AMATH582 Hw3: PCA Analysis

Xiangyu Gao

Feb 24, 2021

Abstract

In this report, we analyzed the movie files of spring experiments created from three different cameras. We illustrated the principal components of 4 different experiments and showed the practical usefulness and the effects of noise on the PCA algorithms.

1 Introduction and Overview

Principal component analysis (PCA) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

Using PCA, we tried to analyze the main movement modes of spring experiments in this report. Particularly, we have 4 different experiments: 1. A small displacement of the mass in the z direction and the ensuing oscillations (ideal case); 2. We repeat the ideal case experiment, but introduce camera shake into the video recording (noisy case); 3. The mass is released off-center so as to produce motion in the $x-y$ plane as well as the z direction (horizontal displacement case); 4. The mass is released off-center and rotates so as to produce motion in the $x-y$ plane, rotation as well as the z direction (horizontal displacement and rotation case).

To perform the PCA analysis, we first extract the position of the mass from each frame of the movie files by identifying the bright spots at limited regions. Then, we concatenate the position vectors from all 6 cameras and calculate the eigenvalues and eigenvectors of the concatenated vectors. We pick the eigenvectors with corresponding eigenvalues that take up 95% of energy, and project the chosen eigenvectors to position vector domain as our principal components.

2 Theoretical Background

The key to analyzing a PCA problem is to consider the covariance matrix:

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T$$

where the matrix X contains the experimental data of the system. In particular, $\mathrm{X} \in \mathbb{C}^{m\times n}$ where m are the number of probes or measuring positions, and n is the number of experimental data points taken at each location. [1].

The most straightforward way to diagonalize the covariance matrix is by using eigenvectors and eigenvalues. We can observe that $\mathrm{X}\mathrm{X}^T$ is a square, symmetric $m \times m$ matrix, i.e. it is self-adjoint so that the m eigenvalues are real and distinct. Linear algebra provides theorems which state that such a matrix can be rewritten as

$$\mathbf{X}\mathbf{X}^T = \mathbf{S}\Lambda\mathbf{S}^{-1}$$

where the matrix $\mathbf{S}$ is a matrix of the eigenvectors of $\mathbf{X}\mathbf{X}^T$ arranged in columns. Since it is a symmetric matrix, these eigenvector columns are orthogonal so that ultimately the $\mathbf{S}$ can be written as a unitary matrix with $\mathrm{S}^{-1} = \mathbf{S}^T$. Recall that the matrix Λ is a diagonal matrix whose entries correspond to the m distinct eigenvalue of $\mathbf{X}\mathbf{X}^T$.

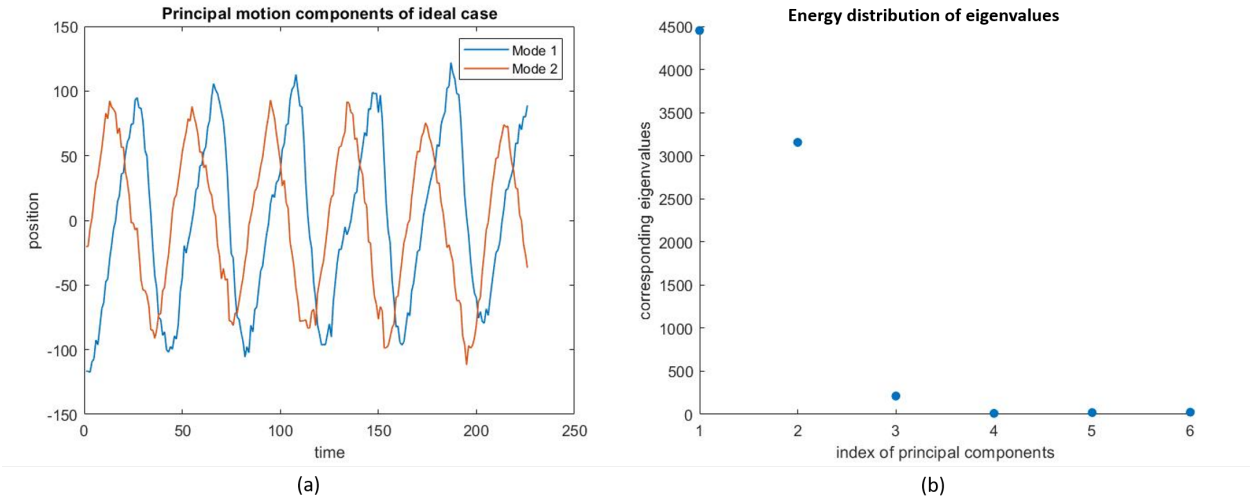

Figure 1: (a) The principal components of test 1; (b) The energy distribution of all eigenvalues.

This suggests that instead of working directly with the matrix $\mathbf{X}$, we consider working with the transformed variable, or in the principal component basis,

$$\mathbf{Y} = \mathbf{S}^T\mathbf{X}$$

3 Algorithm Implementation and Development

We introduce the algorithm implementation details below and attach corresponding MATLAB code in Appendix B.

We first extract the position of mass from each image frame of movie files. To achieve, we transform the RGB image to the grey image and identify the white/bright spots (pixel value greater than 250) from it. The x-positions and y-positions of all identified spots are averaged to estimate the position of mass. We do the same operation for all 3 cameras to obtain their mass positions.

Each camera produces a two-dimensional position representation of the data. All the data collected can then be gathered into a single matrix X by concatenating along the rows.

We subtract the mean value of X from itself and calculate the covariance matrix of X for eigenvalue decomposition. The significant eigenvectors are selected based on their corresponding eigenvalues. Basically, we pick a group of eigenvalues that take up 95% of energy. The chosen eigenvectors are multiplied with X to generate our needed principal components. We repeat the same operation for all 4 experiments and plot the obtained principal components.

4 Computational Results

The principal components of test 1 is shown in Fig. 1. These two components correspond to the first and second eigenvalue respectively. In this case, the entire motion is in the z directions with simple harmonic motion, which is validated by the obtained principal components (follow sin and cos format solution).

The principal components of test 2 is shown in Fig. 2. These three components correspond to the first three eigenvalues respectively. In this case, the entire mass motion is in the z directions with simple harmonic motion with camera shaking. As shown in Fig. 2, the Mode 1 is still with the sin format that corresponds to the harmonic motion, while the jittery Mode 2 and 3 should be caused by the camera shaking.

The principal components of test 3 is shown in Fig. 3. These four components correspond to the first four eigenvalues respectively. In this case, the mass is released off-center so as to produce motion in the $x-y$ plane as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations.

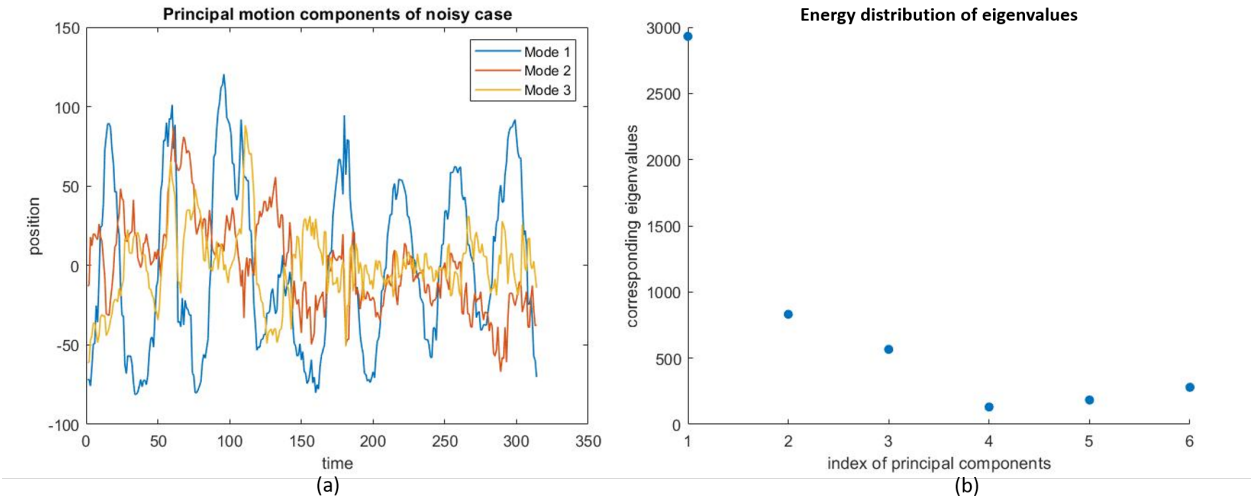

Figure 2: (a) The principal components of test 2; (b) The energy distribution of all eigenvalues.

As presented in Fig. 3, the curve of Mode 1 and 3 are with the sin format, which should correspond to the simple harmonic movement at z direction, while Mode 2 and 4 are supposed to be caused by the pendulum motion. Note that the curves of Mode 2 and 4 have decreased/increased amplitude since the pendulum motion is not a stable status.

The principal components of test 4 is shown in Fig. 4. These four components correspond to the first three eigenvalues and the last one respectively. In this case, the mass is released off-center and rotates so as to produce motion in the $x - y$ plane, rotation as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations. As presented in Fig. 4, the curve of Mode 1 and 2 are with the sin format, which should correspond to the simple harmonic movement at z direction, while Mode 3 and 4 that have fast-decreasing amplitude are supposed to be caused by the pendulum motion.

5 Summary and Conclusions

In this report, we analyzed the movie files of spring experiments created from three different cameras. We illustrated the principal components of 4 different experiments and showed the practical usefulness and the effects of noise on the PCA algorithms.

References

[1] Jose Nathan Kutz. *Data-driven modeling & scientific computation: methods for complex systems & big data*. Oxford University Press, 2013.

Appendix A MATLAB Functions

The important MATLAB functions and thier implementation explanations are listed here:

- `[I,J] = ind2sub(SIZ,IND)` returns the arrays I and J containing the equivalent row and column subscripts corresponding to the index matrix IND for a matrix of size SIZ.
- `[V,D] = eig(A)` produces a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors so that A*V = V*D.

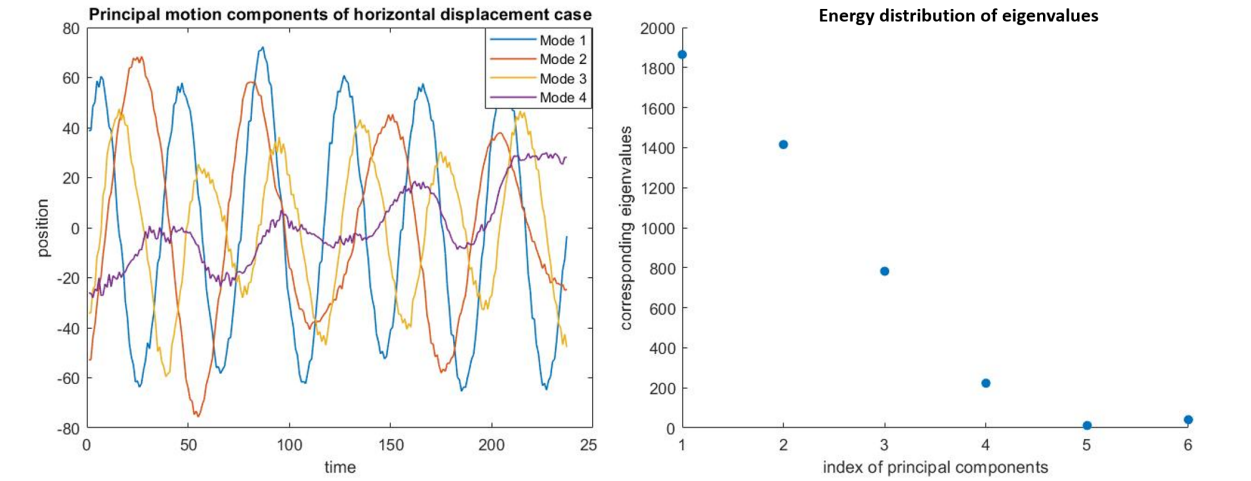

Figure 3: (a) The principal components of test 3; (b) The energy distribution of all eigenvalues.

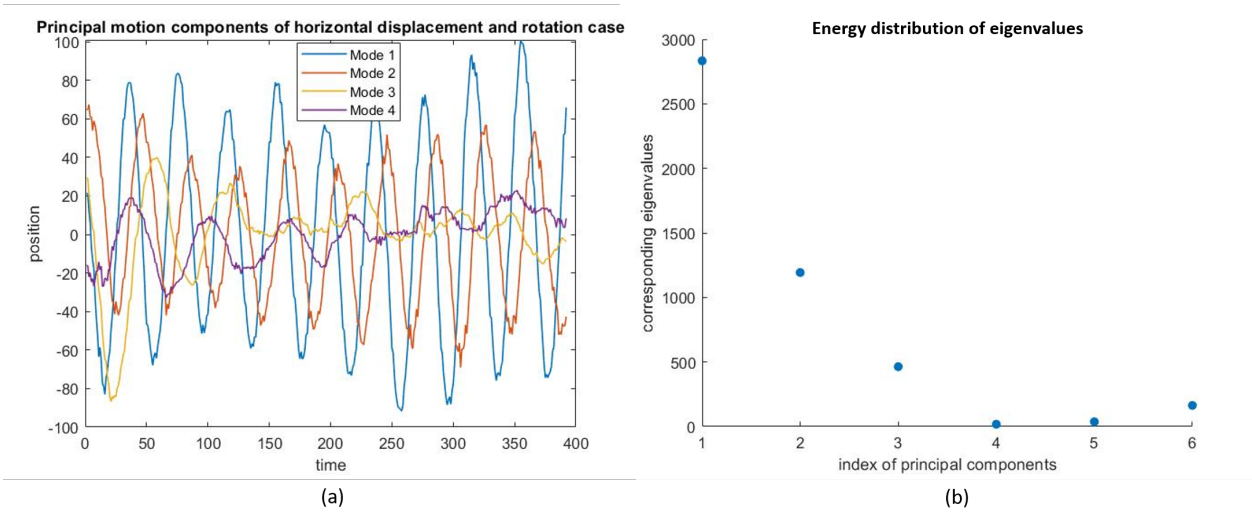

Figure 4: (a) The principal components of test 4; (b) The energy distribution of all eigenvalues.

Appendix B MATLAB Code

The MATLAB codes are shown here. We also stored corresponding codes on Github under the repository (https://github.com/Xiangyu-Gao/AMATH-582-Computational-method-for-data-analysis/tree/main/Hw3).

```
clc
clear all
close all

data = load('C:\Users\Xiangyu Gao\Downloads\582-movie\cam3_4.mat');
vidFrames = data.vidFrames3_4;
numFrames = size(vidFrames, 4);

% cam3_4
y_range = [130, 350];
x_range = [300, 510];

for k = 1 : numFrames
    mov(k).cdata = vidFrames(:,:,:,k);
    mov(k).colormap = [];
end

x_inds_all = [];
y_inds_all = [];

for j = 1:numFrames
    X = frame2im(mov(j));
    J = rgb2gray(X);
    J(1:y_range(1), :) = 0;
    J(y_range(2):end, :) = 0;
    J(:, 1:x_range(1)) = 0;
    J(:, x_range(2):end) = 0;
%     imshow(J); drawnow
    % find the location of white spot top on the mass
    inds = find(J>=235);
    if length(inds) < 1
%         imshow(J); drawnow
        max(J, [], 'all')
        continue
    end
    [y_ind, x_ind] = ind2sub(size(J), inds);
    x_inds_all = [x_inds_all, mean(x_ind)];
    y_inds_all = [y_inds_all, mean(y_ind)];
end

figure(1)
plot(y_inds_all)
save('cam3_4_x.mat', 'x_inds_all', '-v6')
save('cam3_4_y.mat', 'y_inds_all', '-v6')
```

Listing 1: MATLAB code for reading data and extracting mass position.

```
clc
clear all
close all

%% read the saved data
x_inds_all_1 = load('cam1_4_x.mat').x_inds_all;
x_inds_all_2 = load('cam2_4_x.mat').x_inds_all;
x_inds_all_3 = load('cam3_4_x.mat').x_inds_all;

y_inds_all_1 = load('cam1_4_y.mat').y_inds_all;
y_inds_all_2 = load('cam2_4_y.mat').y_inds_all;
y_inds_all_3 = load('cam3_4_y.mat').y_inds_all;

len = min([length(x_inds_all_1), length(x_inds_all_2), length(x_inds_all_3)]);

X = [x_inds_all_1(1:len); y_inds_all_1(1:len); x_inds_all_2(1:len);  ...
    y_inds_all_2(1:len); x_inds_all_3(1:len); y_inds_all_3(1:len)];

%% Eigenvalue Decompsition
[m, n] = size(X);
mn = mean(X, 2); % compute mean for each row
X = X - repmat(mn, 1, n); % subtract mean

Cx = (1/(n-1)) * X * X'; % covariance
[V,D] = eig(Cx); % eigenvectors(V)/eigenvalues(D)
lambda = diag(D); % get eigenvalues

%% PCA analysis
[dummy, m_arrange] = sort(-1*lambda); % sort in decreasing order
lambda = lambda(m_arrange);
V = V(:, m_arrange);
Y = V' * X; % produce the principal components projection

figure(1)
plot(Y(1, :),'LineWidth',1)
hold on
plot(Y(2, :),'LineWidth',1)
plot(Y(3, :),'LineWidth',1)
plot(Y(4, :),'LineWidth',1)

xlabel('time')
ylabel('position')
title('Principal motion components of horizontal displacement and rotation case')
% title('Principal motion components of noisy case')
legend('Mode 1', 'Mode 2', 'Mode 3', 'Mode 4')

figure(2)
scatter(m_arrange, lambda, 'filled')
xlabel('index of principal components')
ylabel('corresponding eigenvalues')
set(gca,'xtick',1:6)
```

Listing 2: MATLAB code for PAC analysis.