18-698 / 42-632
Neural Signal Processing
Problem Set 8 Solutions

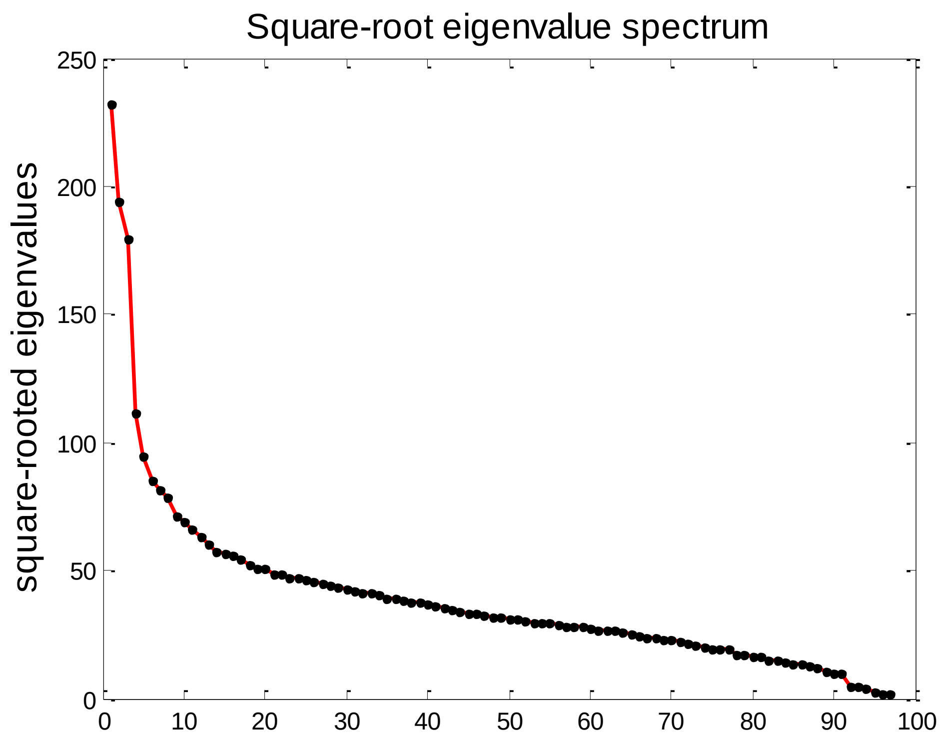
1.) Code:

```
%% 18699: Neural Signal Processing
%% Problem set 8 solutions
clear all; close all; clc;
%% Problem 1a
load ps8_data.mat
mu = mean(Xplan);
Xplan_centered = Xplan-repmat(mu,size(Xplan,1),1);
[U,D] = eig(Xplan_centered'*Xplan_centered);
[eigenvals, indx] = sort(diag(D),'descend');
% arrange the eigenvectors according to the magnitude of the
eigenvalues
U = U(:,indx);
% plot square-root eigenvalue spectrum
figure
plot(sqrt(eigenvals),'r','linewidth',2)
hold on
plot(sqrt(eigenvals),'k.','linewidth',2)
title('Square-root eigenvalue spectrum','fontsize',14);
ylabel('square-rooted eigenvalues','fontsize',14);
set(gcf,'papersize',[5 4],'paperposition',[0 0 5 4]);
print(gcf,'ps8_fig1.eps','-depsc');
% There is an elbow after the 3rd dominant eigenvalue. The top three
% eigenvectors explain 44.8% of the data variance.
sum((eigenvals(1:3)))/sum((eigenvals))
%% Problem 1b
figure
markers = {'r.','k.','y.','g.','b.','m.','c.','k*'};
num_reaches_per_angle = 91;
for reachAngle = 1:8
indx = (1:num_reaches_per_angle)+(reachAngle-1)*num_reaches_per_angle;
X = Xplan(indx,:);
plot3(X*U(:,1),X*U(:,2),X*U(:,3),markers{reachAngle});
hold on
end
xlabel('PC 1','fontsize',14)
ylabel('PC 2','fontsize',14)
zlabel('PC 3','fontsize',14)
title('Reach data projected into three-dimensional PC
space','fontsize',14)
fprintf('Rotate until it looks nice...')
% pause
% set(gcf,'papersize',[5 4],'paperposition',[0 0 5 4]);
% print(gcf,'ps8_fig2.eps','-depsc');
fprintf('done\n');
%% Problem 1c
figure
imagesc(U(:,1:3)');
colorbar
xlabel('Neuron #','fontsize',14)
```

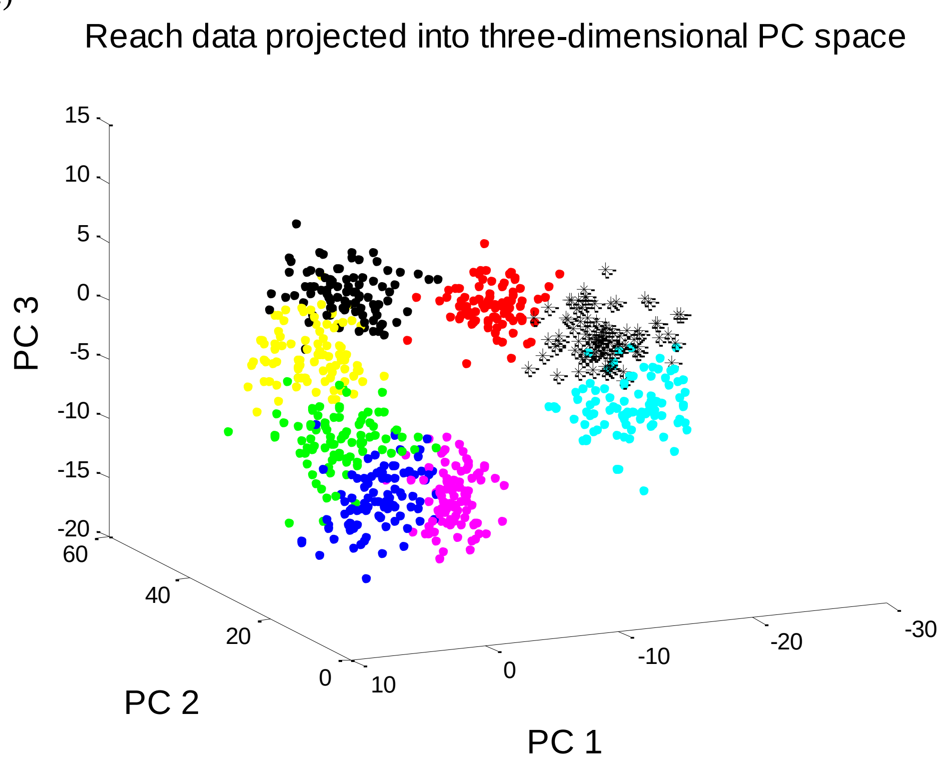
```
ylabel('Principal component #','fontsize',14)
title('Heat map of top three principal component
directions','fontsize',14);
set(gcf, 'renderer', 'OpenGL');
set(gcf,'papersize',[5 3],'paperposition',[0 0 5 3]);
print(gcf,'ps8_fig3.eps','-depsc');
```

a.)

There appears to be an elbow after the 3rd dominant eigenvalue. The top three eigenvectors explain 44.79 percent of the data variance.

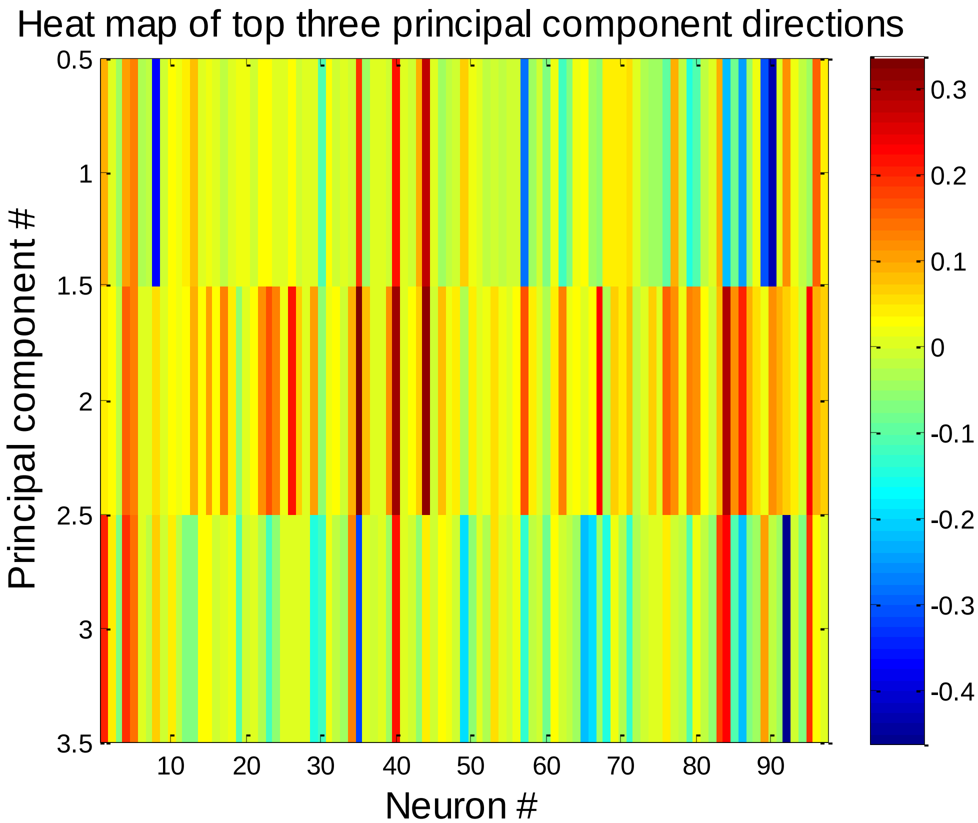


b.)



c.)

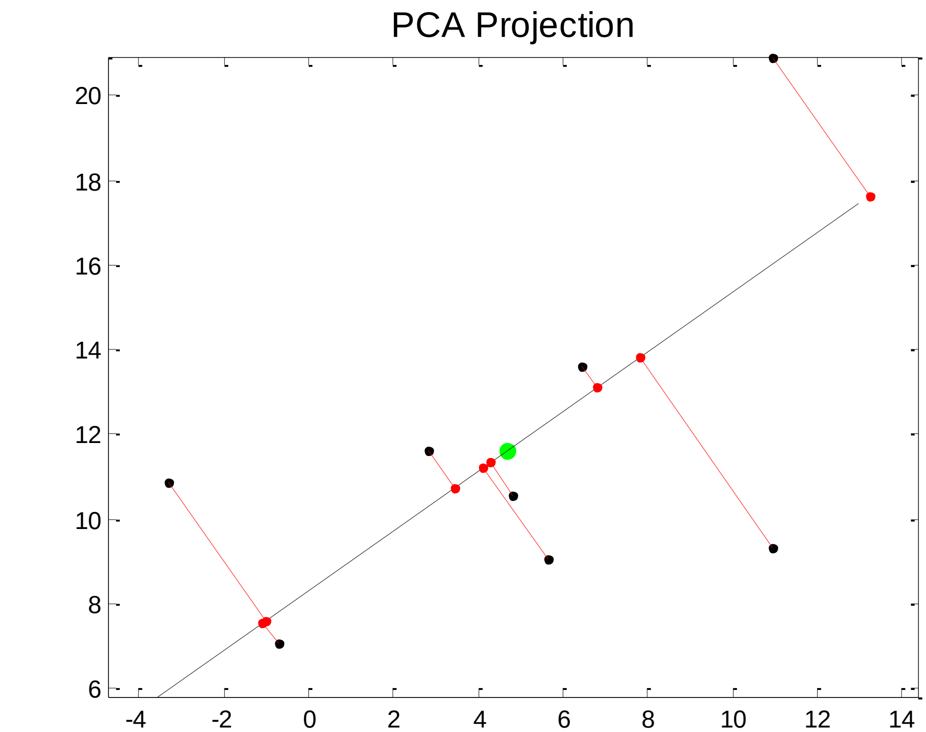
Most elements of the top three eigenvectors are relatively small, with absolute values less than 0.15, indicating that those neurons contibute relatively little to the corresponding principal component. The relatively large elements of a particular eigenvector, with absolute values greater than 0.15, indicate clusters of neurons working together along the direction defined by the eigenvector.



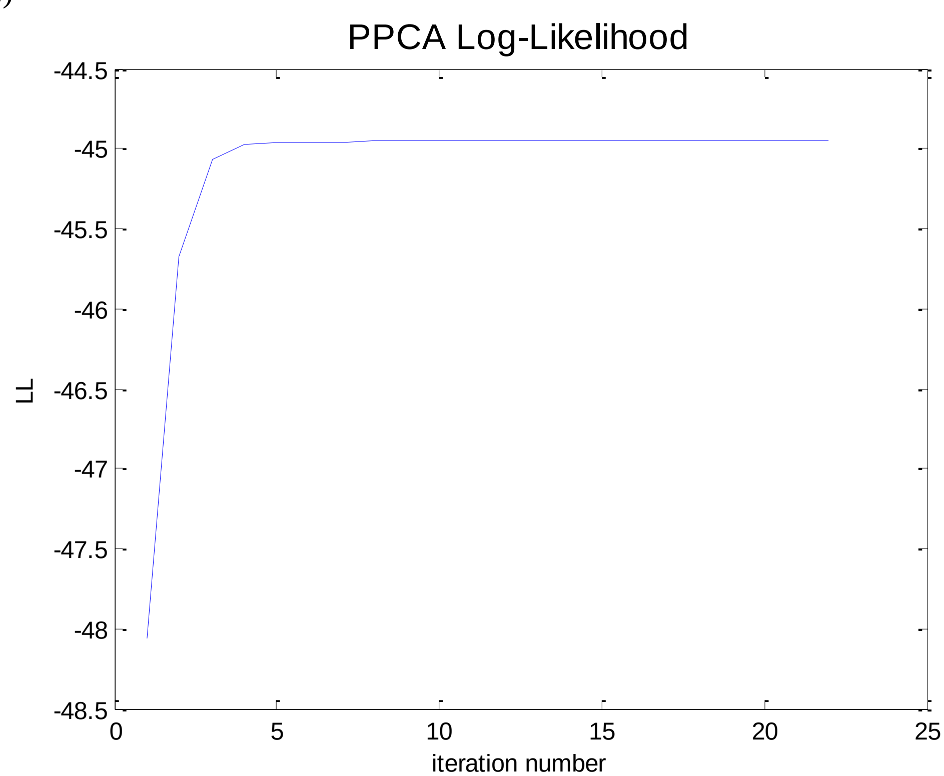
2.) Code:

```
%% Problem 2a
[n,p] = size(Xsim);
% Peform PCA
mu = mean(Xsim);
Xsim_centered = Xsim-repmat(mu,n,1);
[U,D] = eig(Xsim_centered'*Xsim_centered);
[eigenvals, indx] = sort(diag(D),'descend');
% arrange the evectors according to the magnitude of the eigenvalues
U = U(:,indx);
% find the projection of the data onto PC1
z_hat_PCA = Xsim_centered*U(:,1);
% construct the plot described in ps8
figure
dim_reduce_plot(Xsim,z_hat_PCA,U(:,1));
title('PCA Projection','fontsize',14)
set(gcf,'papersize',[5 4],'paperposition',[0 0 5 4]);
print(gcf,'ps8_fig4.eps','-depsc');
%% Problem 2b: PPCA implementation is in fastfa.m,
% written by Zoubin Gharhamani, updated by Byron Yu.
%% Problem 2c
[estParamsPPCA,LL_PPCA] = fastfa(Xsim',1,'ppca');
MU = repmat(estParamsPPCA.d,1,n);
C = estParamsPPCA.L*estParamsPPCA.L'+diag(estParamsPPCA.Ph)
z_hat_PPCA = (estParamsPPCA.L'*C^(-1)*(Xsim'-MU))';
figure
dim_reduce_plot(Xsim,z_hat_PPCA,estParamsPPCA.L)
title('PPCA Projection','fontsize',14)
set(gcf,'papersize',[5 4],'paperposition',[0 0 5 4]);
print(gcf,'ps8_fig6.eps','-depsc');
figure
plot(LL_PPCA);
xlabel('iteration number')
ylabel('LL')
set(gcf,'papersize',[5 4],'paperposition',[0 0 5 4]);
title('PPCA Log-Likelihood','fontsize',14)
print(gcf,'ps8_fig5.eps','-depsc');
%% Problem 2d: PPCA implementation is in fastfa.m,
% written by Zoubin Gharhamani, updated by Byron Yu.
%% Problem 2e
[estParamsFA,LL_FA] = fastfa(Xsim',1,'fa');
C = estParamsFA.L*estParamsFA.L'+diag(estParamsFA.Ph)
z_hat_FA = (estParamsFA.L'*C^(-1)*(Xsim'-MU))';
figure
dim_reduce_plot(Xsim,z_hat_FA,estParamsFA.L)
title('FA Projection','fontsize',14)
set(gcf,'papersize',[5 4],'paperposition',[0 0 5 4]);
print(gcf,'ps8_fig8.eps','-depsc');
figure
plot(LL_FA);
set(gcf,'papersize',[5 4],'paperposition',[0 0 5 4]);
title('FA Log-Likelihood','fontsize',14)
xlabel('iteration number')
ylabel('LL')
print(gcf,'ps8_fig7.eps','-depsc');
```

a.)


PCA Projection
20
18
16
14
12
10
8
6
-4
-2
0
2
4
6
8
10
12
14

b.)


PPCA Log-Likelihood
-44.5
-45
-45.5
-46
-46.5
-47
-47.5
-48
-48.5
LL
0
5
10
15
20
25
iteration number

c.)

The sample covariance approximated by PPCA is:

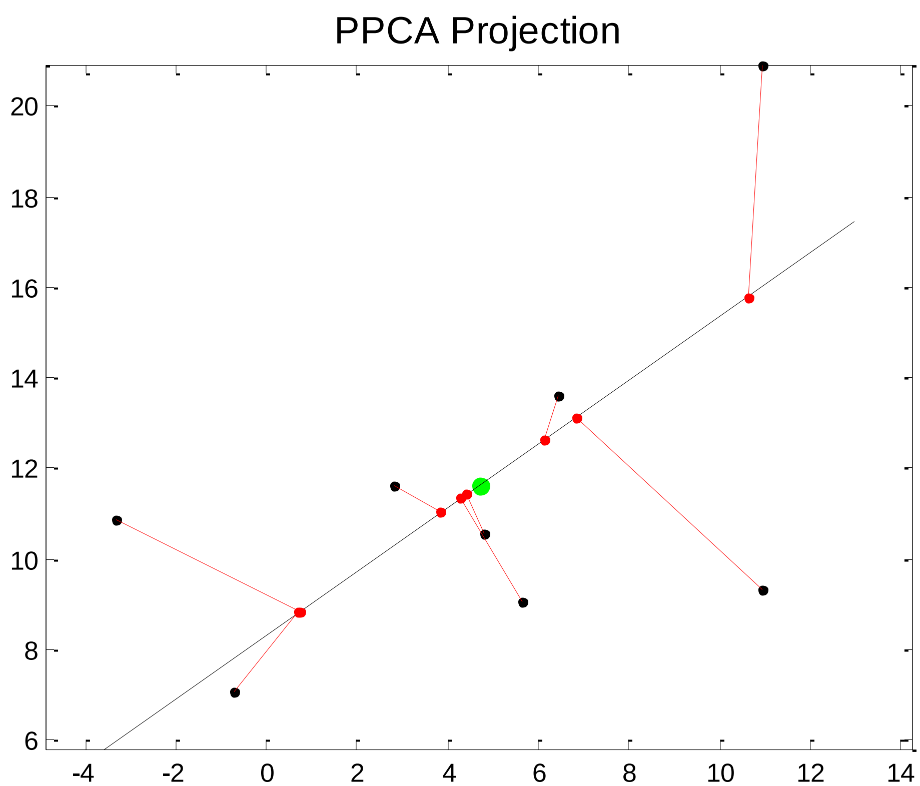
$$WW^T + \sigma^2 I = \begin{bmatrix} 22.4310 & 9.4394 \\ 9.4394 & 15.5921 \end{bmatrix}$$

This is very close to the empirical data covariance:

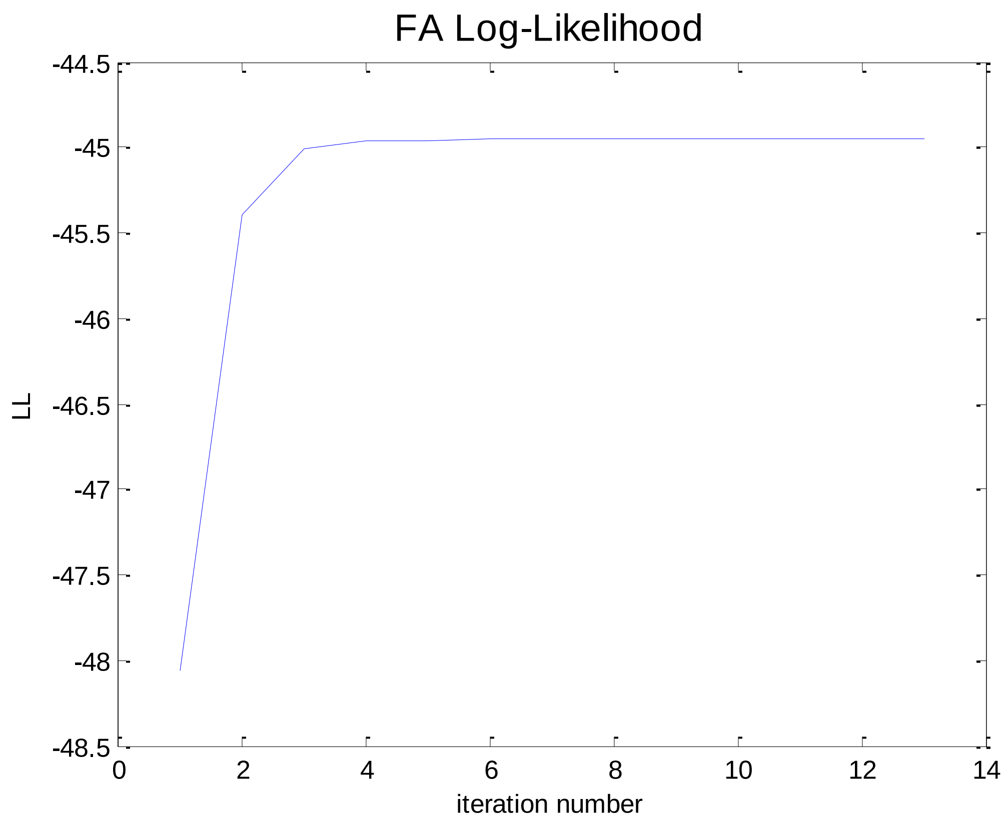
$$\Sigma = \begin{bmatrix} 22.4314 & 9.4398 \\ 9.4398 & 15.5921 \end{bmatrix}$$

d.)

The red lines are no longer orthogonal to the PC space. This is because each point is being projected to a location closer to the mean than in PCA. This is because, in PPCA, some of the points' deviations from the mean are attributed to observation noise, unlike in PCA.



e.)



f.)

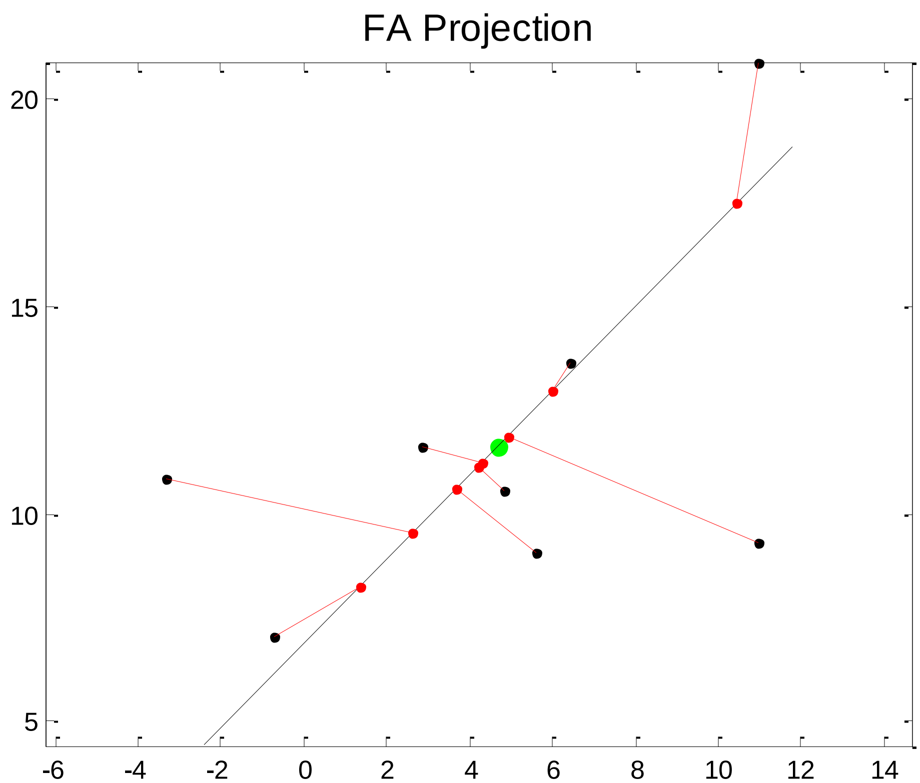
The sample covariance approximated by FA is:

$$WW^T + \Psi = \begin{bmatrix} 22.4315 & 9.4400 \\ 9.4400 & 15.5924 \end{bmatrix}$$

This is very close to the empirircal data covariance:

$$\Sigma = \begin{bmatrix} 22.4314 & 9.4398 \\ 9.4398 & 15.5922 \end{bmatrix}$$

g.)



Functions Used:

```
function dim_reduce_plot(x,z_hat,u1)
% INPUTS:
% x: high dimensional data, each row is a data point
% z_hat: low dimensional data, each row is a data point
% u1: principle direction / factor loading
[n,p] = size(x);
mu = mean(x);
% plot the data
plot(x(:,1),x(:,2),'k.')
hold on
% plot the mean
plot(mu(1),mu(2),'g.','markersize',20)
scale = 2*max(std(x))/norm(u1);
% plot the line defined by the first principal component
xVals = scale*[-u1(1) u1(1)]; % +/- 2 standard deviation
yVals = scale*[-u1(2) u1(2)]; % +/- 2 standard deviation
line(mu(1)+xVals,mu(2)+yVals,'color','k')
% plot the original data projected down into the subspace defined by PC
1
dataPC1 = repmat(mu,n,1) + repmat(u1',n,1).*repmat(z_hat,1,p);
plot(dataPC1(:,1),dataPC1(:,2),'r.')
for i=1:n
      xVals = [x(i,1) dataPC1(i,1)];
      yVals = [x(i,2) dataPC1(i,2)];
      line(xVals,yVals,'color','r');
end
axis equal
end
```

```
function [estParams, LL] = fastfa(X, zDim, typ)
%
% [estParams, LL] = fastfa(X, zDim, ...)
%
% Factor analysis and probabilistic PCA.
%
% xDim: data dimensionality
% zDim: latent dimensionality
% N: number of data points
%
% INPUTS:
%
% X - data matrix (xDim x N)
% zDim - number of factors
% typ - 'fa' or 'ppca'
%
% OUTPUTS:
%
% estParams.L - factor loadings (xDim x zDim)
% estParams.Ph - diagonal of uniqueness matrix (xDim x 1)
% estParams.d - data mean (xDim x 1)
% LL - log likelihood at each EM iteration
%
```

```
% OPTIONAL ARGUMENTS: NEED BYRON's assignopts FOR THIS!
%
% typ - 'fa' (default) or 'ppca'
% tol - stopping criterion for EM (default: 1e-8)
% cyc - maximum number of EM iterations (default: 1e8)
% minVarFrac - fraction of overall data variance for each observed
dimension
% to set as the private variance floor. This is used to combat
% Heywood cases, where ML parameter learning returns one or more
% zero private variances. (default: 0.01)
% (See Martin & McDonald, Psychometrika, Dec 1975.)
% verbose - logical that specifies whether to display status messages
% (default: false)
%
% Code adapted from ffa.m by Zoubin Ghahramani.
%
% @ 2009 Byron Yu -- byronyu@stanford.edu
tol = 1e-8;
cyc = 1e8;
minVarFrac = 0.01;
verbose = false;
% assignopts(who, varargin);
randn('state', 0);
[xDim, N] = size(X);
% Initialization of parameters
cX = cov(X', 1);
if rank(cX) == xDim
      scale = exp(2*sum(log(diag(chol(cX))))/xDim);
else
      % cX may not be full rank because N < xDim
      fprintf('WARNING in fastfa.m: Data matrix is not full rank.\n');
      r = rank(cX);
      e = sort(eig(cX), 'descend');
      scale = geomean(e(1:r));
end
L = randn(xDim,zDim)*sqrt(scale/zDim);
Ph = diag(cX);
d = mean(X, 2);
varFloor = minVarFrac * diag(cX);
I = eye(zDim);
const = -xDim/2*log(2*pi);
LLi = 0;
LL = [];
for i = 1:cyc
      % =======
      % E-step
      % =======
      iPh = diag(1./Ph);
      iPhL = iPh * L;
      MM = iPh - iPhL / (I + L' * iPhL) * iPhL';
      beta = L' * MM; % zDim x xDim
      cX_beta = cX * beta'; % xDim x zDim
      EZZ = I - beta * L + beta * cX_beta;
      % Compute log likelihood
      LLold = LLi;
      ldM = sum(log(diag(chol(MM))));
      LLi = N*const + N*ldM - 0.5*N*sum(sum(MM .* cX));
```

```
        if verbose
              fprintf('EM iteration %5i lik %8.1f \r', i, LLi);
        end
        LL = [LL LLi];
        % =======
        % M-step
        % =======
        L = cX_beta / EZZ;
        Ph = diag(cX) - sum(cX_beta .* L, 2);
        if isequal(typ, 'ppca')
              Ph = mean(Ph) * ones(xDim, 1);
        end
        if isequal(typ, 'fa')
              % Set minimum private variance
              Ph = max(varFloor, Ph);
        end
        if i<=2
              LLbase = LLi;
        elseif (LLi < LLold)
              disp('VIOLATION');
        elseif ((LLi-LLbase) < (1+tol)*(LLold-LLbase))
              break;
        end
  end
  if verbose
        fprintf('\n');
  end
  if any(Ph == varFloor)
        fprintf('Warning: Private variance floor used for one or more
  observed dimensions in FA.\n');
  end
  estParams.L = L;
  estParams.Ph = Ph;
  estParams.d = d;
```