AMATH 582 Homework 2

Jesse Dumas

September 14, 2018

This is a discussion of principal component analysis (PCA) and singular value decompostion (SVD) methods and their use in analyzing physical systems. For AMATH 582 homework 2, PCA and SVD were applied to data from video footage of an experiment to uncover the underlying dynamics of a simple harmonic oscillator, in this case, a paint can attached to a spring.

Introduction and Overview

Principal component analysis[1] is a useful method for reducing high dimensional data to low rank data by finding the most important directions for use as basis vectors, and it permits discovery of underlying system dynamics from experimental data. For this homework, the data come from an experiment wherein a paint can attached to a spring was perturbed, and three videos taken from three locations were recorded. By tracking the paint can in the videos, a matrix of positions as a function of time was built that could be used with PCA to uncover the simple harmonic motion governing the movement of the can. The purpose of this exercises is to distill the data down to their most important constituents and to interpret the distilled information into a physical law that can predict the system's behavior in the future.

[1] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data.* OUP Oxford, 2013

This exploration will investigate the data set using the PCA method on four test cases:

- **(test 1) Ideal case:** Small displacement of the paint can in the z-direction.
- **(test 2) Noisy case:** Repeat case 1, but now camera shake is an issue.
- **(test 3) Horizontal displacement case:** The paint can is released off center, so now the motion is governed by pendulum and simple harmonic oscillation.
- **(test 4) Horizontal displacement and rotation case:** Repeat case 3 with added rotation of the paint can.

Theoretical Background

PCA is essentially a specific use of the SVD method. If $\mathbf{A}$ is a data matrix, like a collection of positions evolving in time, SVD will decompose it as

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

which can also be expressed as

$$[\mathbf{A}] = \begin{bmatrix} \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ \\ \end{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}. \tag{2}$$

In equations 1 and 2, Σ is a diagonal matrix of eigenvectors comprising the singular values of **A**. The columns in **U** are orthonormal basis vectors for the space and **V** describes how each face is projected onto the principal components in **U**[2,3]. Examining these principal components can reveal underlying structure in the data that explains the governing behavior of the physical system in question.

[2] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013

[3] J Dumas. Amath 582 hw1, January 2017

Algorithm Implementation and Development

As in homework 1, the implementation of SVD/PCA on the data set was fairly straightforward using MATLAB. The real difficulty was in extracting the can's position data from the videos, and that constituted the bulk of the algorimth development.

The Algorithm

1. Import the video data, convert the video to grayscale, and crop the video.
2. Loop through the frames of each video, and set a threshold value for finding high light intensity pixels.
3. Find high intensity pixels.
4. Store x and y position of the paint can.
5. Combine all position data into a single matrix.
6. Perform SVD.
7. Inspect singular value energies to determine principal components of the data.

The full code for the project is presented in Appendix B.

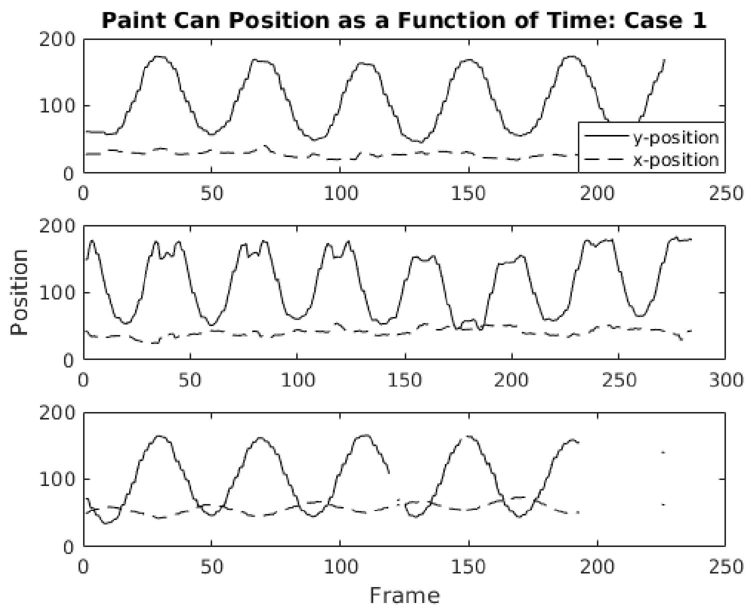

Figure 1: The position of the paint can in case 1 as a function of time as captured by cameras 1, 2, and 3.

Computational Results

Case 1

Case 1 was the most straight forward to approach. Despite some noise in the signals from camera 2 and camera 3, the tracking algorithm was able to capture the motion of the paint can, shown in Figure 1. Performing SVD on the position data from all three cameras reveals that motion in the y-direction accounts for the bulk of the motion, and only some motion occurs in the x-direction. This was the expected behavior, and plots of the singular values of the system in Figure 2 support this.

Case 2

Due to the noise induced by shaking cameras, case 2 was more challenging. Figure 3 shows how noisy the data were. Amazingly, it is still possible to see some semblance of simple harmonic motion, and the singular values of the system suggest that there is one dominant direction in the data, as seen in Figure 4, though that principal component is not as pronounced as in the other cases.

Case 3

Case 3 was successful in capturing the simple harmonic motion coupled with pendulum motion resulting from the paint can's initial

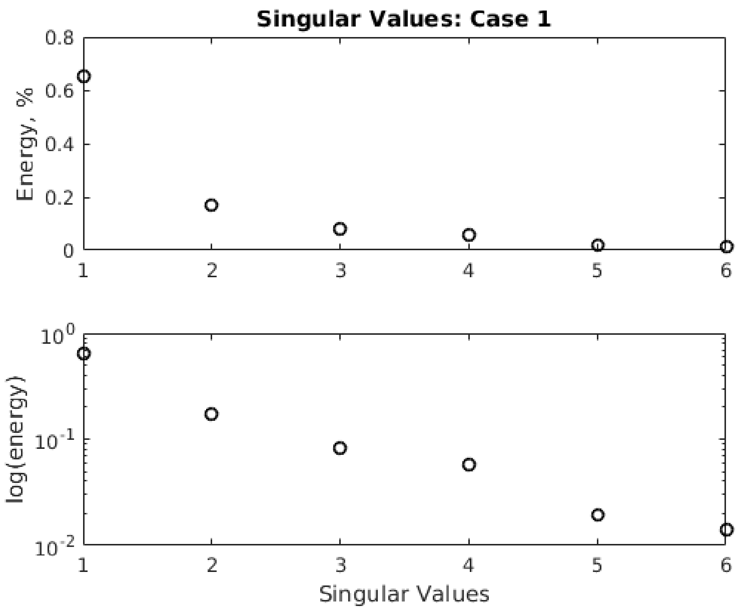

Figure 2: Singular values of the system and their associated energies for case 1.

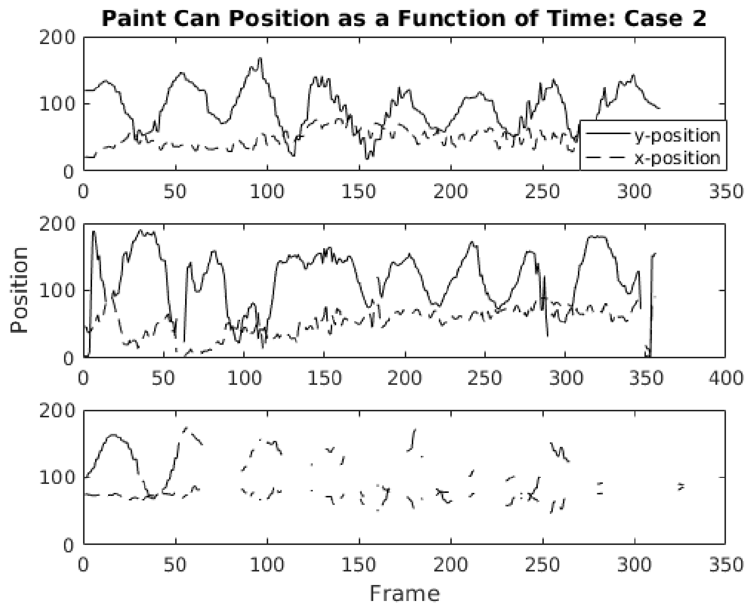

Figure 3: The position of the paint can in case 2 as a function of time as captured by cameras 1, 2, and 3.

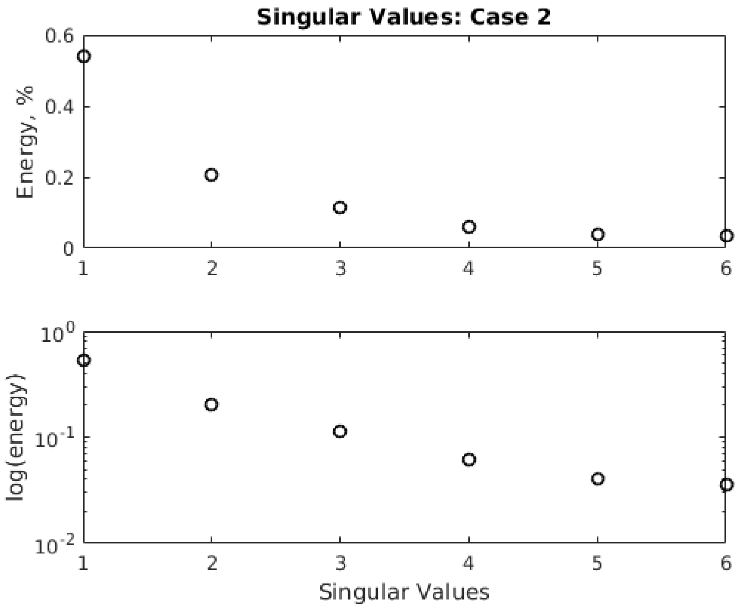

Figure 4: Singular values of the system and their associated energies for case 2.

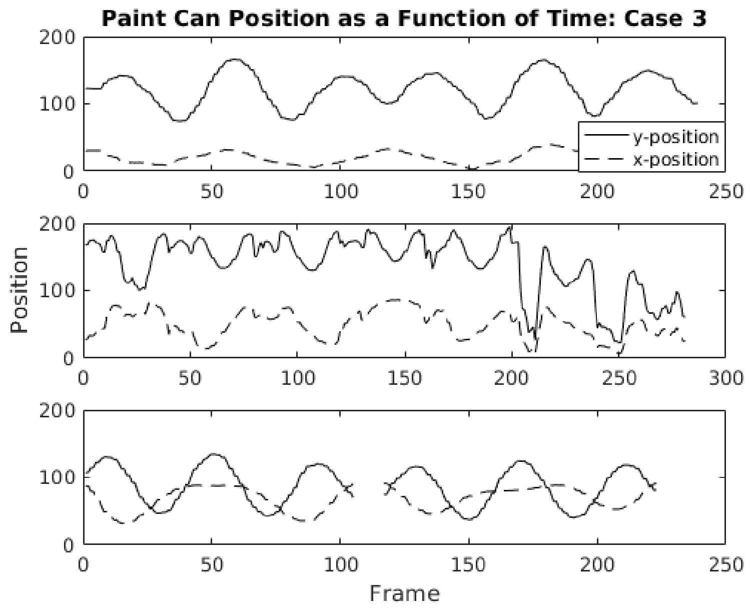

Figure 5: The position of the paint can in case 3 as a function of time as captured by cameras 1, 2, and 3.

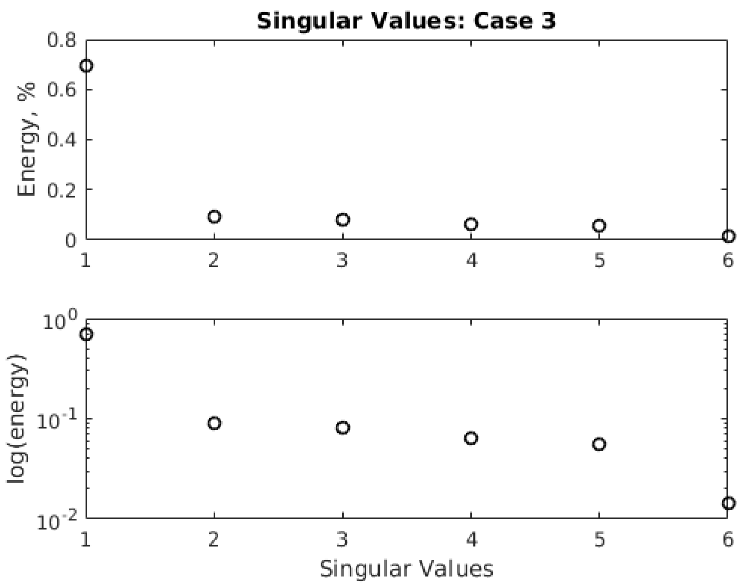

Figure 6: Singular values of the system and their associated energies for case 3.

off-center release (see Figure 5. Surprisingly, as shown in Figure 6, the simple harmonic motion still dominates the singular values of the system. This suggests that the magnitude of the up-down motion was more significant than the side-to-side motion.

Case 4

Case 4 was similar to case 3, however, the tracking algorithm failed to sufficiently capture the pendulum motion. This had a small impact on the singular values for the system. In the interest of space, I have placed the figures for case 4 in Appendix C.

Summary and Conclusions

PCA is a powerful method for reducing data and revealing underlying physical laws of systems. Performing PCA of the oscillating paint can experiment demonstrated the ability of the method to expose important trends in data, allowing me to gain an understanding of the technique. In all cases, PCA overcame both noise and redundancy in the data. It also showed that simply throwing raw data at PCA is fruitless. It was critical in the assignment to know how the data needed to be formatted to extract insights from the PCA results.

References

[1] J.N. Kutz. *Data-Driven Modeling & Scientific Computation: Methods for Complex Systems & Big Data*. OUP Oxford, 2013.

[2] J Dumas. Amath 582 hw1, January 2017.

Appendix A: MATLAB Functions

```
double() -- converts data type to double precision floating point.
find() -- finds indices of points in a matrix with a specified value.
imrotate() -- rotates an image by specified angle.
imcrop() -- crops images to specified size.
load() -- loads saved data.
size() -- returns dimensions of a matrix.
[U,S,V] = svd(A); -- performs SVD.
```

Appendix B: MATLAB Code

The following script represents the code for case 1, but it was changed for each case. For example, I switched the data input and video file names to work on cases 2, 3, and 4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Homework 2 for AMATH 582
% Jesse Dumas
% Paint Can PCA
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all; clc

% load data
load('data/cam1_1.mat');
load('data/cam2_1.mat');
load('data/cam3_1.mat');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAMERA 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% number of frames
n_frames1 = size(vidFrames1_1,4);

% crop video to relavant area
for j=n_frames1:-1:1
    cropVid1_1(:,:,:,j) = imcrop(vidFrames1_1(:,:,:,j), [300 215 90 200]);
```

```
end

% track the can
for j = 1:n_frames1
    j_frame = double(cropVid1_1(:,:,3,j));
    [row_1,col_1] = find(j_frame >= 245);
    row_ave_1(j)=mean(row_1);
    col_ave_1(j)=mean(col_1);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAMERA 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% number of frames
n_frames2 = size(vidFrames2_1,4);

for j=n_frames2:-1:1
    cropVid2_1(:,:,:,j) = imcrop(vidFrames2_1(:,:,:,j), [250 125 90 200]);
end

% track the can
for j = 1:n_frames2
    j_frame = double(cropVid2_1(:,:,3,j));
    [row_2,col_2] = find(j_frame >= 245);
    row_ave_2(j)=mean(row_2);
    col_ave_2(j)=mean(col_2);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAMERA 3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% number of frames
n_frames3 = size(vidFrames3_1,4);

for j=n_frames3:-1:1
    rot_vid(:,:,:,j) = imrotate(vidFrames3_1(:,:,:,j), -90);
    cropVid3_1(:,:,:,j) = imcrop(rot_vid(:,:,:,j), [160 250 90 240]);
end

% track the can
for j = 1:n_frames3
    j_frame = double(cropVid3_1(:,:,3,j));
    [row_3,col_3] = find(j_frame >= 245);
    row_ave_3(j)=mean(row_3);
    col_ave_3(j)=mean(col_3);
end

% make plots
figure(1);

subplot(3,1,1)
min_frames1=min(size(row_ave_1,2));
t1=1:min_frames1;
plot(t1,row_ave_1(1:min_frames1),'k',t1,col_ave_1(1:min_frames1),'k--')
title('Paint Can Position as a Function of Time: Case 1')

subplot(3,1,2)
min_frames2=min(size(row_ave_2,2));
t2=1:min_frames2;
plot(t2,row_ave_2(1:min_frames2),'k',t2,col_ave_2(1:min_frames2),'k--')
ylabel('Position')

subplot(3,1,3)
min_frames3=min(size(row_ave_3,2));
t3=1:min_frames3;
plot(t3,row_ave_3(1:min_frames3),'k',t3,col_ave_3(1:min_frames3),'k--')
xlabel('Frame')

% do SVD
big_data=[row_ave_1(1:200); col_ave_1(1:200); row_ave_2(1:200);
         col_ave_2(1:200); row_ave_3(1:200); col_ave_3(1:200)];
% remove NaNs
big_data(isnan(big_data))=0;

[u, s, v] = svd(big_data, 'econ');

sigma = diag(s);
energy = sigma/sum(sigma);

% make more plots
figure(2)
subplot(2,1,1)
plot(energy, 'ko','LineWidth', [1.4])
title('Singular Values: Case 1')
ylabel('Energy, %')
subplot(2,1,2)
```

```
semilogy(energy, 'ko','LineWidth', [1.4])
ylabel('log(energy)')
xlabel('Singular Values')
```

Appendix C: Additional Figures

To conserve space in the report, additional figures have been placed here.

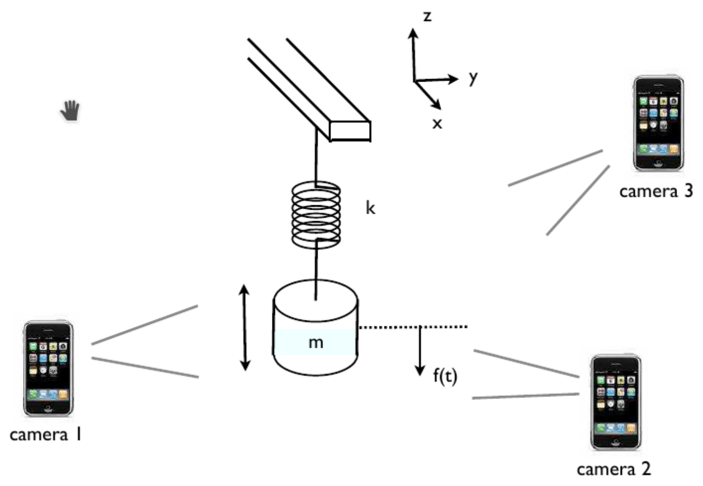

Figure 7: The experimental set up, as illustrated in Kutz.

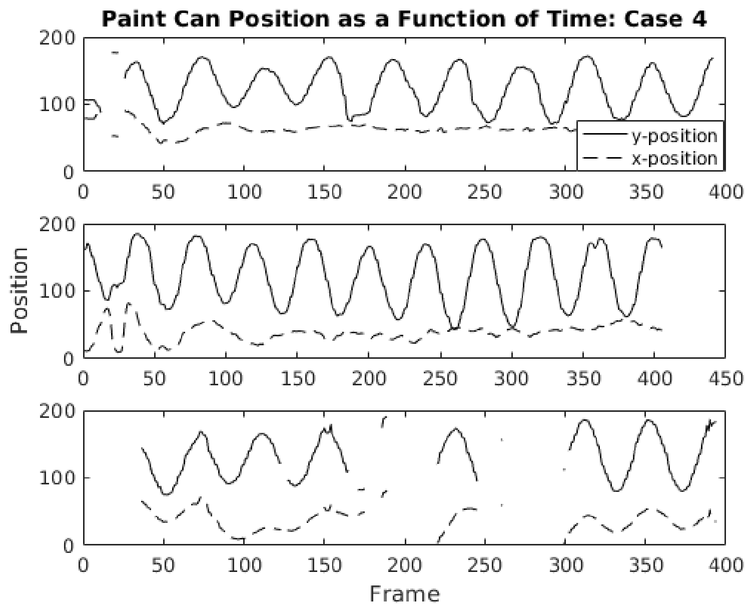

Figure 8: The position of the paint can in case 4 as a function of time as captured by cameras 1, 2, and 3.

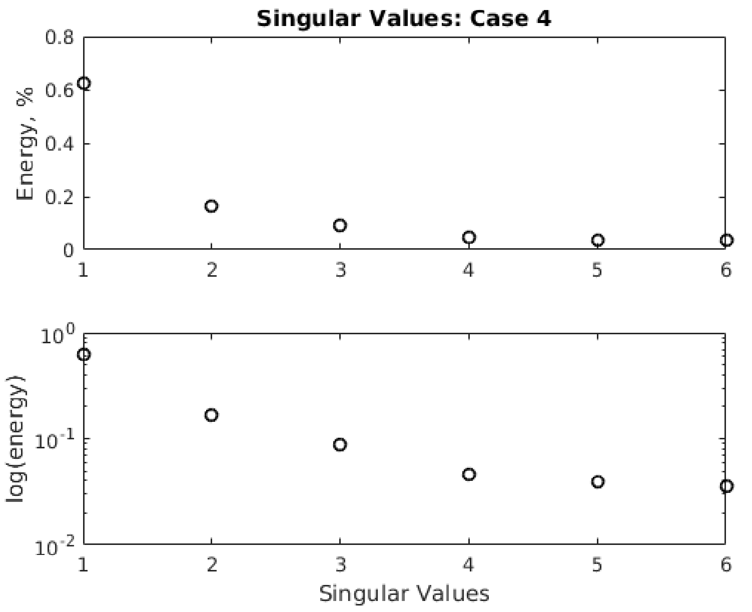

Figure 9: Singular values of the system and their associated energies for case 4.