Principal Component Analysis: Evaluating a Spring-Mass System

Christina Smith

AMATH 482 - Winter 2020

Abstract

In this exercise, we evaluate the motion of a spring-mass system under four test cases using principal component analysis. Our data sets contain video filmed from three different perspectives. We perform singular value decomposition on our data sets and use the results to transform our data to new bases. We then use principal component analysis to extract the most relevant data and evaluate the motion of the spring-mass system under given conditions.

I. Introduction

Our data sets consist of videos of an oscillating paint can suspended from a spring. We have four cases and in each case the can is filmed from three different perspectives. The first case shows near ideal harmonic motion; the can is released perpendicular to the ground with no rotation. In the second case, camera shake is introduced to the video. In the third case the can is released at a horizontal displacement from the connection point of the spring. In the fourth case, there is horizontal displacement and the can is also released with rotation. In order to process this data we must track the motion of a point on the paint can for each video clip. We then combine the data for each test into a single matrix for analysis. Using Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) we transform the data to a new basis and calculate a covariance matrix for the data in the new basis. Using this, we can see which modes contain the highest variance and use this information to create a reduced rank approximation of our data that can be plotted.

II. Theoretical Background

Singular Value Decomposition

Since we want to find out about the variation in our data, we create a new basis that most accurately fits to the data by removing redundancies and orders the variances of different measurements. For any given matrix $\mathbf{A}$ the singular value decomposition of $\mathbf{A}$ is as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \tag{1}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{\Sigma}$ is a diagonal matrix. Since we are considering that $\mathbf{A}$ is a matrix where each column represents coordinates measured at a point in time, then for our use, $\mathbf{U}$ contains the principle component vectors that form a basis, $\mathbf{\Sigma}$ contains the singular values that scale the semiaxes, and $\mathbf{V}$ is an orthonormal rotation matrix which is then scaled by $\mathbf{\Sigma}$. Note that by convention, the singular values in $\mathbf{\Sigma}$ are ordered from largest to smallest, left to right.
The more variation a data set exhibits along an axis, the better that axis is for representing the data in the system. In PCA, we use this fact to create a low-rank approximation of the data. For our exercise here we have a set of data in six dimensions and we will use PCA to determine how we can best represent the data in less dimensions.
Our first step is to find the principal components of the data by performing SVD. Once we have done that, we can project our data onto its principal component basis as follows:

$$\mathbf{Y} = \mathbf{U}^T\mathbf{A} \tag{2}$$

This is the state where the data best aligns with the basis and can be evaluated using a reduced rank approximation. In order to do this, we find the covariance matrix of this transformed data.

Covariance and Covariance Matrices

Covariance describes the dependence between data sets. A high covariance means that the data are statistically dependent, and a zero covariance means that there is no dependency (i.e., no redundancy) between data sets.

The formula to compute the covariance between two vectors $\mathbf{a}$ and $\mathbf{b}$ in R^n is

$$\sigma^2_{\mathbf{ab}} = \frac{1}{n-1}\mathbf{a}\mathbf{b}^T \tag{3}$$

In a covariance matrix, the diagonal terms represent the variance within one data set, and the off-diagonal terms represent the covariance between two data sets. Since $\sigma^2_{\mathbf{ab}} = \sigma^2_{\mathbf{ba}}$, any covariance matrix is symmetric.
The formula for the covariance between the vectors in the $\mathbf{Y}$ matrix is

$$\mathbf{C_Y} = \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^T \tag{4}$$

where n is the number of columns and $\frac{1}{n-1}$ is a normalizing factor which provides an unbiased estimator. Using Formulas (1), (2), and (4) we can derive

$$\mathbf{C_Y} = \frac{1}{n-1}\mathbf{\Sigma}^2 \tag{5}$$

This means that the SVD of our matrix **A** give us information about how our data relates to our ideal basis. The non-zero elements of $\mathbf{\Sigma}^2$, σ^2, represent the variance of the data around a given axis in the ideal basis. The higher the variance around an axis, the better it can be used to describe the data.

When we find the axes with the most variance in the basis we can use these to create a reduced rank approximation of the system.

III. Algorithm Implementation and Development

In order to retrieve the data about the paint can's location in each video frame we use Matlab's ginput() method which captures screen coordinates upon a mouse click (See Appx. A). The axes for the clicks is oriented by matching them up with the spring and the top of the can. For each test, we capture the data for each camera in a matrix and then 'crop' the matrices so that the appropriate time samples align.

We then combine these data sets into in a matrix with columns representing the data-points gathered at each time-step. The next step is to normalize the rows in the matrix by dividing each row by its mean. This will ensure that the SVD will best represent the variation of our data along a mode rather than the mean of that data.

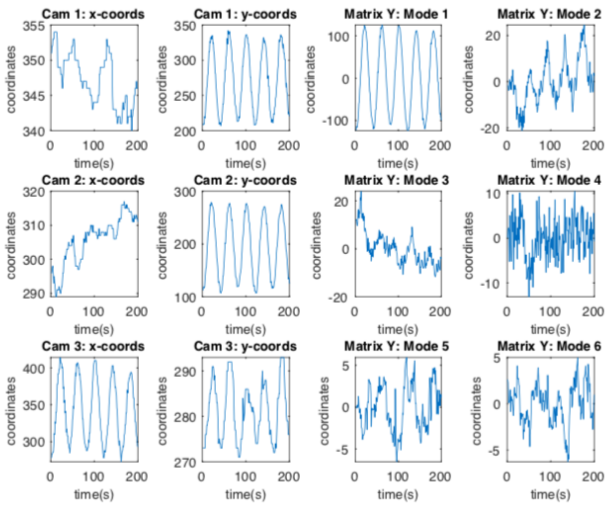

Figure 1: Test 1: Data in original basis and principal component basis

We are now ready to evaluate this data using SVD and PCA. We compute the SVD of $\mathbf{A}$ using Matlab's built in function (See Appx. A) and then use Formula (2) to transform the data to a new basis. We the compute the covariance matrix of the data in the new basis using Formula (5). This gives us the variance around the different axes of the basis. Since the singular values in $\mathbf{\Sigma}$ are ordered from largest to smallest, the variance terms in the covariance matrix are also ordered from largest to smallest.

IV.Computational Results & Supplementary Plots

Test 1: Ideal Signal

Singular values: $\sigma_1 = 1,184,\ \sigma_2 = 134,\ \sigma_3 = 96,\ \sigma_4 = 66,\ \sigma_5 = 38,\ \ \sigma_6 = 30$

Test 2: Noisy Signal (camera shake)

Singular values: $\sigma_1 = 1,047,\ \sigma_2 = 566,\ \sigma_3 = 392,\ \sigma_4 = 218,\ \sigma_5 = 202,\ \ \sigma_6 = 145$

Test 3: Horizontal Displacement

Singular values: $\sigma_1 = 693,\ \sigma_2 = 416,\ \sigma_3 = 202,\ \sigma_4 = 142,\ \sigma_5 = 70,\ \ \sigma_6 = 43$

Test 4: Horizontal Displacement & Rotation

Singular values: $\sigma_1 = 1,403,\ \sigma_2 = 1,046,\ \sigma_3 = 334,\ \sigma_4 = 165,\ \sigma_5 = 117,\ \ \sigma_6 = 71$

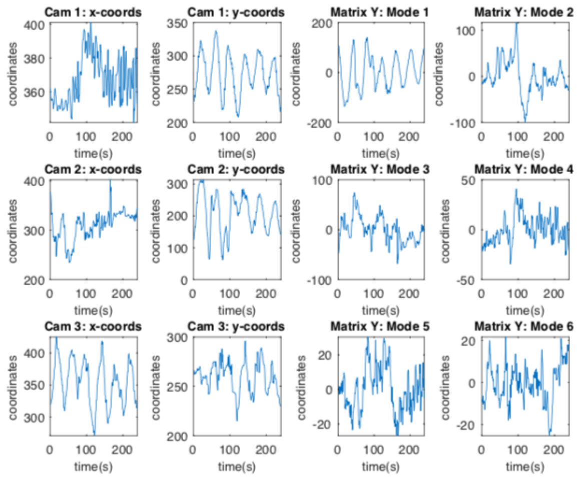

Figure 2: Test 2: Data in original basis and principal component basis

V.Summary and Conclusions

Through the evaluating our data in this exercise by reducing redundancy and we can come to the following conclusions

Test 1 - Ideal Signal (See Fig. 1)

The analysis of this data shows the large majority of the energy in this system can be attributed to one mode and therefore, a rank-1 approximation will provide very accurate data about the behavior of this system. This makes sense because in all three videos, the motion of the can is linear.

Test 2 - Camera Shake (See Fig. 2)

In this case, we see that the the energy is more spread out throughout the modes. Although most of the energy is still contained in one mode, the energy levels in the 2^{nd} and 3^{rd} modes have increased compared to those in the ideal case. When viewing the footage from Cam 2, we can see that some of the camera shake is in the direction of the paint can's oscillation. This would have less of a negative impact on the energy represented by the 1^{st} mode than camera shake that's perpendicular to the line of oscillation.

Test 3 - Horizontal Displacement (See Fig. 3)

In this case a horizontal displacement causes move in an arc perpendicular to the main line of oscillation for the entirety of the video. Of all the cases, the data in this set is the most spread out. Less of the energy is represented by the 1^{st} mode and more represented by the second. This fits with what we know of a spring system. The

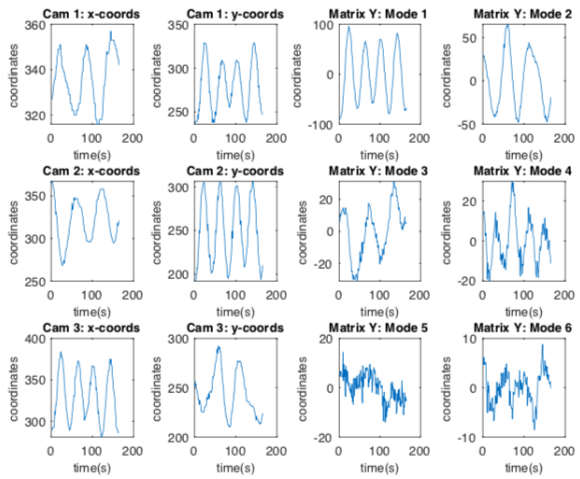

Figure 3: Test 3: Data in original basis and principal component basis

force of the spring increases as it stretches and so stretching away from the original line of oscillation will result in the paint can travelling less distance along that line.

Test 4 - Horizontal Displacement & Rotation (See Fig. 4)

While the energy in this system is more spread out than in cases 1 or 2, more of the energy is contained in the 1st mode here than in case 3. We can see from the video that the effect of the horizontal displacement does not last for the entire video like in case 3. The rotation of the can causes another force to act on it and so the motion becomes near linear again fairly quickly. Again, because we have chosen to track the can at the fixed point where it connects with the spring, this rotation does not have a great effect on our data.

Appendix A: Matlab Functions

implay() / imshow(): These functions takes a video or still image file as an argument and open Matlab's image or video viewer to view the file

ginput(): This function allows us to collect coordinates by clicking on points on a screen. When the function is active a set of axes appear on the screen with the origin at the mouse point

svd(): This function breaks down a matrix in to principal components, singular values and a rotation matrix

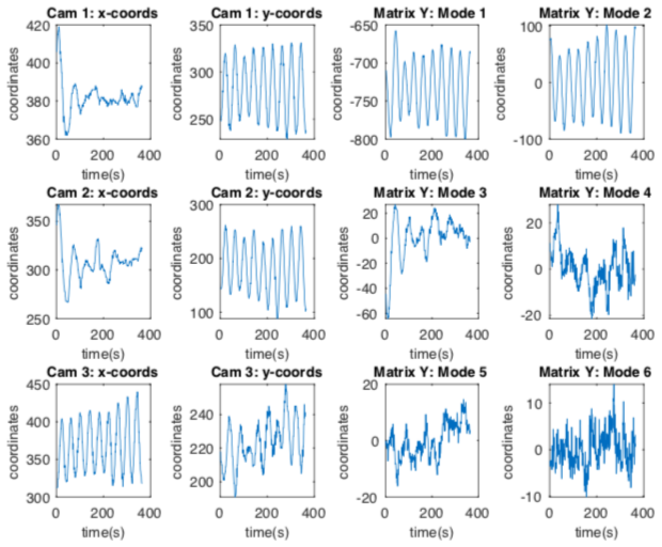

Figure 4: Test 4: Data in original basis and principal component basis

Appendix B: Matlab Code

```
%% HW 03 Principal Component Analysis

clear all; close all; clc

%% ginput() Camera 1 - Test 1

% first vertical min at 30 frames

clear all; close all; clc

load('cam1_1.mat')
implay(vidFrames1_1)
numFrames = size(vidFrames1_1,4);
cam1_1_coords = zeros(numFrames,2);

for j = 1:numFrames
    X = vidFrames1_1(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)
%      [x,y] = ginput(1);
%      cam2_2_coords(j,1) = x;
%      cam2_2_coords(j,2) = y;

end

csvwrite('cam1_1_coords.csv',cam1_1_coords)


%% ginput() Camera 2 - Test 1

clear all; close all; clc

load('cam2_1.mat')
implay(vidFrames2_1)
numFrames = size(vidFrames2_1,4);
cam2_1_coords = zeros(numFrames,2);

for j = 1:numFrames
    X = vidFrames2_1(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)
%      [x,y] = ginput(1);
%      cam2_2_coords(j,1) = x;
%      cam2_2_coords(j,2) = y;

end

csvwrite('cam2_1_coords.csv',cam2_1_coords)

```

```matlab
%% ginput() Camera 3 - Test 1

clear all; close all; clc

load('cam3_1.mat')
implay(vidFrames3_1)
numFrames = size(vidFrames3_1,4);
cam3_1_coords = zeros(numFrames,2);

for j = 1:numFrames
     X = vidFrames3_1(:,:,:,j);
     imshow(X); drawnow
     title(num2str(j), 'Fontsize', 20)
%       [x,y] = ginput(1);
%       cam2_2_coords(j,1) = x;
%       cam2_2_coords(j,2) = y;
     pause(1)
end

csvwrite('cam3_1_coords.csv',cam3_1_coords)

%% ginput() Camera 1 - Test 2

clear all; close all; clc

load('cam1_2.mat');
implay(vidFrames1_2)
numFrames = size(vidFrames1_2,4);
cam1_2_coords = zeros(numFrames,2);

for j = 1:numFrames
     X = vidFrames1_2(:,:,:,j);
     imshow(X); drawnow
     title(num2str(j), 'Fontsize', 20)
%       [x,y] = ginput(1);
%       cam1_2_coords(j,1) = x;
%       cam1_2_coords(j,2) = y;
end

csvwrite('cam1_2_coords.csv',cam1_2_coords)

%% ginput() Camera 2 - Test 2

clear all; close all; clc

load('cam2_2.mat');
implay(vidFrames2_2)
numFrames = size(vidFrames2_2,4);
cam2_2_coords = zeros(numFrames,2);

for j = 1:numFrames
```

```matlab
    X = vidFrames2_2(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)
%     [x,y] = ginput(1);
%     cam2_2_coords(j,1) = x;
%     cam2_2_coords(j,2) = y;
end

% csvwrite('cam2_2_coords.csv',cam2_2_coords)

%% ginput() Camera 3 - Test 2

clear all; close all; clc

load('cam3_2.mat')
implay(vidFrames3_2)
numFrames = size(vidFrames3_2,4);
cam3_2_coords = zeros(numFrames,2);

for j = 1:numFrames
    X = vidFrames3_2(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)
%     [x,y] = ginput(1);
%     cam2_2_coords(j,1) = x;
%     cam2_2_coords(j,2) = y;

end

csvwrite('cam3_2_coords.csv',cam3_2_coords)

%% ginput() Camera 1 - Test 3

clear all; close all; clc

load('cam1_3.mat');
numFrames = size(vidFrames1_3,4);
cam1_3_coords = zeros(numFrames,2);

for j = 1:239
    X = vidFrames1_3(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)
    [x,y] = ginput(1);
    cam1_3_coords(j,1) = x;
    cam1_3_coords(j,2) = y;
    % pause(0.75)
end

csvwrite('cam1_3_coords.csv',cam1_3_coords)

```

```
%% ginput() Camera 2 - Test 3

clear all; close all; clc

load('cam2_3.mat');
numFrames = size(vidFrames2_3,4);
cam2_3_coords = zeros(numFrames,2);

for j = 1:numFrames
    X = vidFrames2_3(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)
    [x,y] = ginput(1);
    cam2_3_coords(j,1) = x;
    cam2_3_coords(j,2) = y;
    % pause(0.75)
end

csvwrite('cam2_3_coords.csv',cam2_3_coords)

%% ginput() Camera 3 - Test 3

clear all; close all; clc

load('cam3_3.mat');
numFrames = size(vidFrames3_3,4);
cam3_3_coords = zeros(numFrames,2);

for j = 1:numFrames
    X = vidFrames3_3(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)
    [x,y] = ginput(1);
    cam3_3_coords(j,1) = x;
    cam3_3_coords(j,2) = y;
    % pause(0.75)
end

csvwrite('cam3_3_coords.csv',cam3_3_coords)

%% ginput() Camera 1 - Test 4

clear all; close all; clc

load('cam1_4.mat')
% implay(vidFrames1_4)
numFrames = size(vidFrames1_4,4);
% clippedGrey = zeros(380,80,1,226); % to hold the clipped data\
cam1_4_coords = zeros(numFrames,2);

for j = 1:392
```

```
    X = vidFrames1_4(:,:,:,j);
    imshow(X); drawnow
    %newFrame = im2double(X(50:429, 310:389));
    %clippedGrey(:,:,:,j) = newFrame;
    %imshow(clippedGrey(:,:,:,j)); drawnow
    title(num2str(j))
    [x,y] = ginput(1);
    cam1_4_coords(j,1) = x;
    cam1_4_coords(j,2) = y;
    pause(1)
end

csvwrite('cam1_4_coords.csv',cam1_4_coords)

%% ginput() Camera 2 - Test 4

clear all; close all; clc

load('cam2_4.mat')
% implay(vidFrames2_4)
numFrames = size(vidFrames2_4,4);
cam2_4_coords = zeros(numFrames,2);

for j = 1:405
    X = vidFrames2_4(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)

    objectRegion = [150 150 300 250];
    if (mod(j,2) ==0)
        objectImage = insertShape(X,'Rectangle',objectRegion,...
            'Color','blue', 'Linewidth', 2);
    figure(2);
    imshow(objectImage);
    else
    objectImage = insertShape(X,'Rectangle',objectRegion,...
        'Color','red','Linewidth', 2);
    figure(2);
    imshow(objectImage);
    end

    [x,y] = ginput(1);
    cam2_4_coords(j,1) = x;
    cam2_4_coords(j,2) = y;
    pause(0.75)
end

csvwrite('cam2_4_coords.csv',cam2_4_coords)


%% ginput() Camera 3 - Test 4
```

```

clear all; close all; clc

load('cam3_4.mat')
% implay(vidFrames3_4)
numFrames = size(vidFrames3_4,4);
cam3_4_coords = zeros(numFrames,2);

for j = 1:394
    X = vidFrames3_4(:,:,:,j);
    imshow(X); drawnow
    title(num2str(j), 'Fontsize', 20)

    objectRegion = [150 150 300 250];
    if (mod(j,2) ==0)
        objectImage = insertShape(X,'Rectangle',objectRegion,...
            'Color','blue', 'Linewidth', 2);
    figure(2);
    imshow(objectImage);
    else
    objectImage = insertShape(X,'Rectangle',objectRegion,...
        'Color','red','Linewidth', 2);
    figure(2);
    imshow(objectImage);
    end

    [x,y] = ginput(1);
    cam3_4_coords(j,1) = x;
    cam3_4_coords(j,2) = y;
    % pause(0.75)
end

csvwrite('cam3_4_coords.csv',cam3_4_coords)

%% Build the Matrices

% build matrix A1

% get full coordinates from csv file
test1_1_coords = table2array(readtable('coordinates/cam1_1_coords.csv'))
    ;
test1_2_coords = table2array(readtable('coordinates/cam2_1_coords.csv'))
    ;
test1_3_coords = table2array(readtable('coordinates/cam3_1_coords.csv'))
    ;

% adjusted coordinates
subset_1_1 = test1_1_coords(10:209, 1:2);
subset_1_2 = test1_2_coords(19:218, 1:2);
subset_1_3 = test1_3_coords(08:207, 1:2);

```

```
% create the matrix the raw data
A_1 = zeros(6,length(subset_1_1));
n = length(A_1(1,:));

for j=1:n
    A_1(1,j) = subset_1_1(j,1);
    A_1(2,j) = subset_1_1(j,2);
    A_1(3,j) = subset_1_2(j,1);
    A_1(4,j) = subset_1_2(j,2);
    A_1(5,j) = subset_1_3(j,1);
    A_1(6,j) = subset_1_3(j,2);
end

csvwrite('A_1.csv',A_1)

% build matrix A2

% get full coordinates from csv file
test2_1_coords = table2array(readtable('coordinates/cam1_2_coords.csv'));
test2_2_coords = table2array(readtable('coordinates/cam2_2_coords.csv'));
test2_3_coords = table2array(readtable('coordinates/cam3_2_coords.csv'));

% adjusted coordinates
subset_2_1 = test2_1_coords(35:274, 1:2);
subset_2_2 = test2_2_coords(18:257, 1:2);
subset_2_3 = test2_3_coords(40:279, 1:2);

% create the matrix for the raw data
A_2 = zeros(6,length(subset_2_1));
n = length(A_2(1,:));
for j=1:n
    A_2(1,j) = subset_2_1(j,1);
    A_2(2,j) = subset_2_1(j,2);
    A_2(3,j) = subset_2_2(j,1);
    A_2(4,j) = subset_2_2(j,2);
    A_2(5,j) = subset_2_3(j,1);
    A_2(6,j) = subset_2_3(j,2);
end

csvwrite('A_2.csv',A_2)

% build matrix A3

% get full coordinates from csv file
test3_1_coords = table2array(readtable('coordinates/cam1_3_coords.csv'));
test3_2_coords = table2array(readtable('coordinates/cam2_3_coords.csv'));
```

```
test3_3_coords = table2array(readtable('coordinates/cam3_3_coords.csv'))
    ;

% adjusted coordinates
subset_3_1 = test3_1_coords(35:199, 1:2);
subset_3_2 = test3_2_coords(25:189, 1:2);
subset_3_3 = test3_3_coords(28:192, 1:2);

% create the matrix for the raw data
A_3 = zeros(6,length(subset_3_1));
n = length(A_3(1,:));

for j=1:length(subset_3_1)
    A_3(1,j) = subset_3_1(j,1);
    A_3(2,j) = subset_3_1(j,2);
    A_3(3,j) = subset_3_2(j,1);
    A_3(4,j) = subset_3_2(j,2);
    A_3(5,j) = subset_3_3(j,1);
    A_3(6,j) = subset_3_3(j,2);
end

csvwrite('A_3.csv',A_3)

% build matrix A4

% get full coordinates from csv file
test4_1_coords = table2array(readtable('coordinates/cam1_4_coords.csv'))
    ;
test4_2_coords = table2array(readtable('coordinates/cam2_4_coords.csv'))
    ;
test4_3_coords = table2array(readtable('coordinates/cam3_4_coords.csv'))
    ;

% adjusted coordinates
subset_4_1 = test4_1_coords(12:375, 1:2);
subset_4_2 = test4_2_coords(18:381, 1:2);
subset_4_3 = test4_3_coords(11:374, 1:2);

% create the matrix for the raw data
A_4 = zeros(6,length(subset_4_1));
n = length(A_4(1,:));

for j=1:length(subset_4_1)
    A_4(1,j) = subset_4_1(j,1);
    A_4(2,j) = subset_4_1(j,2);
    A_4(3,j) = subset_4_2(j,1);
    A_4(4,j) = subset_4_2(j,2);
    A_4(5,j) = subset_4_3(j,1);
    A_4(6,j) = subset_4_3(j,2);
end

```

```
csvwrite('A_4.csv',A_4)

%% Test #1

% Singular Value Decomposition

clear all; close all; clc;

% original data matrix
A1 = readtable('A_1.csv');
A1 = table2array(A1);

% number of time samples
n = length(A1(1,:));

% plot the A coordinates in each of the 6 dimension over time
figure(1)
subplot(3,4,1);
plot(A1(1,:))
title("Cam 1: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,2);
plot(A1(2,:))
title("Cam 1: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,5);
plot(A1(3,:))
title("Cam 2: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,6);
plot(A1(4,:))
title("Cam 2: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,9);
plot(A1(5,:))
title("Cam 3: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,10);
plot(A1(6,:))
title("Cam 3: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)


% normalize the rows by subtracting the mean
for j=1:length(A1(:,1))
```

```
    A1(j,:) = A1(j,:)-mean(A1(j,:));
end


% get the SVD of A
[U1,S1,~] = svd(A1);

% find the data in the new basis
Y1 = U1'*A1;

% plot the Y coordinates in each of the 6 dimension over time

subplot(3,4,3);
plot(Y1(1,:))
title("Matrix Y: Mode 1", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,4);
plot(Y1(2,:))
title("Matrix Y: Mode 2", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,7);
plot(Y1(3,:))
title("Matrix Y: Mode 3", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,8);
plot(Y1(4,:))
title("Matrix Y: Mode 4", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,11);
plot(Y1(5,:))
title("Matrix Y: Mode 5", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,12);
plot(Y1(6,:))
title("Matrix Y: Mode 6", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)


% get the diagonalized covariance matrix
C1 = (1/(n-1))*S1.^2;

% calculate energy in each mode
energy = S1./sum(sum(S1));
energy_squared = S1.^2./sum(sum(S1.^2));
sum(sum(energy))
sum(sum(energy_squared))

% Plot singular values and energy
figure(3)
subplot(2,2,1)
plot(S1,'ko','Linewidth',2)
% axis([0 25 0 50])
ylabel('\sigma')
set(gca,'Fontsize',12,'Xtick',0:6)

subplot(2,2,2)
semilogy(S1,'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('\Sigma (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,5,5))

subplot(2,2,3)
plot(S1.^2/sum(S1.^2),'ko','Linewidth',2)
% axis([0 25 0 1])
ylabel('Energy')
set(gca,'Fontsize',12,'Xtick',0:6)

subplot(2,2,4)
semilogy(S1.^2/sum(S1.^2),'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('Energy (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,0,4))
annotation('textbox', [0 0.9 1 0.1], ...
    'String', 'Test 1 - Ideal Harmonic Motion, Singular Values and
        Energy', ...
    'Fontsize', 15,...
    'Fontweight', 'bold',...
    'EdgeColor', 'none', ...
    'HorizontalAlignment', 'center')

%% Test #2

% Singular Value Decomposition

clear all; close all; clc;

% original data matrix
A2 = readtable('A_2.csv');
A2 = table2array(A2);

% number of time samples
n = length(A2(1,:));

% plot the A coordinates in each of the 6 dimension over time
figure(1)
subplot(3,4,1);
```

```
plot(A2(1,:))
title("Cam 1: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,2);
plot(A2(2,:))
title("Cam 1: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,5);
plot(A2(3,:))
title("Cam 2: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,6);
plot(A2(4,:))
title("Cam 2: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,9);
plot(A2(5,:))
title("Cam 3: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,10);
plot(A2(6,:))
title("Cam 3: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)

% normalize the rows by subtracting the mean
for j=1:length(A2(:,1))
    A2(j,:) = A2(j,:)-mean(A2(j,:));
end


% get the SVD of A
[U2,S2,~] = svd(A2);

% find the data in the new basis
Y2 = U2'*A2;

% plot the Y coordinates in each of the 6 dimension over time

subplot(3,4,3);
plot(Y2(1,:))
title("Matrix Y: Mode 1", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,4);
plot(Y2(2,:))
```

```
title("Matrix Y: Mode 2", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,7);
plot(Y2(3,:))
title("Matrix Y: Mode 3", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,8);
plot(Y2(4,:))
title("Matrix Y: Mode 4", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,11);
plot(Y2(5,:))
title("Matrix Y: Mode 5", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,12);
plot(Y2(6,:))
title("Matrix Y: Mode 6", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)

% get the diagonalized covariance matrix
C2 = (1/(n-1))*S2.^2;

% calculate energy in each mode
energy = S2./sum(sum(S2));
energy_squared = S2.^2./sum(sum(S2.^2));
sum(sum(energy))
sum(sum(energy_squared))

% Plot singular values and energy
figure(3)
subplot(2,2,1)
plot(S2,'ko','Linewidth',2)
% axis([0 25 0 50])
ylabel('\sigma')
set(gca,'Fontsize',12,'Xtick',0:6)

subplot(2,2,2)
semilogy(S2,'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('\Sigma (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,5,5))

subplot(2,2,3)
plot(S2.^2/sum(S2.^2),'ko','Linewidth',2)
% axis([0 25 0 1])
ylabel('Energy')
```

```
set(gca,'Fontsize',12,'Xtick',0:6)

subplot(2,2,4)
semilogy(S2.^2/sum(S2.^2),'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('Energy (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,0,4))
annotation('textbox', [0 0.9 1 0.1], ...
    'String', 'Test 2 - Camera Shake, Singular Values and Energy', ...
    'Fontsize', 13,...
    'Fontweight', 'bold',...
    'EdgeColor', 'none', ...
    'HorizontalAlignment', 'center')

%% Test #3 HORIZONTAL DISPLACEMENT

% Singular Value Decomposition

clear all; close all; clc;

% original data matrix
A3 = readtable('A_3.csv');
A3 = table2array(A3);

% number of time samples
n = length(A3(1,:));

% plot the A coordinates in each of the 6 dimension over time
figure(1)
subplot(3,4,1);
plot(A3(1,:))
title("Cam 1: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,2);
plot(A3(2,:))
title("Cam 1: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,5);
plot(A3(3,:))
title("Cam 2: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,6);
plot(A3(4,:))
title("Cam 2: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,9);
plot(A3(5,:))
```

```
title("Cam 3: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,10);
plot(A3(6,:))
title("Cam 3: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)

% normalize the rows by subtracting the mean
for j=1:length(A3(:,1))
    A3(j,:) = A3(j,:)-mean(A3(j,:));
end


% get the SVD of A
[U3,S3,~] = svd(A3);

% find the data in the new basis
Y3 = U3'*A3;

% plot the Y coordinates in each of the 6 dimension over time
subplot(3,4,3);
plot(Y3(1,:))
title("Matrix Y: Mode 1", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,4);
plot(Y3(2,:))
title("Matrix Y: Mode 2", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,7);
plot(Y3(3,:))
title("Matrix Y: Mode 3", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,8);
plot(Y3(4,:))
title("Matrix Y: Mode 4", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,11);
plot(Y3(5,:))
title("Matrix Y: Mode 5", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,12);
plot(Y3(6,:))
title("Matrix Y: Mode 6", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
```

```
ylabel('coordinates', 'Fontsize', 10)

% get the diagonalized covariance matrix
C3 = (1/(n-1))*S3.^2;

% calculate energy in each mode
energy = S3./sum(sum(S3));
energy_squared = S3.^2./sum(sum(S3.^2));
sum(sum(energy))
sum(sum(energy_squared))

% Plot singular values and energy
figure(3)
subplot(2,2,1)
plot(S3,'ko','Linewidth',2)
% axis([0 25 0 50])
ylabel('\sigma')
set(gca,'Fontsize',12,'Xtick',0:6)

subplot(2,2,2)
semilogy(S3,'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('\Sigma (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,5,5))

subplot(2,2,3)
plot(S3.^2/sum(S3.^2),'ko','Linewidth',2)
% axis([0 25 0 1])
ylabel('Energy')
set(gca,'Fontsize',12,'Xtick',0:6)

subplot(2,2,4)
semilogy(S3.^2/sum(S3.^2),'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('Energy (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,0,4))
annotation('textbox', [0 0.9 1 0.1], ...
    'String', 'Test 3 - Horizontal Displacement, Singular Values and
       Energy', ...
    'Fontsize', 13,...
    'Fontweight', 'bold',...
    'EdgeColor', 'none', ...
    'HorizontalAlignment', 'center')

%% Test #4

close all; clear all; clc;

% Singular Value Decomposition

clear all; close all; clc;
```

```

% original data matrix
A4 = readtable('A_4.csv');
A4 = table2array(A4);

% number of time samples
n = length(A4(1,:));

% plot the A coordinates in each of the 6 dimension over time
figure(1)
subplot(3,4,1);
plot(A4(1,:))
title("Cam 1: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,2);
plot(A4(2,:))
title("Cam 1: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,5);
plot(A4(3,:))
title("Cam 2: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,6);
plot(A4(4,:))
title("Cam 2: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,9);
plot(A4(5,:))
title("Cam 3: x-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,10);
plot(A4(6,:))
title("Cam 3: y-coords", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)


% get the SVD of A
[U4,S4,V4_a] = svd(A4, 'econ');

% find the data in the new basis
Y4 = U4'*A4;

% plot the Y coordinates in each of the 6 dimension over time
subplot(3,4,3);
plot(Y4(1,:))
```

```
title("Matrix Y: Mode 1", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,4);
plot(Y4(2,:))
title("Matrix Y: Mode 2", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,7);
plot(Y4(3,:))
title("Matrix Y: Mode 3", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,8);
plot(Y4(4,:))
title("Matrix Y: Mode 4", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,11);
plot(Y4(5,:))
title("Matrix Y: Mode 5", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)
subplot(3,4,12);
plot(Y4(6,:))
title("Matrix Y: Mode 6", 'Fontsize', 10)
xlabel('time(s)', 'Fontsize', 10)
ylabel('coordinates', 'Fontsize', 10)

% find the covariance matrix of the data in the new basis
C4 = (1/(n-1)).*Y4*Y4';

% diagonalize the covariance matrix with the SVD
[U4_C,S4_C,V4_C] = svd(C4, 'econ');

% calculate energy in each mode

energy1b=S4_C(1)/sum(S4_C)
energy2b = S4_C(1)^2/sum(S4_C.^2)

energy_1 = S4_C./sum(sum(S4_C));
energy_2 = S4_C.^2/sum(S4_C.^2);
energy_3 = S4_C.^2./sum(sum(S4_C.^2));

% Plot singular values and energy
figure(4)
subplot(2,2,1)
plot(S4_C,'ko','Linewidth',2)
% axis([0 25 0 50])
ylabel('\sigma')
set(gca,'Fontsize',12,'Xtick',0:6)
```

```

subplot(2,2,2)
semilogy(S4_C,'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('\Sigma (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,5,5))

subplot(2,2,3)
plot(S4_C.^2/sum(S4_C.^2),'ko','Linewidth',2)
% axis([0 25 0 1])
ylabel('Energy')
set(gca,'Fontsize',12,'Xtick',0:6)

subplot(2,2,4)
semilogy(S4_C.^2/sum(S4_C.^2),'ko','Linewidth',2)
% axis([0 25 10^-(18) 10^5])
ylabel('Energy (log scale)')
set(gca,'Fontsize',12,'Xtick',0:6,'Ytick',logspace(-15,0,4))
annotation('textbox', [0 0.9 1 0.1], ...
    'String', 'Test 4 - Horizontal Displacement & Rotation', ...
    'Fontsize', 15,...
    'Fontweight', 'bold',...
    'EdgeColor', 'none', ...
    'HorizontalAlignment', 'center')
```