2.- LINER ALGEBRA FOR RANKING

PLURARITY VOTE METHOD

```
clc;
close all;
clear all;
[x1 x2 x3 x4 x5] = importdata('rankingcandidates.dat', 1, 240);
A = [x1 x2 x3 x4 x5];
[m n] = size(A);
Hillary = sum(A(:,1) == 'HC');
Donald = sum(A(:,1) == 'DT');
Bernie = sum(A(:,1) == 'BS');
JK = sum(A(:,1) == 'JK');
Ted = sum(A(:,1) == 'TC');
results_array = [Hillary, Donald, Bernie, JK, Ted];
names_array = {'HC','DT','BS','JK','TC'};
answer = [names_array; num2cell(results_array)];
Max = max(results_array);
disp(answer);
for j=1:n
    if results_array(j)==Max
        Winner = names_array(j);
    end
end
celldisp(Winner);
```

```
    'HC'    'DT'    'BS'    'JK'    'TC'
    [85]    [44]    [96]    [ 7]    [ 8]


Winner{1} =

BS
```

AVERAGE VOTE METHOD

```
clc;
close all;
clear all;
[x1 x2 x3 x4 x5] = importdata('rankingcandidates.dat', 1, 240);
A = [x1 x2 x3 x4 x5];
B = fliplr(A);
[m n] = size(A);
Hillarysum = 0;
Donaldsum = 0;
Berniesum = 0;
JKsum = 0;
Tedsum = 0;
for i = 1:n
    Hillary = i*sum(B(:,i) == 'HC');
    Hillarysum = (Hillarysum + Hillary);
    Donald = i*sum(B(:,i) == 'DT');
    Donaldsum = (Donaldsum + Donald);
    Bernie = i*sum(B(:,i) == 'BS');
    Berniesum = (Berniesum + Bernie);
    JK = i*sum(B(:,i) == 'JK');
    JKsum = (JKsum + JK);
    Ted = i*sum(B(:,i) == 'TC');
    Tedsum = (Tedsum + Ted);

end
Hillaryavg = Hillarysum/m;
Bernieavg = Berniesum/m;
Donaldavg = Donaldsum/m;
JKavg = JKsum/m;
Tedavg = Tedsum/m;
results_array = [Hillaryavg, Donaldavg, Bernieavg, JKavg, Tedavg];
names_array = {'HC','DT','BS','JK','TC'};
answer = [names_array; num2cell(results_array)];
```

```
    'HC'        'DT'        'BS'        'JK'        'TC'
    [3.8583]    [2.6958]    [3.7417]    [2.5375]    [2.1667]


Winner{1} =

HC
```

```
Max = max(results_array);
disp(answer);
for j=1:n
    if results_array(j)==Max
        Winner = names_array(j);
    end
end
celldisp(Winner);
```

BORDA COUNT METHOD

```
clc;
close all;
clear all;
[x1 x2 x3 x4 x5] = importdata('rankingcandidates.dat', 1, 240);
A = [x1 x2 x3 x4 x5];
B = fliplr(A);
[m n] = size(A);
Hillarysum = 0;
Donaldsum = 0;
Berniesum = 0;
JKsum = 0;
Tedsum = 0;
for i = 1:n
    Hillary = (i-1)*sum(B(:,i) == 'HC');
    Hillarysum = (Hillarysum + Hillary);
    Donald = (i-1)*sum(B(:,i) == 'DT');
    Donaldsum = (Donaldsum + Donald);
    Bernie = (i-1)*sum(B(:,i) == 'BS');
    Berniesum = (Berniesum + Bernie);
    JK = (i-1)*sum(B(:,i) == 'JK');
    JKsum = (JKsum + JK);
    Ted = (i-1)*sum(B(:,i) == 'TC');
    Tedsum = (Tedsum + Ted);
    %Flip matrix A so that if we multiply i candidate
    %gets (i-1) points.
end
results_array = [Hillarysum, Donaldsum, Berniesum, JKsum, Tedsum];
names_array = {'HC','DT','BS','JK','TC'};
answer = [names_array; num2cell(results_array)];
Max = max(results_array);
disp(answer);
for j=1:n
    if results_array(j)==Max
        Winner = names_array(j);
    end
end
celldisp(Winner);
```

'HC'	'DT'	'BS'	'JK'	'TC'
[686]	[407]	[658]	[369]	[280]

```
Winner{1} =

HC
```

W-BORDA COUNT METHOD

```
[x1, x2, x3, x4, x5,] = importdata('rankingcandidates.dat', 1, 240);
A = [x1 x2 x3 x4 x5];
B = fliplr(A);
names_array = {'HC','DT','BS','JK','TC'};
W = [11 7 6 6 4] %[9 7 5 3 1] [8 6 4 2 1]
%[5 4 3 2 1] [9 4 4 2 1] [6 6 5 4 2]
W_n = fliplr(W);
[m n] = size(A);
Hillarysum = 0;
Donaldsum = 0;
Berniesum = 0;
JKsum = 0;
Tedsum = 0;
for i = 1:n
    Hillary = (W_n(i))*sum(B(:,i) == 'HC');
    Hillarysum = (Hillarysum + Hillary);
```

```
    Donald = (W_n(i))*sum(B(:,i) == 'DT');
    Donaldsum = (Donaldsum + Donald);
    Bernie = (W_n(i))*sum(B(:,i) == 'BS');
    Berniesum = (Berniesum + Bernie);
    JK = (W_n(i))*sum(B(:,i) == 'JK');
    JKsum = (JKsum + JK);
    Ted = (W_n(i))*sum(B(:,i) == 'TC');
    Tedsum = (Tedsum + Ted);

end
results_array = [Hillarysum, Donaldsum,
 Berniesum, JKsum, Tedsum];
answer = [names_array; num2cell(results_array)];
Max = max(results_array);
disp(answer);
for j=1:n
    if results_array(j)==Max
        Winner = names_array(j);
    end
end
celldisp(Winner);
```

```
>> WBordaCountMethod
W =
     8     6     4     2     1

    'HC'      'DT'      'BS'      'JK'      'TC'
    [1389]    [886]     [1335]    [784]     [646]

Winner{1} =
HC

W =
     9     7     5     3     1

    'HC'      'DT'      'BS'      'JK'      'TC'
    [1612]    [1054]    [1556]    [978]     [800]

Winner{1} =
HC

W =
     6     6     5     4     2

    'HC'      'DT'      'BS'      'JK'      'TC'
    [1304]    [1011]    [1263]    [1036]    [906]

Winner{1} =
HC

W =
     9     4     4     2     1

    'HC'      'DT'      'BS'      'JK'      'TC'
    [1290]    [862]     [1321]    [735]     [592]

Winner{1} =
BS

W =
     5     4     3     2     1

    'HC'      'DT'      'BS'      'JK'      'TC'
    [926]     [647]     [898]     [609]     [520]

Winner{1} =
HC

W =
    11     7     6     6     4

    'HC'      'DT'      'BS'      'JK'      'TC'
    [1923]    [1550]    [1937]    [1411]    [1339]

Winner{1} =
BS
```

PAGERANK ALGORITHM METHOD

```
clc;
close all;
clear all;
[x1, x2, x3, x4, x5,] = importdata('rankingcandidates.dat', 1, 240);
A = [x1 x2 x3 x4 x5];
B = fliplr(A);
names_array = {'HC','DT','BS','JK','TC'};
e = 0.8;
[m, n] = size(A);
Hillarysum = 0;
Donaldsum = 0;
Berniesum = 0;
JKsum = 0;
Tedsum = 0;
HC_BS = 0;
HC = ismember(A,'HC'); %Create 1,0 logical array
DT = ismember(A,'DT');
BS = ismember(A,'BS');
JK = ismember(A,'JK');
TC = ismember(A,'TC');
HCvsDT = 0;
HCvsBS = 0;
HCvsJK = 0;
HCvsTC = 0;
for i = 1:m %Find num of times candidate i win to j
    if find(HC(i,:) == 1) > find(DT(i,:) == 1)
        HCvsDT = HCvsDT +1;
    end
end
for i = 1:m
    if find(HC(i,:) == 1) > find(BS(i,:) == 1)
            HCvsBS = HCvsBS + 1;
    end
end
```

```
StochasticM =

         0    0.4205    0.3038    0.3283    0.2897
    0.2628    0.2550         0    0.1963    0.2088
    0.4124         0    0.2948    0.3181    0.2853
    0.1679    0.1722    0.2242         0    0.2162
    0.1569    0.1523    0.1772    0.1574         0

    'HC'      'DT'      'BS'      'JK'      'TC'
    [0.2425]  [0.1793]  [0.2584]  [0.1696]  [0.1503]


Winner{1} =

BS
```

```
for i = 1:m
    if find(HC(i,:) == 1) > find(TC(i,:) == 1)
            HCvsTC = HCvsTC + 1;
    end
end
for i = 1:m
    if find(HC(i,:) == 1) > find(JK(i,:) == 1)
            HCvsJK = HCvsJK + 1;
    end
end

BSvsHC = 0;
BSvsDT = 0;
BSvsJK = 0;
BSvsTC = 0;
for i = 1:m
    if find(BS(i,:) == 1) > find(DT(i,:) == 1)
        BSvsDT = BSvsDT +1;
    end
end
for i = 1:m
    if find(BS(i,:) == 1) > find(HC(i,:) == 1)
            BSvsHC = BSvsHC + 1;
    end
end
for i = 1:m
    if find(BS(i,:) == 1) > find(TC(i,:) == 1)
            BSvsTC = BSvsTC + 1;
    end
end
for i = 1:m
    if find(BS(i,:) == 1) > find(JK(i,:) == 1)
            BSvsJK = BSvsJK + 1;
    end
end

DTvsHC = 0;
DTvsBS = 0;
DTvsJK = 0;
DTvsTC = 0;
for i = 1:m
    if find(DT(i,:) == 1) > find(BS(i,:) == 1)
        DTvsBS = DTvsBS +1;
    end
end
for i = 1:m
    if find(DT(i,:) == 1) > find(HC(i,:) == 1)
            DTvsHC = DTvsHC + 1;
    end
end
for i = 1:m
    if find(DT(i,:) == 1) > find(TC(i,:) == 1)
            DTvsTC = DTvsTC + 1;
    end
end
for i = 1:m
    if find(DT(i,:) == 1) > find(JK(i,:) == 1)
            DTvsJK = DTvsJK + 1;
    end
end

TCvsHC = 0;
TCvsBS = 0;
TCvsJK = 0;
TCvsDT = 0;
for i = 1:m
    if find(TC(i,:) == 1) > find(BS(i,:) == 1)
        TCvsBS = TCvsBS +1;
    end
end
for i = 1:m
    if find(TC(i,:) == 1) > find(HC(i,:) == 1)
            TCvsHC = TCvsHC + 1;
    end
end
for i = 1:m
    if find(TC(i,:) == 1) > find(DT(i,:) == 1)
            TCvsDT = TCvsDT + 1;
    end
end
for i = 1:m
    if find(TC(i,:) == 1) > find(JK(i,:) == 1)
            TCvsJK = TCvsJK + 1;
    end
end

JKvsHC = 0;
JKvsBS = 0;
JKvsTC = 0;
JKvsDT = 0;
for i = 1:m
    if find(JK(i,:) == 1) > find(BS(i,:) == 1)
        JKvsBS = JKvsBS +1;
    end
end
for i = 1:m
    if find(JK(i,:) == 1) > find(HC(i,:) == 1)
            JKvsHC = JKvsHC + 1;
    end
end
for i = 1:m
    if find(JK(i,:) == 1) > find(DT(i,:) == 1)
            JKvsDT = JKvsDT + 1;
    end
end
for i = 1:m
    if find(JK(i,:) == 1) > find(TC(i,:) == 1)
            JKvsTC = JKvsTC + 1;
    end
end

for j= 1:m %sum num loss against other
candidates
    for i = 1:n
        if (HC(j,i) == 0)
            Hillarysum = Hillarysum + 1;
        else if (HC(j,i) == 1)
                break
            end
        end
    end
end
for j= 1:m
    for i = 1:n
        if (DT(j,i) == 0)
            Donaldsum = Donaldsum + 1;
        else if (DT(j,i) == 1)
                break
            end
        end
    end
end
for j= 1:m
    for i = 1:n
        if (BS(j,i) == 0)
            Berniesum = Berniesum + 1;
        else if (BS(j,i) == 1)
                break
            end
        end
```

```
    end
end
for j= 1:m
    for i = 1:n
        if (JK(j,i) == 0)
            JKsum = JKsum + 1;
        else if (JK(j,i) == 1)
                break
            end
        end
    end
end
for j= 1:m
    for i = 1:n
        if (TC(j,i) == 0)
            Tedsum = Tedsum + 1;
        else if (TC(j,i) == 1)
                break
            end
        end
    end
end
%Create Stochastic Matrix
StochMat = [0, HCvsDT/Hillarysum,
HCvsBS/Hillarysum, HCvsJK/Hillarysum,
HCvsTC/Hillarysum;
   BSvsHC/Berniesum, BSvsDT/Berniesum, 0,
BSvsJK/Berniesum, BSvsTC/Berniesum;
   DTvsHC/Donaldsum, 0, DTvsBS/Donaldsum,
DTvsJK/Donaldsum, DTvsTC/Donaldsum;
   JKvsHC/JKsum, JKvsDT/JKsum, JKvsBS/JKsum,
0, JKvsTC/JKsum;
   TCvsHC/Tedsum, TCvsDT/Tedsum,
TCvsBS/Tedsum,TCvsJK/Tedsum, 0];
StochasticM = StochMat'
%Perron-Forbenius eigenvector
Perron = e * StochasticM +((1-e)/n)*ones(n);
[P D] = eig(Perron);
NormPer =  P(:,1)'/sum(P(:,1));
answer = [names_array; num2cell(NormPer)];
Max = max(NormPer);
disp(answer);
for j=1:n
    if NormPer(j)==Max
        Winner = names_array(j);
    end
end
celldisp(Winner);
```

According to my final resuld for ranking the candidates, Bernie Sanders and Hillary Clinton will win depending on the method we used. Surprisedly, there was some results that I did not expect. For example, I did not notice at the beggining that using the W-borda count method, depending on the vector you choose, it will change the winner. In addition, the pagerank method is really interesting because it considers the importance of losing against a particular candidate in this case.

I did the PageRank implementation a little bit different than the one it was showed in the slides. I create a binary matrix for each candidate considering when a particular candidate appears and then I compare each matrix with the location of the particular value 1. In other words, I create a network with the results of each candidate against another candidate. At the end, the ressults where basically the same.

I think the PageRank method considers more factors which is more interesting for me, so I Believe this is the fairest mehod.

3 .- Using SVD's or a network to decide the ranking of dificulty:

MASSEY SCORES

```
clc;
close all;
clear all;
Q = importexamscores('examscores.dat', 1, 31);
[row col] = size(Q);
N = zeros(col);
D = 0;
diff = 0;
Y_k(1:col) = 0;
sum = 1;
for i = 1:row%Create 1,-1 matrix for Q_b>Q_a
    for j = 1:col
        for k = 1:col
            if Q(i,j)>Q(i,k)
             N(sum,j) = 1;
             N(sum,k) = -1;
             diff = Q(i,j)-Q(i,k);
             D(sum) = diff;
             sum = sum + 1;
            end
        end
    end
end
D = D';
[row1 col1] = size(N);
for i = 1:row1%Create Y difference Q_b-Q_a
    for j = 1:col1
        if N(i,j) == 1
        Y_k(j) = Y_k(j) + D(i);
        end
    end
end
Y = Y_k';
X = N'*N;
AX_Y = [N D]% Matrix Ar=y
X(end,:) = 1%add 1 last row
Y(end,:) = 0 % add 0 las row
result = linsolve(X,Y); %Solve Ar = y
results_array = result';
names_array = {'Q1','Q2', 'Q3','Q4','Q5','Q6','Q7'};
answer = [names_array; num2cell(results_array)];
disp(answer);
```

X =

159	-29	-27	-26	-25	-23	-29
-29	158	-24	-23	-25	-26	-31
-27	-24	134	-17	-20	-21	-25
-26	-23	-17	132	-21	-23	-22
-25	-25	-20	-21	133	-21	-21
-23	-26	-21	-23	-21	140	-26
1	1	1	1	1	1	1

Y =

225
123
136
148
177
206
0

'Q1'	'Q2'	'Q3'	'Q4'	'Q5'	'Q6'	'Q7'
[1.1116]	[0.5023]	[0.7513]	[0.9685]	[1.1864]	[1.1450]	[-5.6651]

As shown in class, once we find the rank-one matrices. It is possible to find a vector which points excatly where the data is more important. In other words, we can find a vector s which give us information about where the data is concentrated. In this example, it shows us where students received a lower score which is as a result is the hardest example.

Now for the Massey scores logarithm, once we created a network which give us the relationship between two scores for a particular student k(1,-1,0...). Then we create a matrix, which contains in the diagonal, the number of times a score was greater than another one. And around the diagonal, the numer of times two, different scores where compared.
Note that we do not considered when two scores where equal.
Then we create Matrix Y which is the difference between each of the scores. After that, We sve the system of equations Xr = y where r is the ranking of those scores.

Simiarly, Colley's method, use the same concept that Massey's but this is modified so that we do not get a matrix that is not full rank, and we can solve the system directly

COLLEY SCORES

```
clc;
close all;
clear all;
Q = importexamscores('examscores.dat', 1, 31);
[row col] = size(Q);
D = 0;
diff = 0;
Y_k(1:col) = 0;
sum = 1;
for i = 1:row%Create 1,-1 matrix for Q_b>Q_a
    for j = 1:col
        for k = 1:col
            if Q(i,j)>Q(i,k)
             N(sum,j) = 1;
             N(sum,k) = -1;
             diff = Q(i,j)-Q(i,k);
             D(sum) = diff;
             sum = sum + 1;
            end
        end
    end
end
[row1 col1] = size(N);
for i = 1:row1 %1+1/2(Y)
    for j = 1:col1
        if N(i,j) == 1
        Y_k(j) = Y_k(j) + D(i);
        Y_k(j) = 1 + 0.5*(Y_k(j));
        end
    end
end
Y = Y_k';
X = N'*N;
for i = 1:col%add to diagonal
    for j = 1:col
        if i == j
        X(i,j) = X(i,j) + 2;
        end
    end
end
X
Y
result = linsolve(X,Y); %Solve Ar = y
results_array = result';
names_array = {'Q1','Q2', 'Q3','Q4','Q5','Q6','Q7'};
answer = [names_array; num2cell(results_array)];
disp(answer);
Min = min(results_array);
for j=1:col
    if results_array(j)==Min
        Hardest = names_array(j);
    end
end
celldisp(Hardest);
```

```
X =

   161   -29   -27   -26   -25   -23   -29
   -29   160   -24   -23   -25   -26   -31
   -27   -24   136   -17   -20   -21   -25
   -26   -23   -17   134   -21   -23   -22
   -25   -25   -20   -21   135   -21   -21
   -23   -26   -21   -23   -21   142   -26
   -29   -31   -25   -22   -21   -26   156

Y =

    4.6702
    3.2555
    4.5152
    4.2702
    4.4388
    4.1419
    3.6727

'Q1'        'Q2'        'Q3'        'Q4'        'Q5'        'Q6'        'Q7'
[2.0716]    [2.0641]    [2.0712]    [2.0697]    [2.0708]    [2.0687]    [2.0662]

Hardest{1} =

Q2
```

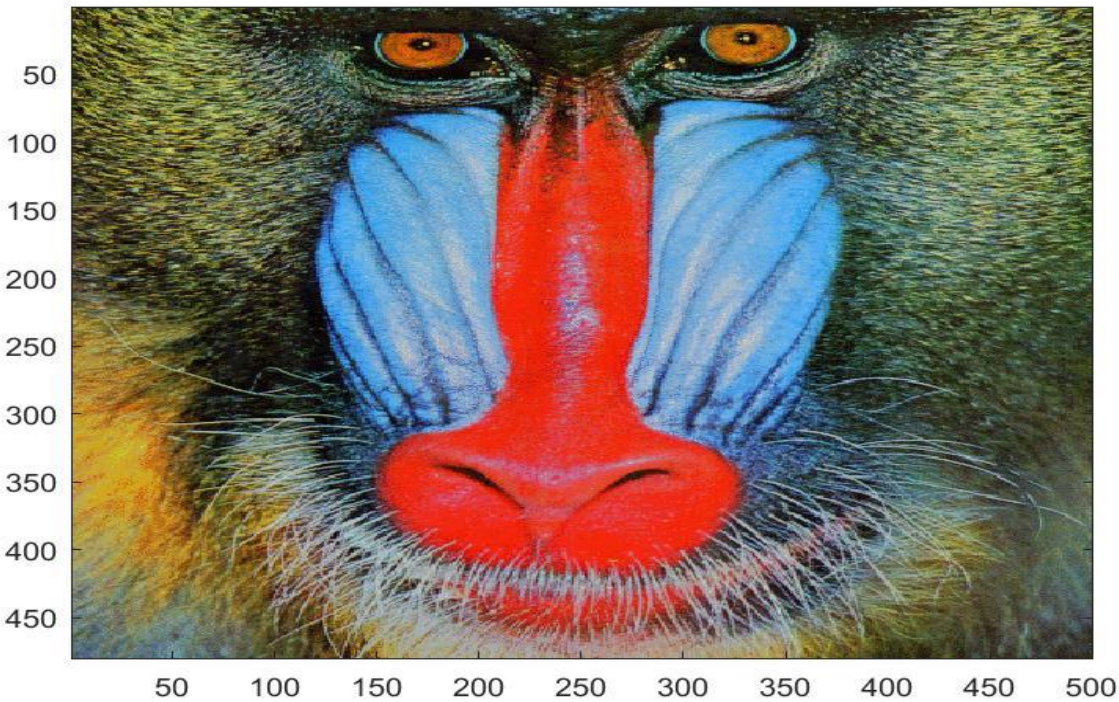
SVD scores

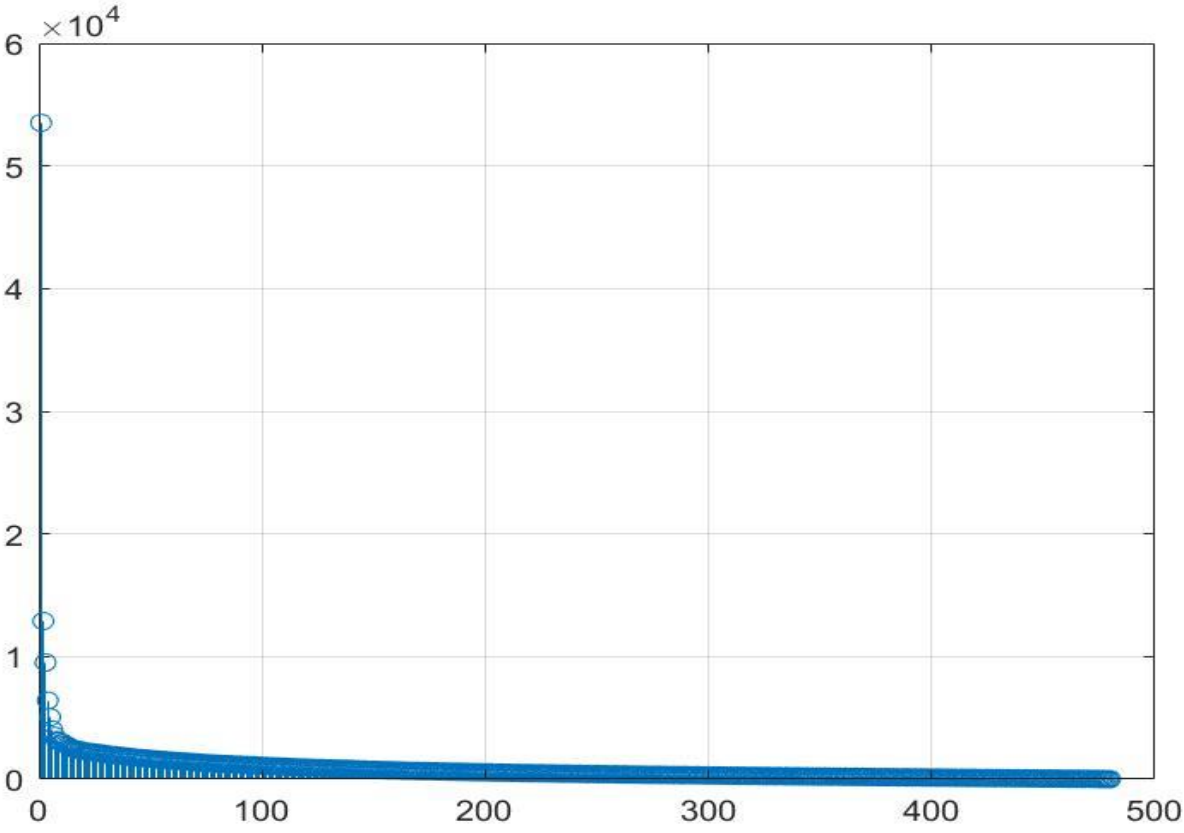
```
clc;
close all;
clear all;
Q = importexamscores('examscores.dat', 1, 31);
[row col] = size(Q);
r = rank(Q);
[U, S, V] = svds(Q,r)
D = diag(S);
E=inf;
while 1 %error SVD
    [U,S,V] = svds(Q,r);
    X=U*S*V';
    Nor=norm(Q-X,2)^2;
   if(Nor<E)
       E=Nor;
   else
       break;
   end
end

eigs = D.^2;%eigenvales square
var_seq =cumsum(eigs);
tot_var = sum(eigs);
exp_var = var_seq/tot_var;
Porc = 100*exp_var
```

4 .- USING SVD TO ANALYZE IMAGES

```
close all;
clear all;
load('mandril.mat', 'X','map');
image(X); colormap(map)
[U, S, V] = svd(double(X));
stem(diag(S)); grid;
subplot(2,2,1);
Aproximation(1,S,U,V);
title('SVD for k = 1');
subplot(2,2,2);
Aproximation(6,S,U,V);
title('SVD for k = 6');
subplot(2,2,3);
Aproximation(11,S,U,V);
title('SVD for k = 11');
subplot(2,2,4);
Aproximation(31,S,U,V);
title('SVD for k = 31');
%subplot(2,2,1);
%residuals(1,S,U,V,X);
%title('Residuals for k = 1');
%subplot(2,2,2);
%residuals(6,S,U,V,X);
%title('Residuals for k = 6');
%subplot(2,2,3);
%residuals(11,S,U,V,X);
%title('Residuals for k = 11');
%subplot(2,2,4);
%residuals(31,S,U,V,X);
%title('Residuals for k = 31');
h=gcf;
set(h,'PaperPositionMode','auto');
set(h,'PaperOrientation','landscape');
set(h,'Position',[10 10 1200 850]);
print(gcf, '-dpdf', 'Residuals.pdf')
[k1 R] =fnorm(1,S,U,V,X)
[k6 R] = fnorm(6,S,U,V,X)
[k11 R] = fnorm(11,S,U,V,X)
[k31 R] = fnorm(31,S,U,V,X)
function X_k = Aproximation(k,S,U,V)
for j = 1:k
C=S;
C(j+1:end,:)=0;
C(:,j+1:end)=0;
X_k = U*C*V';
end
X_k=uint8(X_k);
imagesc(X_k), colormap(gray);
end
function E = residuals(k,S,U,V,X)
for j = 1:k
C=S;
C(j+1:end,:)=0;
C(:,j+1:end)=0;
X_k = U*C*V';
end
E = X -(X_k)
stem(diag(E)); grid;
end
function [N  RE]= fnorm(k,S,U,V,X)
for j = 1:k
D = diag(S);
C=S;
C(j+1:end,:)=0;
C(:,j+1:end)=0;
X_k = U*C*V';
end
N = norm(X-(X_k),2);
RE = abs((S(k+1) - N)/S(k+1));
end
```

```
k1 =

   1.2891e+04


R =

   Inf


k6 =

   3.5379e+03


R =

   Inf


k11 =

   2.8201e+03


R =

   Inf


k31 =

   1.9855e+03


R =

   Inf
```


50
100
150
200
250
300
350
400
450
50
100
150
200
250
300
350
400
450
500


×10^4
6
5
4
3
2
1
0
0
100
200
300
400
500