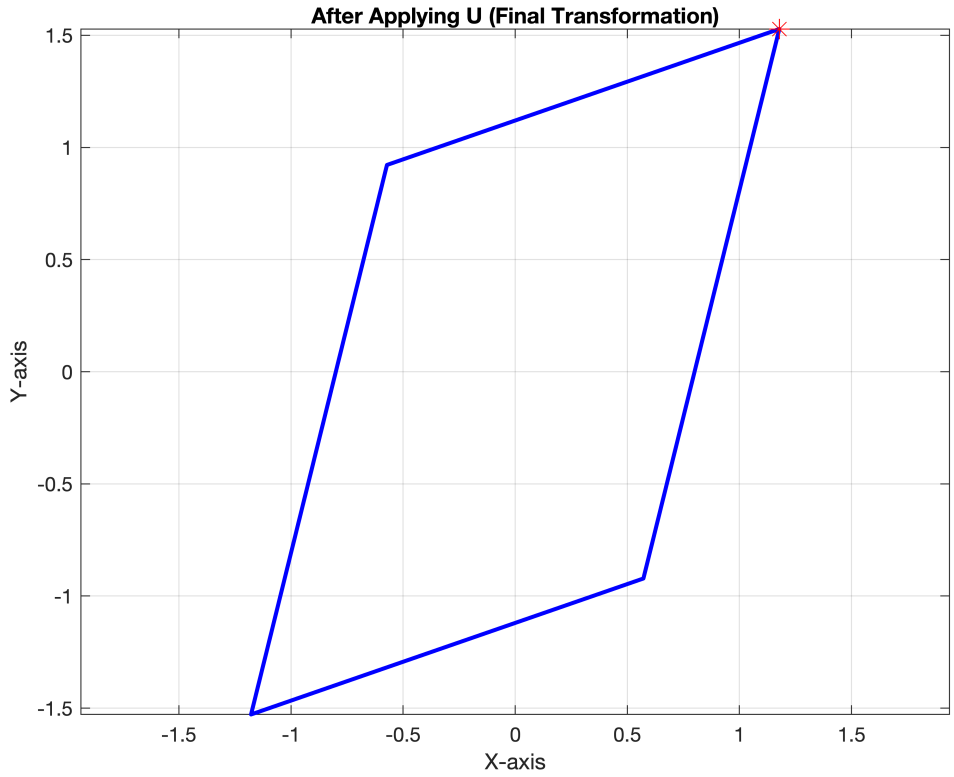

```
Plotting square passing through System 4
```


Original Square
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Y-axis
-1
-0.5
0
0.5
1
X-axis


Transformed Square using M
1.5
1
0.5
0
-0.5
-1
-1.5
Y-axis
-1.5
-1
-0.5
0
0.5
1
1.5
X-axis


After Applying V^T
Y-axis
X-axis
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5


After Applying S
Y-axis
X-axis
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5




```
disp('Overall, having the square as a visualization helped better understanding how th
```

```
Overall, having the square as a visualization helped better understanding how those systems are changing t
```

Overall, having the square as a visualization helped better understanding how those systems are changing the inputs.

MathTools HW1

```
%% Question 4
% 2024-9-13
```

a) If the retinal neuron is only reaction at certain location, then in general the expression of that neuron can ba [0, .., v1, .., v2, .., v3, .., v4, .., v5]. In short, it would be as stated in question, v = [v1, v2, v3, v4, v5].

Given that the vector is v = [v1, v2, v3, v4, v5] and the weighted output is r = v1 + 2.3v2 + 1.5v3 + v4 + 6v5, we can obtain that the system, let's say w, would be w = [1, 2.3, 1.5, 1, 1,6].

Therefore, the system is linear.

b) Since it is asked to get the maximum output with unit-length input, then ||v|| = 1. The question is asking r = WV to be maximum, thus r to be maximum.

Understanding that WV = ||w||.||v||.cos(theta). Then, cos(theta) need to be largest, where when theta is 0, cos(theta) would be 1. In geometry representation, the vector v and w would need to be in same direciton. Following calculation please see attached hand written section.

In conclusion, thinking geometrically and in 2D, the dot product will reach its maximum when the vectors lie in the same line and in positive values.

c) To reach the minimal output, the vector needed would be any unit vector that has 0 input anywhere else, but 1 at one of the smallest weighted term. For example, in w, 1st and 4th term is weighted 1, so the input can be:

v(min) = [1, 0, 0, 0, 0]; or, v(min) = [0, 0, 0, 1, 0].

For a vector to be physically realizable, it means that it can actually occur in reality, thus the vector can not be all-zeros.

Q4 b) $r = \|w\| \cdot \|v\| \cdot \cos\theta$ to be largest

$\because \cos\theta = 1, \theta = 0 \therefore \vec{v} = \frac{w}{\|w\|}$

$\|w\| = \sqrt{1^2 + 2.3^2 + 1.5^2 + 1^2 + 6^2} = \sqrt{45.54}$

≈ 6.75

$\because \vec{v}$ need to be unit length

$\therefore \|v\| = 1$

Thus, $r = w^T v = w^T \left(\frac{w}{\|w\|}\right)$

$= \frac{w^T w}{\|w\|} = \frac{\|w\|^2}{\|w\|} = \|w\|$

$\therefore r = 6.75$

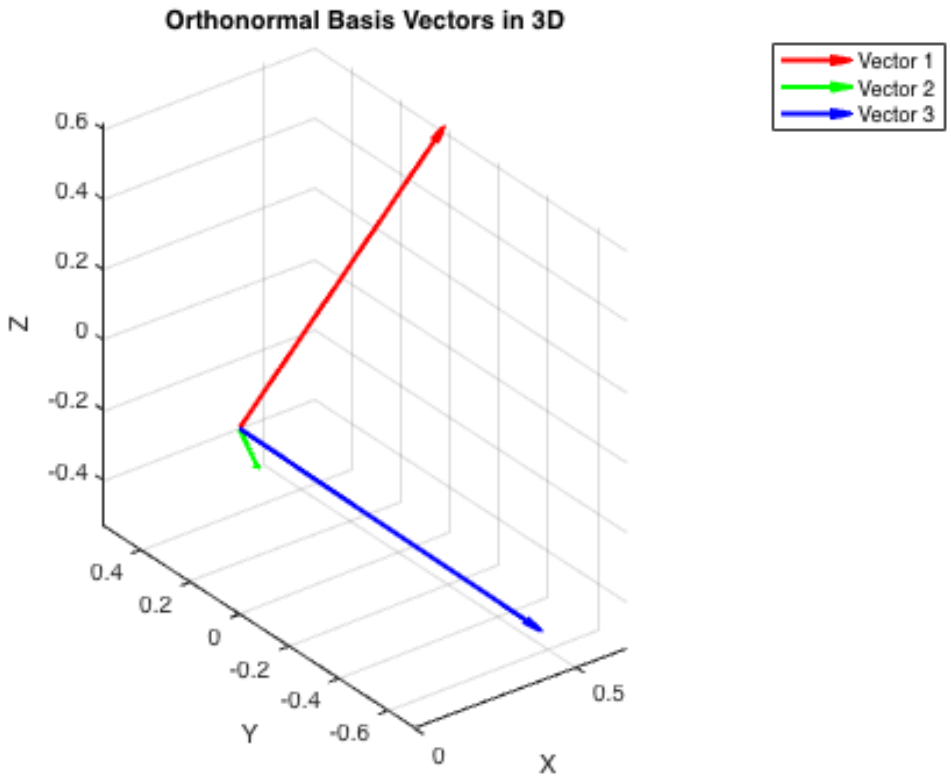
$$\overrightarrow{V_{max}} = \frac{1}{6.75} \begin{bmatrix} 1 \\ 2.3 \\ 1.5 \\ 1 \\ 6 \end{bmatrix}$$

MathTools HW1

```
%% Question 5
%2024-9-20
```

Testing for n=3

```
% Generate the orthonormal basis
Q4 = gramschmidt(3);

% Plot the basis vectors
figure;
quiver3(0, 0, 0, Q4(1, 1), Q4(2, 1), Q4(3, 1), 'r', 'LineWidth', 2);
hold on;
quiver3(0, 0, 0, Q4(1, 2), Q4(2, 2), Q4(3, 2), 'g', 'LineWidth', 2);
quiver3(0, 0, 0, Q4(1, 3), Q4(2, 3), Q4(3, 3), 'b', 'LineWidth', 2);
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Orthonormal Basis Vectors in 3D');
legend('Vector 1', 'Vector 2', 'Vector 3');
grid on;
axis equal;
rotate3d on;
hold off;
```



Verifying Orthonormality for n=1000

Reasoning: if the matrix is an orthonormal matrix, then Q4' * Q4 should be an identity matrix if the columns are orthonormal. If the rows are orthonormal, then Q4*Q4' should also be an identity matrix.

```
n = 1000;
Q4b = gramschmidt(n);

% Check orthonormality of columns
orthogonality_columns = norm(Q4b' * Q4b - eye(n));

% Check orthonormality of rows
orthogonality_rows = norm(Q4b * Q4b' - eye(n));

disp(['Orthogonality of columns: ', num2str(orthogonality_columns)]);
```

```
Orthogonality of columns: 4.2513e-13
```

```
disp(['Orthogonality of rows: ', num2str(orthogonality_rows)]);
```

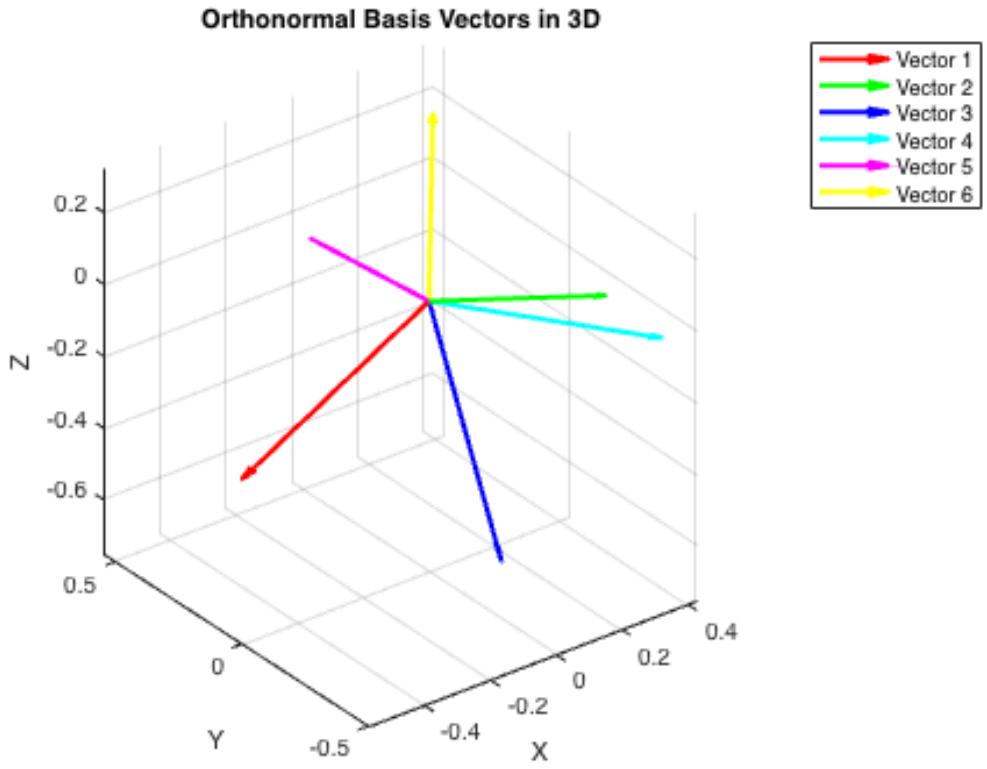
```
Orthogonality of rows: 4.2514e-13
```

```
```

To sum up, since values of *orthogonality_columns* and *orthogonality_rows* are close enought to zero, so we can say that both Q4' * Q4 and Q4 * Q4' are two identity matrix. Thus, the matrix is an orthonormal matrix.

Testing recursive function

```
Q4_recursive = gramschmidt_recursive_main(6);
figure;
quiver3(0, 0, 0, Q4_recursive(1, 1), Q4_recursive(2, 1), Q4_recursive(3, 1), 'r', 'Lin
hold on;
quiver3(0, 0, 0, Q4_recursive(1, 2), Q4_recursive(2, 2), Q4_recursive(3, 2), 'g', 'Lin
quiver3(0, 0, 0, Q4_recursive(1, 3), Q4_recursive(2, 3), Q4_recursive(3, 3), 'b', 'Lin
quiver3(0, 0, 0, Q4_recursive(1, 4), Q4_recursive(2, 4), Q4_recursive(3, 4), 'c', 'Lin
quiver3(0, 0, 0, Q4_recursive(1, 5), Q4_recursive(2, 5), Q4_recursive(3, 5), 'm', 'Lin
quiver3(0, 0, 0, Q4_recursive(1, 6), Q4_recursive(2, 6), Q4_recursive(3, 6), 'y', 'Lin
xlabel('X');
ylabel('Y');
zlabel('Z');
title('Orthonormal Basis Vectors in 3D');
legend('Vector 1', 'Vector 2', 'Vector 3', 'Vector 4','Vector 5','Vector 6');
grid on;
axis equal;
rotate3d on;
hold off;
```




```
function Q4 = gramschmidt(n)
Q4 = zeros(n,n);

for i = 1:n

    %Generating a random unit vector
    unit_vector = randn(n,1);

    for j = 1:i-1
        projection = (Q4(:,j)'*unit_vector)*Q4(:,j);
        unit_vector = unit_vector - projection;
    end

    %Normalizing teh unit vector
    unit_vector_norm = sqrt(sum(unit_vector.^2)); %getting the norm first
    Q4(:,i) = unit_vector/unit_vector_norm;

end %for loop
end %function

%% Making a recursive
function Q4_recursive = gramschmidt_recursive_main(n)
%a function that's calling itself
Q4_recursive = gramschmidt_recursive(n, [], 0);
end

%the recursive part
```

```
function Q4_recursive = gramschmidt_recursive(n, Q4_previous, z)

if z == n
   Q4_recursive = Q4_previous;
else
    %generate a random vector
    v_recursive = randn(n,1);

    for u = 1:z
        projection_recursive = (Q4_previous(:, u)' * v_recursive) * Q4_previous(:, u);
        v_recursive = v_recursive - projection_recursive;
    end %for

    %normalizing the vector
    v_recursive_norm = sqrt(sum(v_recursive.^2)); %getting the norm first
    v_recursive = v_recursive/v_recursive_norm;

    %adding a new column
    Q4_previous = [Q4_previous, v_recursive];

    %calling itself again
    Q4_recursive = gramschmidt_recursive(n, Q4_previous, z+1);

end %ifelse
end %function
```

MathTools HW1

```
%% Question 6
%2024-9-25
```

a)

The null space is also called the kernal of a matrix. It refers to the space where all vectors in the inputs space will be mapping to zero in the output space.

The range space refers to the column space of a matrix, the set of all possible linear combinations of its column vectors. No matter what you put in input. The output is always lying in the range space (the possible space that the output vector can lies in).

b)

If the creature has a non-zero null space, that means when it received non-zero pressure inputs, it will result in zero neuronal responses. More specifically, it means that the creature has limited tactile snesory scope, meaning that it's unable to perceive certain pressure stimuli.

```
%c)
S = load('mtxExamples2024.mat');

for idx = 1:5
    fileName = sprintf('mtx%d',idx);
    M = S.(fileName);

    fprintf('--- Processing Matrix %s ---\n', fileName);

    %Goal 1: determine if there's a nullspace
    [U, S_matrix, V] = svd(M); singular_values = diag(S_matrix);

    boundary = 10^-3;

    %if the diagnol of s matrix is zero
    null_columns = find(all(S_matrix < boundary, 1));

    if size(null_columns,2)>0
        fprintf('There is a nullspace of size %d\n', size(null_columns, 2))
        xCord =  V(:, null_columns) * randn(size(null_columns, 2), 1);
        fprintf('The vector in the null space is:')
        disp(xCord);
        yCord = M*xCord;
        fprintf('Sanity check (should be near zero):')
        disp(yCord);
    else
        fprintf('No nullspace exists for this matrix')

    end %if
```

```
    range_rows = find(singular_values > boundary, 2);

    range_indices = find(singular_values >= boundary);
    S_pinv = zeros(size(S_matrix'));

        for i = 1:length(range_indices)
        idk = range_indices(i);
        S_pinv(idk, idk) = 1 / singular_values(idk);
        end %for


    y_range = U(:, range_rows) * randn(size(range_rows, 1), 1);
    %x_range = pinv(M) * y_range; %custom function for pseudoinverse

    M_pinv = V * S_pinv * U';
    x_range = M_pinv * y_range;
    fprintf('The vector in the range space is:')
    y_range_recomputed = M * x_range;
    disp(y_range_recomputed);

    fprintf('Sanity check (should be same as y_range_recomputed):')
    disp(y_range);

end %for
```

```
--- Processing Matrix mtx1 ---
No nullspace exists for this matrix
The vector in the range space is:
    0.2594
   -0.0598
   -0.4061
Sanity check (should be same as y_range_recomputed):
    0.2594
   -0.0598
   -0.4061
--- Processing Matrix mtx2 ---
There is a nullspace of size 2
The vector in the null space is:
   -0.3858
   -0.1679
   -0.1337
Sanity check (should be near zero):
   1.0e-15 *

   -0.0069
   -0.1110
         0
The vector in the range space is:
   -0.1384
   -1.8457
    0.8183
Sanity check (should be same as y_range_recomputed):
   -0.1384
   -1.8457
    0.8183
--- Processing Matrix mtx3 ---
There is a nullspace of size 1
The vector in the null space is:
```

```
   -1.6327
   -1.5201
    0.3517
Sanity check (should be near zero):
   1.0e-14 *

   -0.0278
    0.1082
The vector in the range space is:
    1.1312
   -1.9506
Sanity check (should be same as y_range_recomputed):
    1.1312
   -1.9506
--- Processing Matrix mtx4 ---
No nullspace exists for this matrix
The vector in the range space is:
    0.1405
    1.9364
Sanity check (should be same as y_range_recomputed):
    0.1405
    1.9364
--- Processing Matrix mtx5 ---
There is a nullspace of size 2
The vector in the null space is:
    0.4504
    0.2808
    0.3791
Sanity check (should be near zero):
   1.0e-16 *

   -0.5551
    0.0694
The vector in the range space is:
    0.4181
   -0.0625
Sanity check (should be same as y_range_recomputed):
    0.4181
   -0.0625
```