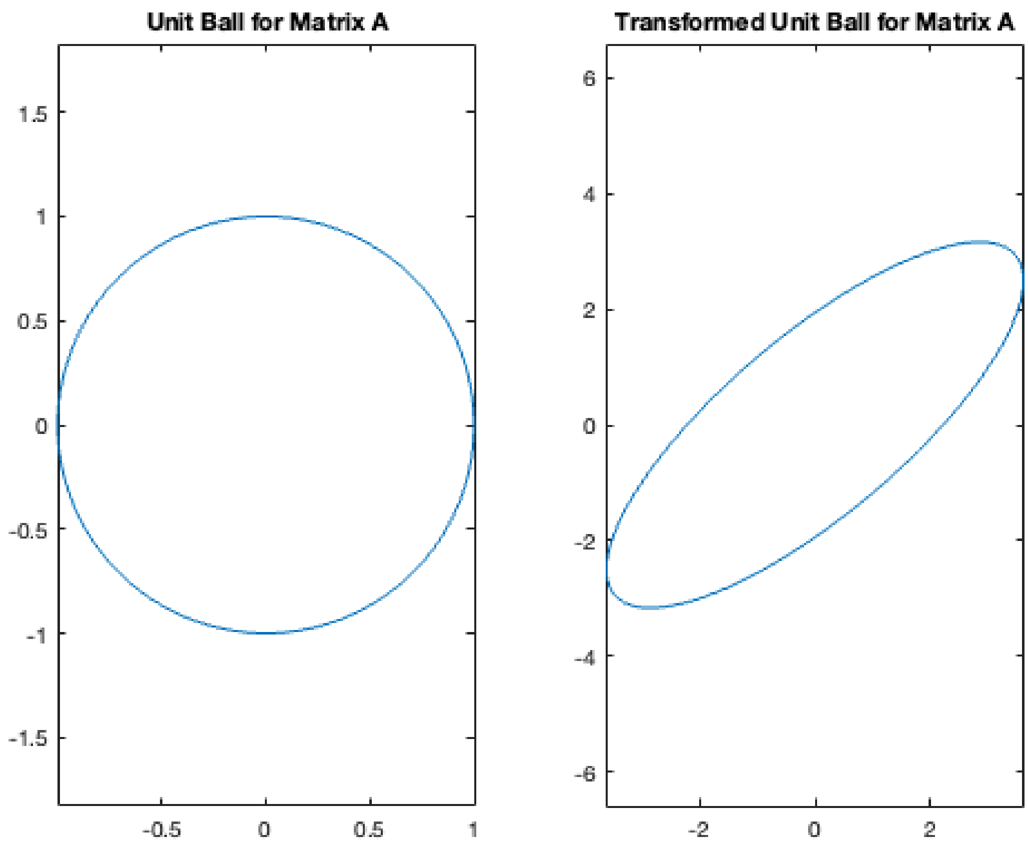
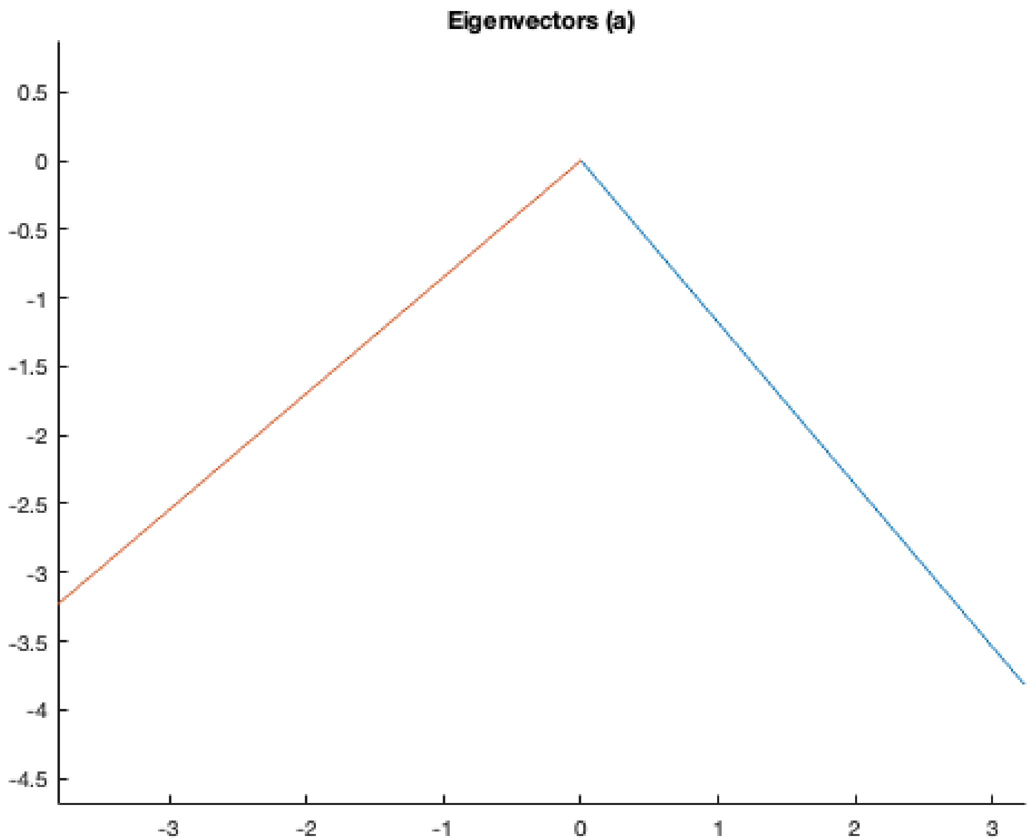
```
% ME564_HW5
% Q1

% a
% Part i: Find Eigenvalues and Eigenvectors
A = [2, 3; 3, 1];
[V, D] = eig(A);

% Part ii: Plot the Unit Ball and Its Transformation
theta = linspace(0, 2*pi, 100);
x = cos(theta);
y = sin(theta);
unitBall = [x; y];

transformedBall = A * unitBall;

figure;
subplot(1,2,1);
plot(x, y);
title('Unit Ball for Matrix A');
axis equal;

subplot(1,2,2);
plot(transformedBall(1,:), transformedBall(2,:));
title('Transformed Unit Ball for Matrix A');
axis equal;

% Part iii: Plot the Eigenvectors
maxLength = 5;
length = linspace(0, maxLength, 100);

figure;
hold on;
for i = 1:size(V, 2)
    v = V(:, i);
    plot(length * v(1), length * v(2));
end
title('Eigenvectors (a)');
axis equal;
hold off;

% Part iv: Find the Value of maxLength
eigenvalues = diag(D);
maxLengthValues = 1 ./ eigenvalues;

disp('Values of maxLength for each eigenvector (a):');
disp(maxLengthValues);

% b

% answer to d: The reason eigenvectors were not plotted for part (b) is
% likely because the matrix A in that part is a rotation matrix & for a
```

```
% 2D rotation matrix, the eigenvalues & eigenvectors are complex.
% So the reason might be: Plotting complex eigenvectors in the same
% 2D space as the unit ball and its transformation would not be meaningful,
% as the eigenvectors would not lie in the same real plane.

% Part i: Find Eigenvalues and Eigenvectors for Matrix B
A = [cos(pi/5), -sin(pi/5); sin(pi/5), cos(pi/5)];
[V, D] = eig(A);

% Display Eigenvalues and Eigenvectors
disp('Eigenvalues (b):');
disp(diag(D));
disp('Eigenvectors (b):');
disp(V);

% Part ii: Plot the Unit Ball and Its Transformation for Matrix B
theta = linspace(0, 2*pi, 100);
x = cos(theta);
y = sin(theta);
unitBall = [x; y];

transformedBall = A * unitBall;

figure;
subplot(1,2,1);
plot(x, y);
title('Unit Ball for Matrix B');
axis equal;

subplot(1,2,2);
plot(transformedBall(1,:), transformedBall(2,:));
title('Transformed Unit Ball for Matrix B');
axis equal;

% c
% Part i: Find Eigenvalues and Eigenvectors for Matrix C
A = [7/8, -1/4; -1/8, 1];
[V, D] = eig(A);

% Display Eigenvalues and Eigenvectors
disp('Eigenvalues (c):');
disp(diag(D));
disp('Eigenvectors (c):');
disp(V);

% Part ii: Plot the Unit Ball and Its Transformation for Matrix C
theta = linspace(0, 2*pi, 100);
x = cos(theta);
y = sin(theta);
unitBall = [x; y];

transformedBall = A * unitBall;

figure;
```

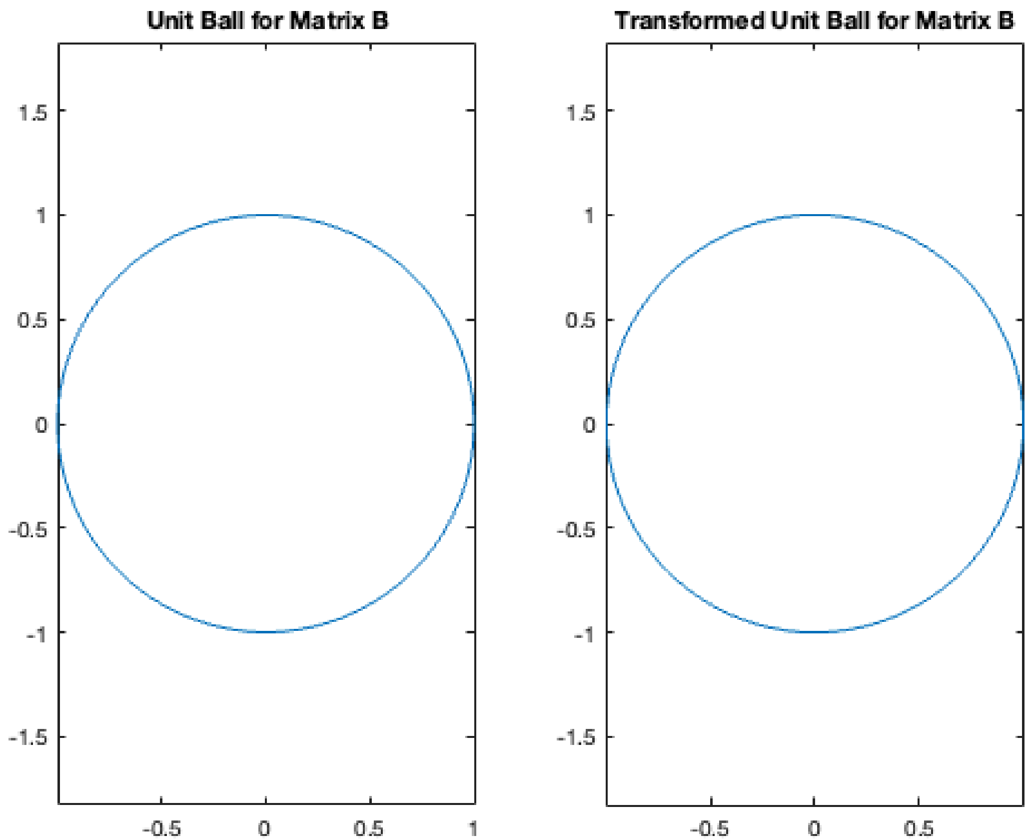
```
subplot(1,2,1);
plot(x, y);
title('Unit Ball for Matrix C');
axis equal;

subplot(1,2,2);
plot(transformedBall(1,:), transformedBall(2,:));
title('Transformed Unit Ball for Matrix C');
axis equal;

% Part iii: Plot the Eigenvectors for Matrix C
maxLength = 5;
length = linspace(0, maxLength, 100);

figure;
hold on;
for i = 1:size(V, 2)
    v = V(:, i);
    plot(length * v(1), length * v(2));
end
title('Eigenvectors for Matrix C');
axis equal;
hold off;

% Part iv: Find the Value of maxLength for Matrix C
eigenvalues = diag(D);
maxLengthValues = 1 ./ eigenvalues;

disp('Values of maxLength for each eigenvector for Matrix C:');
disp(maxLengthValues);
```

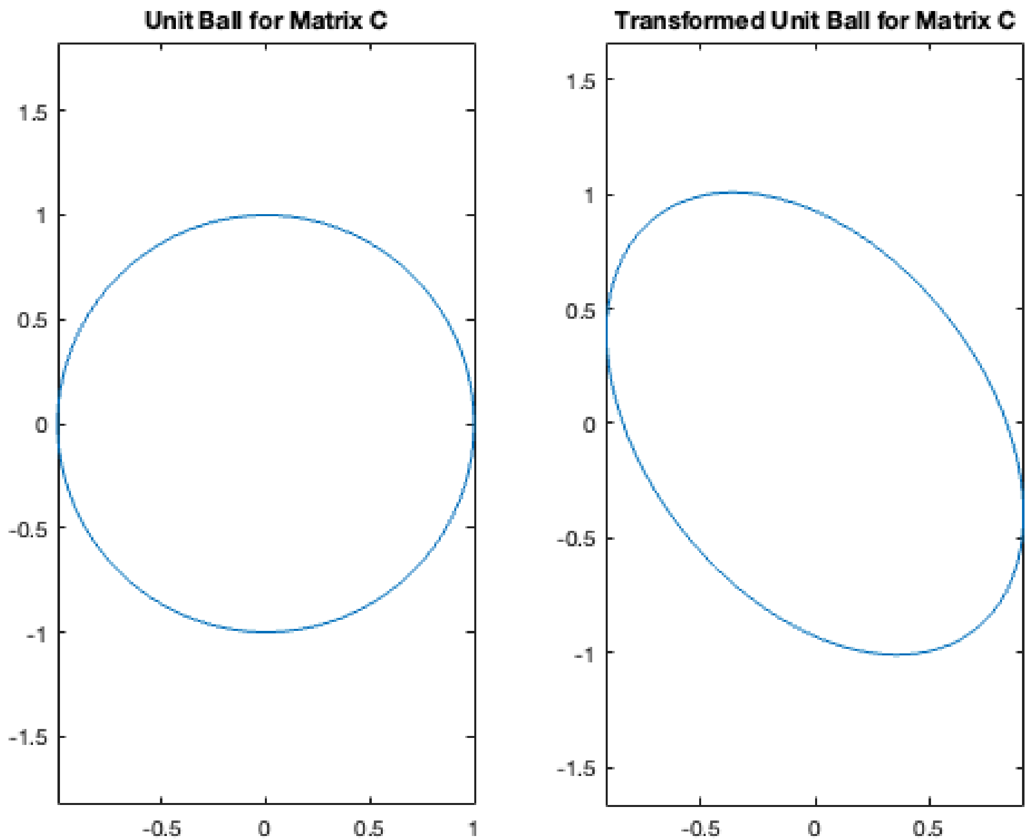
```
Values of maxLength for each eigenvector (a):
   -0.6488
    0.2202

Eigenvalues (b):
   0.8090 + 0.5878i
   0.8090 - 0.5878i

Eigenvectors (b):
   0.7071 + 0.0000i   0.7071 + 0.0000i
   0.0000 - 0.7071i   0.0000 + 0.7071i

Eigenvalues (c):
    0.7500
    1.1250

Eigenvectors (c):
   -0.8944    0.7071
   -0.4472   -0.7071

Values of maxLength for each eigenvector for Matrix C:
    1.3333
    0.8889
```

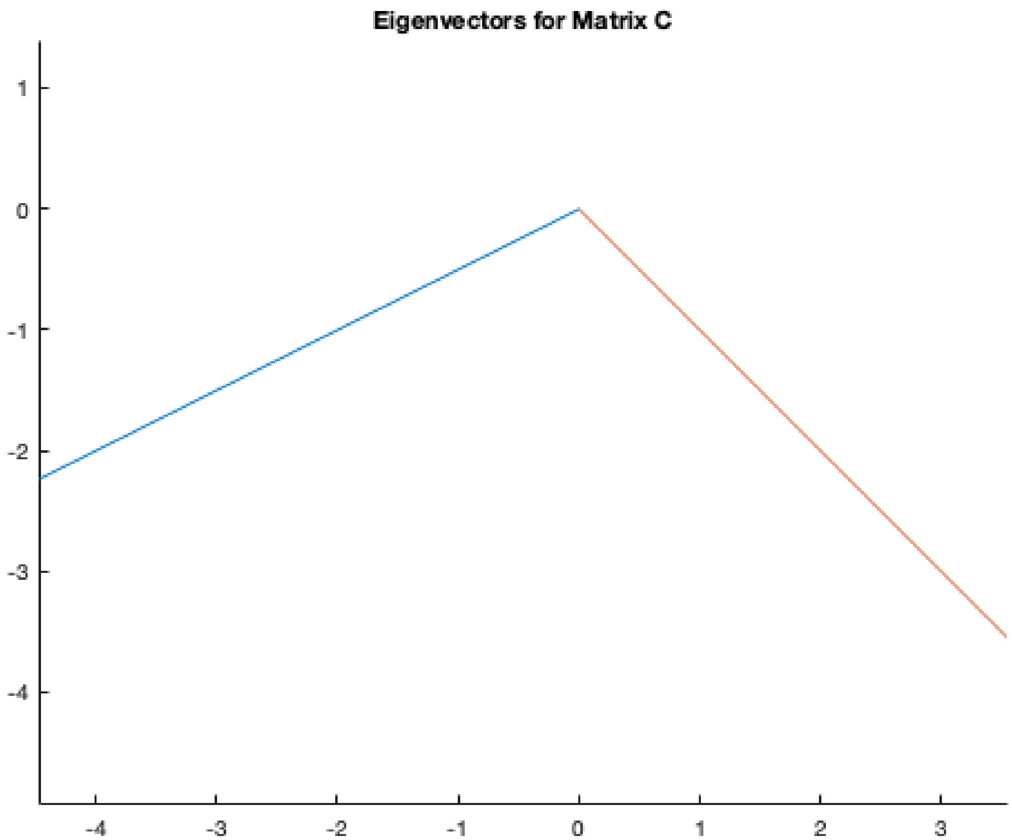

Unit Ball for Matrix A
1.5
1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5
1
Transformed Unit Ball for Matrix A
6
4
2
0
-2
-4
-6
-2
0
2



Eigenvectors (a)
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
-4.5
-3
-2
-1
0
1
2
3


Unit Ball for Matrix B
Transformed Unit Ball for Matrix B
1.5
1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5
1


Unit Ball for Matrix C
Transformed Unit Ball for Matrix C
1.5
1
0.5
0
-0.5
-1
-1.5
-0.5
0
0.5
1



Published with MATLAB® R2022b

```
% ME564_HW5
% Q3

% Part a
disp('Part a:');

% i. Find the eigenvalues
B = [8, -8, -2; 4, -3, -2; 3, -4, 1];
eigenvalues = eig(B);
disp('Eigenvalues:');
disp(eigenvalues);

% ii. Find eigenvectors and/or generalized eigenvectors
[V, J] = jordan(B);
disp('Eigenvectors/Generalized Eigenvectors (columns of P):');
disp(V);

% iii. Compute the Jordan form
J_computed = inv(V) * B * V;
disp('Jordan Form:');
disp(J_computed);

% Double-check with MATLAB's jordan function
[J_check, P_check] = jordan(B);
disp('Jordan Form (MATLAB check):');
disp(J_check);

% Part b
disp('Part b:');

% i. Find the eigenvalues
B = [2, 1, 1; 0, 3, 1; 0, -1, 1];
eigenvalues = eig(B);
disp('Eigenvalues:');
disp(eigenvalues);

% ii. Find eigenvectors and/or generalized eigenvectors
[V, J] = jordan(B);
disp('Eigenvectors/Generalized Eigenvectors (columns of P):');
disp(V);

% iii. Compute the Jordan form
J_computed = inv(V) * B * V;
disp('Jordan Form:');
disp(J_computed);

% Double-check with MATLAB's jordan function
[J_check, P_check] = jordan(B);
disp('Jordan Form (MATLAB check):');
disp(J_check);
```

Part a:

```
Eigenvalues:
    1.0000
    3.0000
    2.0000

Eigenvectors/Generalized Eigenvectors (columns of P):
    2.0000    2.0000    3.0000
    1.0000    1.5000    2.0000
    1.0000    1.0000    1.0000

Jordan Form:
     3     0     0
     0     1     0
     0     0     2

Jordan Form (MATLAB check):
    2.0000    2.0000    3.0000
    1.0000    1.5000    2.0000
    1.0000    1.0000    1.0000

Part b:
Eigenvalues:
    2.0000
    2.0000
    2.0000

Eigenvectors/Generalized Eigenvectors (columns of P):
     1     0     0
     1     1    -1
    -1     0     1

Jordan Form:
     2     1     0
     0     2     0
     0     0     2

Jordan Form (MATLAB check):
     1     0     0
     1     1    -1
    -1     0     1
```

Published with MATLAB® R2022b

```
% ME564_HW5
% Q4

% Define System Matrices
% The system is defined by the equation x(t+1) = Ax(t) + Bu(t) and y(t) =
 Cx(t)
% A is the state transition matrix, B is the input matrix, and C is the output
 matrix
A = [3, 0, -2; 0, 2, 5; 4, 3, -1];
B = [2, 0; 0, 0; 0, 1];
C = [1, 0, 1];

% Proposed Solution for Controllability
% The controllability of the system is checked using the controllability
 matrix.
% The controllability matrix is formed by [B, AB, A^2B, ..., A^(n-1)B]
% If the controllability matrix is of full rank, then the system is
 controllable.

% Calculate the controllability matrix
n = size(A, 1);  % Number of states
ControllabilityMatrix = [];
for i = 0:n-1
    ControllabilityMatrix = [ControllabilityMatrix, A^i * B];
end

% Check if the system is controllable
rank_C = rank(ControllabilityMatrix);
if rank_C == n
    disp('The system is controllable.');
else
    disp('The system is not controllable.');
end

% Proposed Solution for Observability
% The observability of the system is checked using the observability matrix.
% The observability matrix is formed by [C; CA; CA^2; ...; CA^(n-1)]
% If the observability matrix is of full rank, then the system is observable.

% Calculate the observability matrix
ObservabilityMatrix = [];
for i = 0:n-1
    ObservabilityMatrix = [ObservabilityMatrix; C * A^i];
end

% Check if the system is observable
rank_O = rank(ObservabilityMatrix);
if rank_O == n
    disp('The system is observable.');
else
    disp('The system is not observable.');
end
```

```
The system is controllable.
The system is observable.
```

Published with MATLAB® R2022b

```
% ME564_HW5
% Q7

% a
% Define the matrix A
A = [2, 1; 1, -8];

% Initialize the approximation for e^A as the identity matrix
approx_eA = eye(size(A));

% Compute the approximation using the first five terms of the series
 definition
terms = 5;
for k = 1:terms
    approx_eA = approx_eA + (A^k) / factorial(k);
end

disp('Approximation for e^A using the first five terms:');
disp(approx_eA);

% b
% Initialize the approximation for e^-A as the identity matrix
approx_e_neg_A = eye(size(A));

% Compute the approximation using the first five terms of the series
 definition
for k = 1:terms
    approx_e_neg_A = approx_e_neg_A + ((-1)^k) * (A^k) / factorial(k);
end

% Compute the approximation for e^A as the inverse of approx_e_neg_A
approx_eA = inv(approx_e_neg_A);

disp('Approximation for e^A using the first five terms and inverse method:');
disp(approx_eA);

% c
% Compute the eigenvalues and eigenvectors
[V, D] = eig(A);

disp('Eigenvalues of A:');
disp(diag(D));
disp('Eigenvectors of A (columns):');
disp(V);

% d
% Compute e^D
eD = exp(diag(D));

% Compute e^A using the formula e^A = P e^D P^{-1}
eA_exact = V * diag(eD) / V;
```

```
% Compute e^A using MATLAB's expm function
eA_matlab = expm(A);

% Display the results
disp('Exact value of e^A:');
disp(eA_exact);
disp('Value of e^A using MATLAB''s expm function:');
disp(eA_matlab);

% e
% Construct matrix P using the eigenvectors
P = V;

% Construct diagonal matrix # using the eigenvalues
Lambda = D;

% Compute e^#
eLambda = exp(diag(Lambda));

% Compute e^A using the formula e^A = P * (e^#) * P^(-1)
eA_diagonalization = P * diag(eLambda) / P;

% Display the results
disp('Value of e^A using diagonalization:');
disp(eA_diagonalization);
disp('Value of e^A using MATLAB''s expm function for comparison:');
disp(expm(A));
```

```
Approximation for e^A using the first five terms:
    6.2250   17.8417
   17.8417 -172.1917

Approximation for e^A using the first five terms and inverse method:
   31.0825    3.0776
    3.0776    0.3064

Eigenvalues of A:
   -8.0990
    2.0990

Eigenvectors of A (columns):
   -0.0985   -0.9951
    0.9951   -0.0985

Exact value of e^A:
    8.0790    0.7999
    0.7999    0.0795

Value of e^A using MATLAB's expm function:
    8.0790    0.7999
    0.7999    0.0795

Value of e^A using diagonalization:
    8.0790    0.7999
```

```
    0.7999    0.0795

Value of e^A using MATLAB's expm function for comparison:
    8.0790    0.7999
    0.7999    0.0795
```

Published with MATLAB® R2022b