CH5440
ASSIGNMENT 4

Prem Sagar S - AE14B021

April 10, 2017

Problem 1

Part a

The number of model constraints is unknown. It has to be less than 28, the maximum number of variables.

The IPCA gives a converged solution for different choices of number of constraints. The eigen values for each choice of number of constraints are presented in table 1. The SCREE plot for each of these can predict the right number of constraints clearly. They are presented in figure 1.

If we choose less number of constraints than the actual, since the last m eigen values are small, including extra eigen vectors to obtain the constraint matrix affects very little. But if we choose more constraints than the true constraints, some of the eigen vectors that may actually contribute to the constraint matrix might be excluded. This can be seen in the SCREE plots for m=15 and m=20, where the separation between the dominant PCs and noise is reducing. However, the last m eigen values are all not equal to 1, but they are significantly lower than the rest.

Thus, from the SCREE plots it is clearly possible to estimate the number of constraints to be 11. (The least 11 eigen values)

Part b

The first seventeen variables are indepent.

$$\begin{aligned} A\bar{Z} &= 0 \\ A_D\bar{Z}_D + A_I\bar{Z}_I &= 0 \\ \bar{Z}_D &= -(A_D^T A_D)^{-1} A_D^T A_I \bar{Z}_I \\ &= B\bar{Z}_I \end{aligned}$$

where B is the regression matrix.

The maximum absolute difference between true regression matrix and estimated regression matrix for different choice of number of constraints is below. The estimation of the regression matrix is best when the right number of constraints is chosen. This was also evident from the fact that the convergence tolerance had to be kept high for the remaining choices and they took way more number of iterations to converge than m=11.

m=8	m=10	m=11	m=12	m=20
1851965.93324290	2371631.77667389	4417507.27111028	2648680.29547111	2623302.38646008
875657.668282495	1265832.77415886	2116885.85210932	1420415.99479632	1488632.73471233
145665.953409263	294871.890894447	565810.907271507	259427.922641556	330116.275868767
116803.050042759	178913.250622527	316209.033910630	191825.649431976	138553.332145231
56840.4674728384	89920.3271813423	204865.029028996	149162.041305520	102534.460616476
30928.3149624901	83709.0638556466	105732.277027884	62486.7012784825	74751.3149420528
17506.8800858548	32063.8175315455	60815.0982666470	58477.6774362228	60091.7496529112
11939.7977448139	20247.1304705885	44897.0487591439	31940.6919627730	45095.3883788547
8614.79064372355	15285.0260334702	28355.4745834380	21459.6506617463	20217.1667057180
6497.25441332932	12488.3873577761	22038.6500158883	17084.2428517215	17185.9957456354
4663.93233387900	11017.7675035197	18176.0310243797	14358.7421219805	13144.7042111218
3800.61311828857	6481.22326709634	10484.0784680069	9135.12587368575	12048.3419608154
3008.00071090766	5064.91764994853	8446.09576172402	5163.79891484780	8847.52728792830
1961.04306847472	2906.32103495848	5790.24466167593	3957.82526661569	6858.32680509848
1327.92093367081	1797.93576425417	3461.79627284864	2890.93547000044	5612.87875195548
899.642173440407	1441.92671783497	2577.74083230488	2155.96138950565	4470.72000716899
677.989313827669	1183.51225564389	2141.91234370968	1256.73721958837	3921.53320118824
4.34149152073016	6.10372731094158	8.61075138509408	6.55106306139646	5.47373971167707
1.88014024947449	2.59106223625699	4.39483365805475	5.28224337988179	4.53587326022490
1.50476861458314	2.08004834412172	4.08159271559063	3.76262388436638	3.99463955999016
1.27656222547369	1.64755147909944	2.69909460126619	2.55661604312312	3.36334367113611
0.849256202950845	1.59617496807225	2.36754362285925	2.34963570223197	2.97902543314791
0.699476927715203	1.36980673420588	1.97431208027464	1.88145937616793	2.91554471500974
0.688245140432243	1.29376120787409	1.82974666133203	1.38921782423494	2.34531035410630
0.340918072437973	1.15129279106469	1.59315661458567	1.02500435162413	2.12512686660007
0.327874098602211	0.975936220449357	1.39955024279149	0.941255469542402	1.74641435368103
0.299138808887262	0.801200691095789	1.10419965859166	0.866186677988422	1.49212058720544
0.194835330792776	0.452437057623951	0.818985376533508	0.558441783146040	0.846519228973717

Table 1: Eigen values for different choices of number of constraints

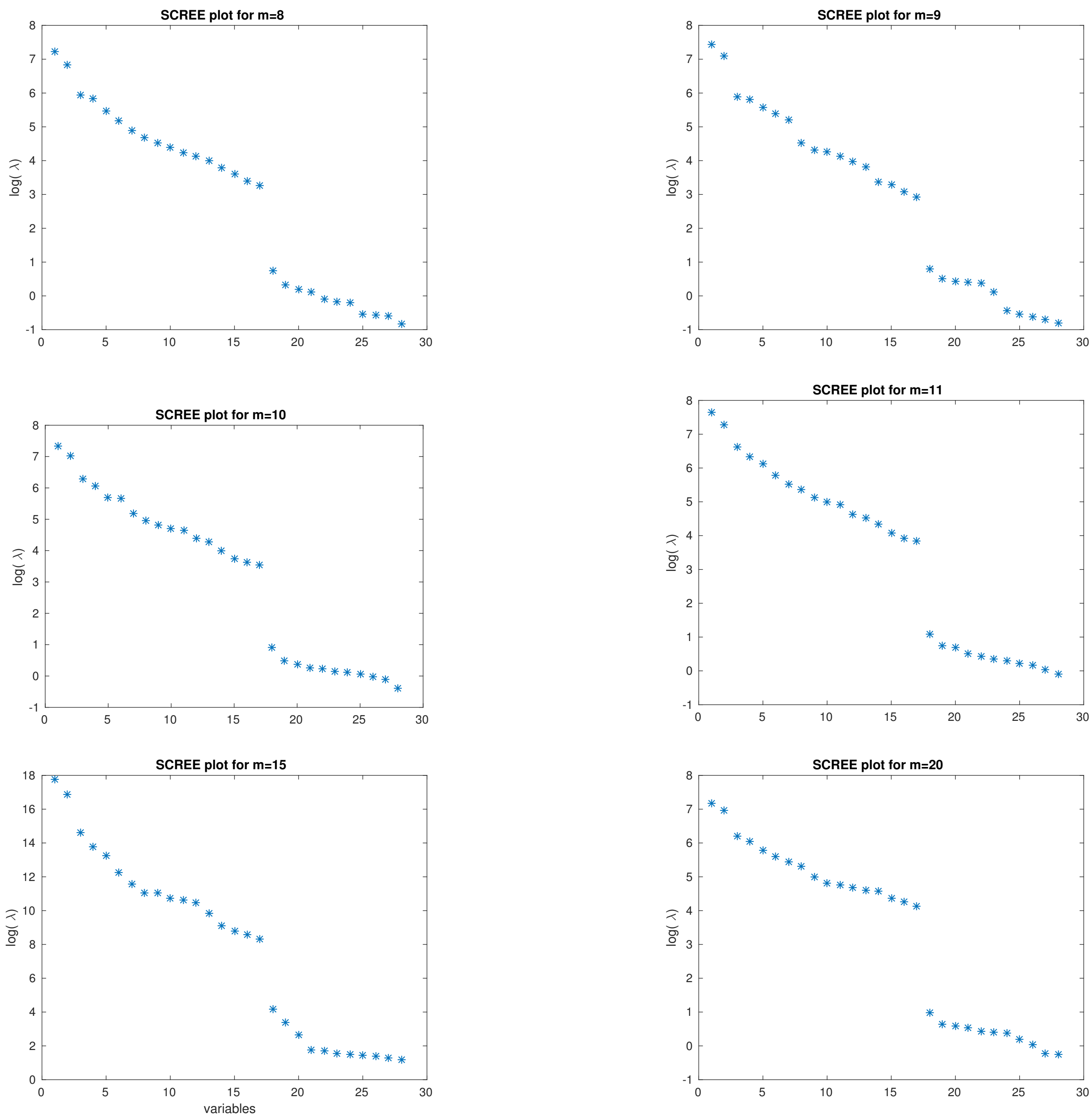

Figure 1: SCREE plots for different choices of number of constraints

Constraints	max(abs$[R-\hat{R}]$)
8	1.0000
9	1.0001
10	1.0109
11	**0.0059**
12	0.6242
15	5.7526
20	1.7498

Table 2: Maximum absolute difference between true and estimated regression matrices

Problem 2

Part a

The corrupted samples are removed and the constraint matrix is obtained applying PCA to the remaing samples giving a constraint matrix $\hat{A}_0$.

Maximum absolute error between true regression matrix and the regression matrix estimated by removing corrupted samples =0.007564108

Part b

The corrupted data is sampled with the variable average, i.e, for each missing data, the mean value of that variable obtained from the uncorrupted data is imputed. Now PCA is applied to obtaina a constraint matrix $\hat{A}_{mean}$.

Maximum absolute error between true regression matrix and the regression matrix estimated by imputing mean= 8.214714

Part c

Now using $\hat{A}_{mean}$, the missing data is estimated just like we do a regression. The new data set Y_1 is again used to obtain a new constraint matrix $\hat{A}_1$. This is imputed to get a new data set Y_2 and then subsequently $\hat{A}_2$ and so on. This is done until the maximum error between the true and estimated regression matrices reach a value below that obtained for the mean imputed data. A reasonable value for tolerance can be chosen and thus, the loop can be stopped.

Choosing the loop end criterion to be err_PCA<0.9*err_mean, the algorithm converged after 9 iterations.

The maximum absolute error between true regression matrix and the regression matrix estimated by PCA imputation= 0.005962763

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PCA on uncorrupted data:
Max error with PCA on autoscaled data = 7.564108e-03
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
PCA on mean imputed data:
Max error with mean imputed data = 8.214714e+00
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Data matrix iteratively imputed by PCA:
Max error with iteration 0 PCA imputed data = 4.385424e-02
Max error with iteration 1 PCA imputed data = 6.184575e-03
Max error with iteration 2 PCA imputed data = 5.989214e-03
Max error with iteration 3 PCA imputed data = 5.965460e-03
Max error with iteration 4 PCA imputed data = 5.963029e-03
Max error with iteration 5 PCA imputed data = 5.962788e-03
Max error with iteration 6 PCA imputed data = 5.962765e-03
Max error with iteration 7 PCA imputed data = 5.962763e-03
Max error with iteration 8 PCA imputed data = 5.962762e-03
Max error with iteration 9 PCA imputed data = 5.962762e-03
Solution has converged!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Part d

The same procedure is followed from a through instead with IPCA.

- Maximum absolute error between true regression matrix and the regression matrix estimated by removing corrupted samples = 0.007294555. This has improved slightly, although it can still be improved further with lower tolerance, but requires lot of computation time.
- Maximum absolute error between true regression matrix and the regression matrix estimated by imputing mean= *0.5816338.* This has significantly better than what usual PCA did to the mean imputed data.
- The maximum absolute error between true regression matrix and the regression matrix estimated by IPCA imputation= 0.005599. This too converged in about 9 iterations giving a slightly better estimate than PCA. Again, much better results can be obtained with lower tolerance requiring more computation time.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
IPCA on uncorrupted data:
Max error with IPCA on autoscaled data = 7.294555e-03
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
IPCA on mean imputed data:
Max error with mean imputed data = 5.816338e-01
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Data matrix iteratively imputed by IPCA:
Max error with iteration 0 IPCA imputed data = 1.877488e-02
Max error with iteration 1 IPCA imputed data = 5.730123e-03
Max error with iteration 2 IPCA imputed data = 5.669772e-03
Max error with iteration 3 IPCA imputed data = 5.606442e-03
Max error with iteration 4 IPCA imputed data = 5.599965e-03
Max error with iteration 5 IPCA imputed data = 5.599306e-03
Max error with iteration 6 IPCA imputed data = 5.599243e-03
Max error with iteration 7 IPCA imputed data = 5.599237e-03
Max error with iteration 8 IPCA imputed data = 5.599236e-03
Max error with iteration 9 IPCA imputed data = 5.599236e-03
Solution has converged!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Table 3: *Summary of maximum absolute difference between true and estimated regression matrix*

Data Samples used	**PCA**	**IPCA**
Corrupted samples removed	0.007564108	0.007294555
After imputing mean of variable data	8.214714	0.5216338
After iteratively imputing by PCA/IPCA	0.005962	0.005599

Appendix

Problem 1 code:

```
clear all; clc;
load('steamdata.mat')
[nvar nsamples]=size(Fmeas);
m=15;              % number of constraints
Y=Fmeas;           % data matrix of measurements
Sy=Y*Y'/nsamples;          % data covariance matrix
%% initialize
k=0;               % counter
lambda0=0;
tol=0.05;     % error tolerance
err=1;
%% initial guess
for i=1:nvar
    stderr(i,i)=0.0001*Sy(i,i);
end
%% Initial guess from PCA
%  Ahat =myPCA1(Y,std,m);
% stderr=diag(stdest(Ahat,Y));
%% IPCA to estimate constraint model and error covariance matrix
```

```
while(err>tol)
%  for i=1:1
L=chol(stderr);              % Cholesky decomposition
L=L';
Ys=inv(L)*Y;                 % transformation
[U ,S ,V]=svd(Ys);              % svd of transformed data
A=(U(:,nvar-m+1:nvar))'*inv(L);      % constraint matrix of original data
% Lambda1=diag(S.*S);
Lambda1=diag(S);
Lambda=Lambda1(nvar-m+1:nvar);                       % eigen values
lambda=sum(Lambda);              % sum of singular values
err=abs(lambda-lambda0);              % relative error in singular value sum
lambda0=lambda;
stderr=diag(stdest(A,Ys));             % new estimate for error covariance matrix
k=k+1
end
%%
% Compare Atrue and A
% theta_pca = 180*subspace(Atrue', A')/pi
% Determine how well the model matches with the true constraint matrix.
% For this determine the minimum distance of each true constraint vector from the
% row space of model constraints
% for i = 1:3
%     bcol = Atrue(i,:)';
%     dist_pca(i) = norm(bcol - A'*inv(A*A')*A*bcol);
% end
% DIST_PCA=norm(dist_pca)
%%
plot(log(Lambda1),'*')             % SCREE plot
% xlabel(' ')
ylabel('log(\lambda)')
title('SCREE plot for m=20')
%%
% true regression model
nind=17;   % number of independent variables
% Rtrue=-Atrue(:,1:nind)\Atrue(:,nind+1:nvar);
% Rtrue=-inv(Atrue(:,nind+1:nvar)'*Atrue(:,nind+1:nvar))*(Atrue(:,nind+1:nvar))'*Atrue(:,1:
    nind);
% estimated regression model
% Rhat=-A(:,1:nind)\A(:,nind+1:nvar);
%  Rhat=-inv(A(:,nind+1:nvar)'*A(:,nind+1:nvar))*(A(:,nind+1:nvar))'*A(:,1:nind);
Rtrue=regress(Atrue(:,nind+1:nvar),Atrue(:,1:nind));
Rhat=regress(A(:,nind+1:nvar),A(:,1:nind));
maxerror=max(max(abs(Rtrue-Rhat)))
eig=diag(S.^2);
```

```
function [R]=regress(Adep,Aind)

R=-Adep\Aind;
```

Problem2 code:

```
clear all; clc;
```

```
load('steamdatamiss.mat')
%% inputs
Y=Fmeas;
nind=17;
m=12;                                    % number of constraints
[nvar nsamples]=size(Y);
Rtrue=regress(Atrue(:,nind+1:nvar),Atrue(:,1:nind));
%% eliminate samples with missing data
Ynew=Y;
k=[];
for j=1:nsamples
    for  i=1:nvar
        if isnan(Ynew(i,j))
             k=[k; i j]    ;          % keep track which element is Nan
        end
    end
end
 k1=k(:,2)';                          % columns to be removed
 Ynew(:,k1)=[];                       % uncorrupted unscaled data matrix
 Ymean=(mean(Ynew'))';
 %% Constraint model using PCA on autoscaled uncorrupted data
     fprintf('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')

 %  [Rhat1,Ahat1,err0,s1]=myPCA(Rtrue,Ynew,std,m,nind);
   [Rhat1,Ahat1,err0,s1]=myIPCA(Rtrue,Ynew,std,m,nind,0.05);

%  fprintf('Max error with PCA on autoscaled data = %s\n',err0)
 fprintf('Max error with IPCA on autoscaled data = %s\n',err0)
   %% Imputation using mean
   Y2=Y;
for i=1:max(size(k1))
     Y2(:,k1(i))=imputeMean1(Ymean,Y(:,k1(i)));     % impute mean to the k(i)th sample
%      Y2(:,k1(i))=imputeMean(Y(:,k1(i)));     % impute mean to the k(i)th sample

end
   %%
        fprintf('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')

    fprintf('PCA on mean imputed data:\n')
%       [Rhat2,Ahat2,err2,s2]=myPCA(Rtrue,Y2,std,m,nind);
          [Rhat2,Ahat2,err2,s2]=myIPCA(Rtrue,Y2,std,m,nind,0.15);
     fprintf('Max error with mean imputed data = %s\n',err2)
%       fprintf('Max error with mean imputed data = %s\n',err2)

     %% iterate
     fprintf('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')
     fprintf('Data matrix iteratively imputed by PCA: \n')
 AHat=Ahat2;
 tol=1; err=s2;
 s=s2; i=0;
 err1=1.5*err2;
        while err >0.0000000001
```

```
        err=err1;
        YNew=imputePCA1(AHat,Y,k1);
%          [RHat,AHat,err1,s]=myPCA(Rtrue,YNew,std,m,nind);
        [RHat,AHat,err1,s]=myIPCA(Rtrue,YNew,std,m,nind,0.5);
%          fprintf('Max error with iteration %d PCA imputed data = %s\n',i,err1)
                fprintf('Max error with iteration %d IPCA imputed data = %s\n',i,err1)
            err=err-err1;


         i=i+1;
      end

      fprintf('Yo! Solution has converged!!\n')
    fprintf('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n')

%%
```

```
function [Rhat,Ahat,err,s]=myPCA(Rtrue,Y,std,m,nind)

[n,N]=size(Y);                              % get size of  experimental samples
     xbar=(mean(Y'))';                              % mean of columns of experimental samples
    scale=std*ones(1,N);
%     Y=Y-(ones(n,1)*xbar);              % mean shift
Y=Y-xbar*(ones(1,N));
    Y=Y./scale;                         % scale by standard deviation
    Ss=Y*Y'/N;                          % covariance matrix
    [U S V] = svd(Ss);                            % svd
    s=sum(diag(S(n-m+1:n)));                          %sum of eigen values
    Ahat = U(:,n-m+1:n)'*diag(1./std);                       % Estimated constraint model in unscaled variables
Rhat=regress(Ahat(:,nind+1:n),Ahat(:,1:nind));
%
    err=max(max(Rtrue-Rhat)) ;
```

```
function [Rhat,Ahat,maxerror,lambda]=myIPCA(Rtrue,Y,std,m,nind,tol)
[nvar nsamples]=size(Y);
Sy=Y*Y'/nsamples;            % data covariance matrix
%% initialize
k=0;                 % counter
lambda0=0;
%tol=0.9;             % error tolerance
err=1;
% for i=1:nvar
%    stderr(i,i)=0.0001*Sy(i,i);       % initialize error covariance matrix
% end
%% Initial guess from PCA
 Ahat =myPCA1(Y,std,m);
stderr=diag(stdest(Ahat,Y));
%% IPCA to estimate constraint model and error covariance matrix
while(err>tol)

```

```
L=chol(stderr);                % Cholesky decomposition
L=L';
Ys=inv(L)*Y;                   % transformation
[U S V]=svd(Ys);               % svd of transformed data
Ahat=(U(:,nvar-m+1:nvar))'*inv(L);      % constraint matrix of original data
Lambda1=diag(S);
Lambda=Lambda1(nvar-m+1:nvar);                      % eigen values
lambda=sum(Lambda);                % sum of singular values
err=abs(lambda-lambda0);              % relative error in singular value sum
lambda0=lambda;
stderr=diag(stdest(Ahat,Ys));             % new estimate for error covariance matrix
k=k+1           ;
end
%%
% Rtrue=-Atrue(:,1:nind)\Atrue(:,nind:nvar);

% estimated regression model
Rhat=regress(Ahat(:,nind+1:nvar),Ahat(:,1:nind));
maxerror=max(max(abs(Rtrue-Rhat)));
```

```
function [out]=imputeMean(v)

nvar=max(size(v));
k=[];
 for  i=1:nvar
      if isnan(v(i))
          k=[k; i]    ;           % keep track which element is Nan
      end
 end

 % mean
 v1=v;
 v1(k)=[];
v(k)=mean(v1);
out=v;
 end
```

```
function [Y1]=imputePCA1(A,Y,k1)
Ynew=Y;
    for i=1:max(size(k1))
        Ynew(:,k1(i))=imputePCA(A,Y(:,k1(i)));
    end

    Y1=Ynew;

end
```

```
function [v3]=imputePCA(A,v)

%% identify independent variables or nan variables
k=[];
v1=v;
```

```
for i=1:max(size(v))
    if isnan(v(i))
        k=[k;i];
                % dependent variable
    end
end
v1(k)=[];
%% get regression matrix
P=A; P(:,k)=[];    % Aindependent
% Q=A(:,k);          % Adependent
% R=-inv(Q'*Q)*Q'*P;            % regression matrix

A1=A(:,k);
B1=-P*v1;
V=A1\B1;
%% evaluate the missing terms and impute
v(k)=V;            % impute
v3=v;              % return the imputed sample
 end
```