AMATH 482 Project 3 : Princip

Yixuan Liu

February 2020

1 Abstract

In the real world, people always deal with a large amount of data, which may contain thousands of dimensions. No tool can handle such a large amount of information. The proper way to extract data is by principal component analysis, which can decrease the dimensions of data massively. In this project, we will focus on the application of principal component analysis in the spring-mass system.

2 Introduction and Overview

In this project, we are given four sets of data that contains the motion of the paint: The first set of data is the ideal case, where the paint is released vertically. The subset of this set contains the data record by the camera from different angles. The second set of data is the noisy case where the camera, which records the vertical movement of the paint, is shaking. The third set is the horizontal case, where the paint is released off-center, which leads to the movement in both x-y direction and z-direction. The last set is the rotational case, where not only the paint is release off-center, but also the paint is released with rotation. Our goal is to use PCA to understand the motion of the spring-mass system by reducing the dimensions of data.

3 Theoretical Background

3.1 Singular value decomposition

The image of an n-dimensional unit-sphere can be transformed by a matrix $A \in R^{mxn}$, where m $\geq$ n, to an m-dimensional hyper-ellipse, which has principal semiaxis with length $\sigma_1, \sigma_2, ...\sigma_n$, pointing in direction $\overrightarrow{u_1}, \overrightarrow{u_2}, ...\overrightarrow{u_n}$ for $\overrightarrow{v_1}, \overrightarrow{v_2}, ...\overrightarrow{v_n}$ such that

$$A\overrightarrow{V} = \hat{U}\hat{\Sigma} \tag{1}$$

Where $\overrightarrow{V}$ is a matrix where columns are orthonormal basis. Then, we have

$$A = \hat{U}\hat{\Sigma}\vec{V^{-1}} \tag{2}$$

One useful aspect of SVD is that we can diagonalize any matrix with proper bases(see PCA section for more details)

3.2 Principal Component Analysis

The PCA is commonly used to reduce the dimension of data. To begin with, we first should normalize our data into X:

$$X = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix} \tag{3}$$

Where X is mxn matrix and each row(single measurement) is normalized by subtracting it's mean. Then, we define that

$$C_X = \frac{1}{n-1} X X^T \tag{4}$$

Where C_x is a square, symmetric m by m matrix. The diagonal of C_x are the variance of each measurement and off-diagonal are covariance of two measurements. If the covariance is high, we can conclude that two variables are fairly dependent, which means we can eliminate a lots of redundant information. If the variance is high, we can conclude that the information carried by that measurement are very important. On the contrary, the low value of covariance shows the Independence between two measurements and the low value of variance shows the insignificance of the measurement. To see the connection between SVD and PCA, we should understand the concept of diagonalization: there exists an ideal basis in which C_x can be written. In this basis, all redundancies have been removed and the variance of each measurement are ordered from largest to smallest. We can show that

$$Y = U^{-1} X \tag{5}$$

$$C_Y = \frac{1}{n-1} \Sigma^2 \tag{6}$$

Where $X = \hat{U}\hat{\Sigma}\vec{V^{-1}}$ and Y is the projection of X onto U and is the principal component. Also, we can compute energy level of each principal component by dividing it's corresponding σ^2 with sum of diag(Σ^2).

4 Algorithm Implementation and Development

4.1 Loading data

The most challenge part is how to track the position of the paint. First, I manually select the range of pixels where the paint moves in each video, then turn the environment to dark

outside that range. The white part and the flash light of the paint provide us the brightest area in that range. Then, we go through frame by frame, finding the brightest pixel in that frame. Because the paint have area, which means we need multiple pixels to represent the object. Then, we choose all pixels which have around 90% of the max brightness. This area should contain the paint. After that, we average the x,y value to get a representation of location of the center of the paint.(treat the paint as a point). We save these values into px,py vectors. For each set of data, we have three set of px,py which has different length. In order to do SVD, we need construct a X matrix as same as X in the theoretical Background. To do so, we need make sure each px,px2,px3,py,py2,py3 have same length. Then, I find the first peak in y direction and cut off all data before that peak. Finally, I find the minimum length among those 6 vectors and cut the data after that length. Then, we can put all these vectors into a matrix. Also, for the last two cases, we have horizontal oscillations. In order to align all vectors, I manually watch the video and find the frame where three objects filmed by different cameras are approximately at same location. Also, the third camera for each case flip the x-y coordinate. In order to make data coherent, I manually change the px,py for third camera.

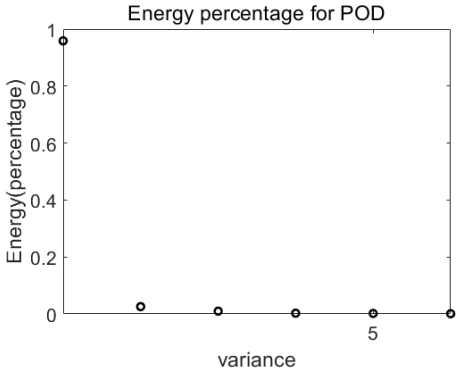
4.2 Principal Component Analysis

We first subtract mean from each row in X to normalize X. Then, we do SVD on X. To compute the energy level, we divide sigma with sum of sigma to see the proportion. Also, we plot the X projection onto the U(principal component) to see where the motion happens.

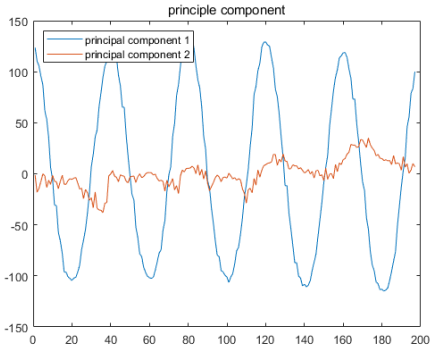
I simply do the same process for every case with varying some parameters.

5 Computational Results

5.1 ideal case



(a) Figure 1 : Energy level for each principal component



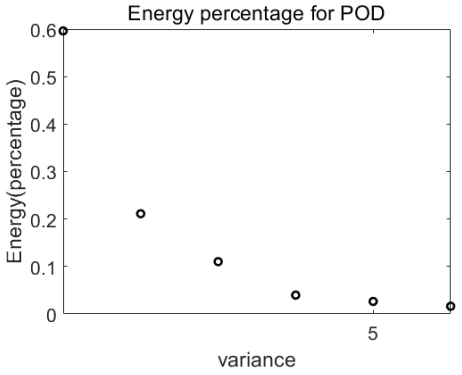
(b) Figure 2 : main principal component

In the ideal case, we should expect a one-dimensional motion. In figure 1, we notice that the energy for the first principal component is dominant over all other principal components.

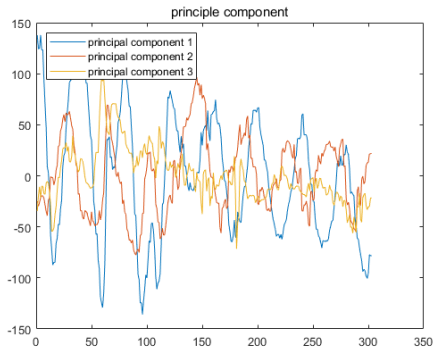
Also, In figure 2, we notice the oscillation in principal component 1 and not much oscillation in principal component 2. We can conclude that indeed the paint is in the simple harmonic oscillations in one dimension. The PCA works pretty well in this situation.

5.2 Shaking Case

In the shaking case, though the main motion of paint is still simple harmonic oscillation, there are some other motions in different direction due to the noise. We should not expect the dominant of one principal component in PCA. By reading the figure 3, we figure out that though the first principal component has the largest energy level, the second and third also has relatively large energy level. Consequently, we can see different oscillations happened in different principal component basis, which means the motion is not in one direction or one dimension. However, we notice that the first three component dominant over others, as a result, we can conclude that at least our paint does not move in six dimension.

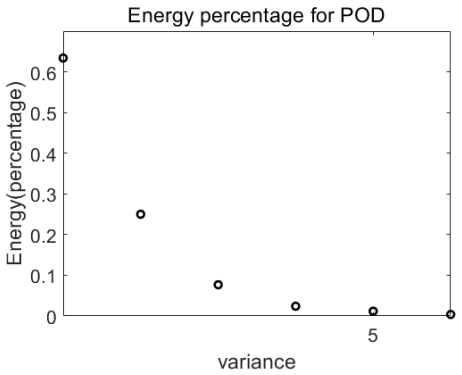


(a) Figure 3 : Energy level for each principal component

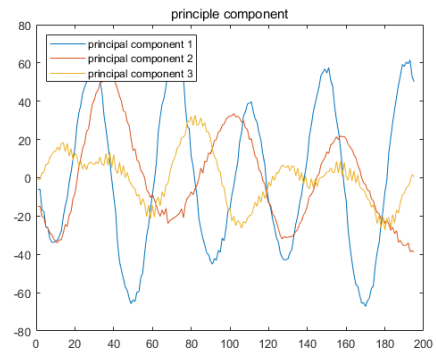


(b) Figure 4 : main principal component

5.3 Horizontal Case



(a) Figure 5 : Energy level for each principal component

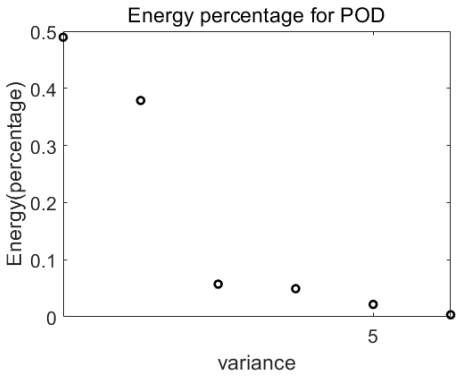


(b) Figure 6 : main principal component

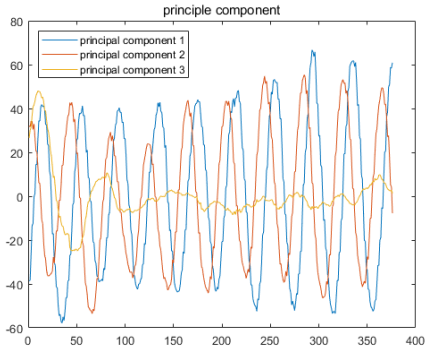
In the horizontal case, the paint is moving in both z and x-y direction(harmonic oscillation and pendulum motion), which means we expect two dominant principal component from PCA. From figure 5, we notice that the sum of the energy of first two principal component dominant over others. The corresponding principal component in the figure 6 confirmed that there are some oscillations in principal component 1 and principal component 2. The third component may be due to some measurement's error.

5.4 Rotational Case

In the rotational case, the paint is not only moving in both z and x-y direction but also have a rotational motion. Due to the complexity of the system, I am doubt about the ability of PCA. In figure 7, we see that the first two dominant over others, which represent the horizontal and vertical motions. We can also conclude the same result from figure 8. However, there is no evidence show that paint is rotating. In this case, PCA does not work well.



(a) Figure 7 : Energy level for each principal component



(b) Figure 8 : main principal component

6 Conclusion

We notice that the PCA works decent in the ideal and horizontal case. Though the noise, we can tell something from the PCA in the noise case: the dimension of system can be reduced. However, I think PCA failed in the fourth case is due to the fact that we does not have ability to represent the rotation in our data, which reveal the disadvantage of PCA: it requires sophisticated data. We should always think about this disadvantage when we apply PCA.

7 Appendix A: MATLAB functions used and brief implementation explanation

- rgb2gray: change the color x to only black and white

- [xindex,yindex] = find(a>b): find all index in a where its corresponding value is greater than b
- SVD(X):decompose X in U, V and Σ

8 Appendix B: MATLAB codes

```
%% simple case
load('cam1_1.mat')
load('cam2_1.mat')
load('cam3_1.mat')
numFrames = size(vidFrames1_1,4);
py=zeros(1,numFrames);
px=zeros(1,numFrames);
for j = 1:numFrames
    X= vidFrames1_1(:,:,:,j);
    X_dgray = double(rgb2gray(X));%we can handle double data in black and white
        color domain
    X_dgray(:,1:304)=0;%eliminate the range that does not contain the mass
    X_dgray(:,388:end)=0;
    X_dgray([1:169 425:end],305:387)=0;
    Mx = max(X_dgray(:));
    [yrange,xrange] = find(X_dgray>0.95*Mx);
    py(j)=mean(yrange);
    px(j)=mean(xrange);
end
[~,idx] = max(py(1:30));
py=py(idx:end);
px=px(idx:end);

numFrames2 = size(vidFrames2_1,4);
py2=zeros(1,numFrames2);
px2=zeros(1,numFrames2);
for j = 1:numFrames2
    X2= vidFrames2_1(:,:,:,j);
    X2_dgray = double(rgb2gray(X2));%we can handle double data in black and white
        color domain
    X2_dgray(:,1:241)=0;%eliminate the range that does not contain the mass
    X2_dgray(:,369:end)=0;
    X2_dgray([1:97 371:end],242:368)=0;
    Mx = max(X2_dgray(:));
    [yrange,xrange] = find(X2_dgray>0.95*Mx);
    py2(j)=mean(yrange);
    px2(j)=mean(xrange);
end
[~,idx] = max(py2(1:30));
```

```
py2=py2(idx:end);
px2=px2(idx:end);

numFrames3 = size(vidFrames3_1,4);
py3=zeros(1,numFrames3);
px3=zeros(1,numFrames3);
for j = 1:numFrames3
    X3= vidFrames3_1(:,:,:,j);
    X3_dgray = double(rgb2gray(X3));%we can handle double data in black and white
        color domain
    X3_dgray(:,1:224)=0;%eliminate the range that does not contain the mass
    X3_dgray(:,507:end)=0;
    X3_dgray([1:212 357:end],225:506)=0;
    Mx = max(X3_dgray(:));
    [yrange,xrange] = find(X3_dgray>0.95*Mx);
    py3(j)=mean(xrange); %reverse the coordinate due to the camera angle
    px3(j)=mean(yrange);
end
[~,idx] = max(py3(1:30));
py3=py3(idx:end);
px3=px3(idx:end);

cut=min([length(py) length(py2) length(py3)]);
py=py(1:cut)-mean(py(1:cut));
px=px(1:cut)-mean(px(1:cut));
py2=py2(1:cut)-mean(py2(1:cut));
px2=px2(1:cut)-mean(px2(1:cut));
py3=py3(1:cut)-mean(py3(1:cut));
px3=px3(1:cut)-mean(px3(1:cut));
X=[px;py;px2;py2;px3;py3];
[u,s,~]=svd(X); % perform the SVD
lambda=diag(s).^2; % produce diagonal variances
Y=u'*X; % produce the principal components projection
figure(1)
plot(lambda/sum(lambda),'ko','Linewidth',2);
ylabel('Energy(percentage)')
xlabel('variance')
title('Energy percentage for POD','Fontsize',16)
set(gca,'Fontsize',16,'Xtick',0:5:25)
figure(2)
plot(Y(1,:)); hold on
plot(Y(2,:))
title('principle component','Fontsize',16)

set(gca,'Fontsize',10)
legend('principal component 1','principal component 2','Location','northwest')
```

```
%%
clc;clear all;close all
load('cam1_2.mat')
load('cam2_2.mat')
load('cam3_2.mat')
numFrames = size(vidFrames1_2,4);
py=zeros(1,numFrames);
px=zeros(1,numFrames);
for j = 1:numFrames
    X= vidFrames1_2(:,:,:,j);
    X_dgray = double(rgb2gray(X));%we can handle double data in black and white
        color domain
    X_dgray(:,1:290)=0;%eliminate the range that does not contain the mass
    X_dgray(:,431:end)=0;
    X_dgray([1:203 427:end],291:430)=0;
    Mx = max(X_dgray(:));
    [yrange,xrange] = find(X_dgray>0.95*Mx);
    py(j)=mean(yrange);
    px(j)=mean(xrange);
end
[~,idx] = max(py(1:40));
py=py(idx:end);
px=px(idx:end);

numFrames2 = size(vidFrames2_2,4);
py2=zeros(1,numFrames2);
px2=zeros(1,numFrames2);
for j = 1:numFrames2
    X2= vidFrames2_2(:,:,:,j);
    X2_dgray = double(rgb2gray(X2));%we can handle double data in black and white
        color domain
    X2_dgray(:,[1:215 448:end])=0;%eliminate the range that does not contain the
        mass
    X2_dgray([1:61 417:end],216:447)=0;
    Mx = max(X2_dgray(:));
    [yrange,xrange] = find(X2_dgray>0.95*Mx);
    py2(j)=mean(yrange);
    px2(j)=mean(xrange);
end

[~,idx] = max(py2(1:40));
py2=py2(idx:end);
px2=px2(idx:end);

numFrames3 = size(vidFrames3_2,4);
py3=zeros(1,numFrames3);
px3=zeros(1,numFrames3);
```

```
for j = 1:numFrames3
    X3= vidFrames3_2(:,:,:,j);
    X3_dgray = double(rgb2gray(X3));%we can handle double data in black and white
        color domain
    X3_dgray(:,[1:241 514:end])=0;%eliminate the range that does not contain the
        mass
    X3_dgray([1:189 345:end],242:513)=0;
    Mx = max(X3_dgray(:));
    [yrange,xrange] = find(X3_dgray>0.95*Mx);
    py3(j)=mean(xrange); %reverse the coordinate due to the camera angle
    px3(j)=mean(yrange);
end
[~,idx] = max(py3(1:40));
py3=py3(idx:end);
px3=px3(idx:end);

cut=min([length(py) length(py2) length(py3)]);
py=py(1:cut)-mean(py(1:cut));
px=px(1:cut)-mean(px(1:cut));
py2=py2(1:cut)-mean(py2(1:cut));
px2=px2(1:cut)-mean(px2(1:cut));
py3=py3(1:cut)-mean(py3(1:cut));
px3=px3(1:cut)-mean(px3(1:cut));
X=[px;py;px2;py2;px3;py3];
[u,s,~]=svd(X); % perform the SVD
lambda=diag(s).^2; % produce diagonal variances
Y=u'*X; % produce the principal components projection
figure(1)
plot(lambda/sum(lambda),'ko','Linewidth',2);
ylabel('Energy(percentage)')
xlabel('variance')
title('Energy percentage for POD','Fontsize',16)
set(gca,'Fontsize',16,'Xtick',0:5:25)
figure(2)
plot(Y(1,:)); hold on
plot(Y(2,:))
plot(Y(3,:))
title('principle component','Fontsize',16)

set(gca,'Fontsize',10)
legend('principal component 1','principal component 2','principal component
    3','Location','northwest')
%%
clc;clear all;close all
load('cam1_3.mat')
load('cam2_3.mat')
load('cam3_3.mat')
```

```
numFrames = size(vidFrames1_3,4);
py=zeros(1,numFrames);
px=zeros(1,numFrames);
for j = 1:numFrames
    X= vidFrames1_3(:,:,:,j);
    X_dgray = double(rgb2gray(X));%we can handle double data in black and white
        color domain
    X_dgray(:,[1:282 415:end])=0;%eliminate the range that does not contain the
        mass
    X_dgray([1:225 422:end],283:414)=0;
    Mx = max(X_dgray(:));
    [yrange,xrange] = find(X_dgray>0.9*Mx);
    py(j)=mean(yrange);
    px(j)=mean(xrange);

end

py=py(8:end);%manully allign
px=px(8:end);

numFrames2 = size(vidFrames2_3,4);
py2=zeros(1,numFrames2);
px2=zeros(1,numFrames2);
for j = 1:numFrames2
    X2= vidFrames2_3(:,:,:,j);
    X2_dgray = double(rgb2gray(X2));%we can handle double data in black and white
        color domain
    X2_dgray(:,[1:220 446:end])=0;%eliminate the range that does not contain the
        mass
    X2_dgray([1:134 429:end],221:445)=0;
    Mx = max(X2_dgray(:));
    [yrange,xrange] = find(X2_dgray>0.9*Mx);
    py2(j)=mean(yrange);
    px2(j)=mean(xrange);

end

py2=py2(16:end);
px2=px2(16:end);

numFrames3 = size(vidFrames3_3,4);
py3=zeros(1,numFrames3);
px3=zeros(1,numFrames3);
for j = 1:numFrames3
    X3= vidFrames3_3(:,:,:,j);
    X3_dgray = double(rgb2gray(X3));%we can handle double data in black and white
        color domain
```

```
    X3_dgray(:,[1:246 519:end])=0;%eliminate the range that does not contain the
        mass
    X3_dgray([1:145 361:end],247:518)=0;
    Mx = max(X3_dgray(:));
    [yrange,xrange] = find(X3_dgray>0.9*Mx);
    py3(j)=mean(xrange); %reverse the coordinate due to the camera angle
    px3(j)=mean(yrange);
end

py3=py3(43:end);
px3=px3(43:end);

cut=min([length(py) length(py2) length(py3)]);
py=py(1:cut)-mean(py(1:cut));
px=px(1:cut)-mean(px(1:cut));
py2=py2(1:cut)-mean(py2(1:cut));
px2=px2(1:cut)-mean(px2(1:cut));
py3=py3(1:cut)-mean(py3(1:cut));
px3=px3(1:cut)-mean(px3(1:cut));
X=[px;py;px2;py2;px3;py3];
[u,s,~]=svd(X/sqrt(cut-1)); % perform the SVD
lambda=diag(s).^2; % produce diagonal variances
Y=u'*X; % produce the principal components projection
figure(1)
plot(lambda/sum(lambda),'ko','Linewidth',2);
ylabel('Energy(percentage)')
xlabel('variance')
title('Energy percentage for POD','Fontsize',16)
set(gca,'Fontsize',16,'Xtick',0:5:25)
figure(2)
plot(Y(1,:)); hold on
plot(Y(2,:))
plot(Y(3,:))
title('principle component','Fontsize',16)

set(gca,'Fontsize',10)
legend('principal component 1','principal component 2','principal component
    3','Location','northwest')


%%
clc;clear all;close all
load('cam1_4.mat')
load('cam2_4.mat')
load('cam3_4.mat')
numFrames = size(vidFrames1_4,4);
py=zeros(1,numFrames);
```

```
px=zeros(1,numFrames);
for j = 1:numFrames
    X= vidFrames1_4(:,:,:,j);
    X_dgray = double(rgb2gray(X));%we can handle double data in black and white
        color domain
    X_dgray(:,[1:304 461:end])=0;%eliminate the range that does not contain the
        mass
    X_dgray([1:216 420:end],305:460)=0;
    Mx = max(X_dgray(:));
    [yrange,xrange] = find(X_dgray>0.9*Mx);
    py(j)=mean(yrange);
    px(j)=mean(xrange);

end

py=py(11:end);%manully allign
px=px(11:end);

numFrames2 = size(vidFrames2_4,4);
py2=zeros(1,numFrames2);
px2=zeros(1,numFrames2);
for j = 1:numFrames2
    X2= vidFrames2_4(:,:,:,j);
    X2_dgray = double(rgb2gray(X2));%we can handle double data in black and white
        color domain
    X2_dgray(:,[1:220 418:end])=0;%eliminate the range that does not contain the
        mass
    X2_dgray([1:70 407:end],221:417)=0;
    Mx = max(X2_dgray(:));
    [yrange,xrange] = find(X2_dgray>0.9*Mx);
    py2(j)=mean(yrange);
    px2(j)=mean(xrange);

end
py2=py2(11:end);
px2=px2(11:end);

numFrames3 = size(vidFrames3_4,4);
py3=zeros(1,numFrames3);
px3=zeros(1,numFrames3);
for j = 1:numFrames3
    X3= vidFrames3_4(:,:,:,j);
    X3_dgray = double(rgb2gray(X3));%we can handle double data in black and white
        color domain
    X3_dgray(:,[1:265 533:end])=0;%eliminate the range that does not contain the
        mass
    X3_dgray([1:131 276:end],266:532)=0;
```

```
    Mx = max(X3_dgray(:));
    [yrange,xrange] = find(X3_dgray>0.9*Mx);
    py3(j)=mean(xrange); %reverse the coordinate due to the camera angle
    px3(j)=mean(yrange);
end

py3=py3(19:end);
px3=px3(19:end);

cut=min([length(py) length(py2) length(py3)]);
py=py(1:cut)-mean(py(1:cut));
px=px(1:cut)-mean(px(1:cut));
py2=py2(1:cut)-mean(py2(1:cut));
px2=px2(1:cut)-mean(px2(1:cut));
py3=py3(1:cut)-mean(py3(1:cut));
px3=px3(1:cut)-mean(px3(1:cut));
X=[px;py;px2;py2;px3;py3];
[u,s,~]=svd(X/sqrt(cut-1)); % perform the SVD
lambda=diag(s).^2; % produce diagonal variances
Y=u'*X; % produce the principal components projection
figure(1)
plot(lambda/sum(lambda),'ko','Linewidth',2);
ylabel('Energy(percentage)')
xlabel('variance')
title('Energy percentage for POD','Fontsize',16)
set(gca,'Fontsize',16,'Xtick',0:5:25)
figure(2)
plot(Y(1,:)); hold on
plot(Y(2,:))
plot(Y(3,:))
title('principle component','Fontsize',16)
set(gca,'Fontsize',10)
legend('principal component 1','principal component 2','principal component
    3','Location','northwest')
```