Blind Source Separation

HW6-Section-3

Mohammadreza Arani :::::::::::: 810100511

1402/02/20

```
clear; clc; close all;
```

Part-1:

بخش سوم)

در این تمرین می خواهیم دو روش MOD و K-SVD را برای جداسازی کور منابع اسپارس یا به عبارت دیگر یادگیری دیکشنری برای نمایش اسپارس سیگنال ها پیاده سازی کنیم.

ماتریس دیکشنری با ابعاد 10×40، ماتریس منابع با ابعاد 40×1500 و ماتریس مشاهدات با ابعاد 10×1500 در یک فایل با نام hw6-part3.mat قرار داده شده است.

۱- معیار mutual coherence را محاسبه و گزارش کنید.

```
Data_Q3  = load("hw6-part3.mat");
D_Part_1 = Data_Q3.D;
% Calculation of MU based on implemented function in previous Section:
[MU_out, MU_Index_Out] = MU_Calc_Mine(D_Part_1);
disp("Mutual Coherence is : "+ MU_out);
```

```
Mutual Coherence is : 0.86803
```

```
disp(" Maximum Coherence happens for column "+MU_Index_Out(1)+" and column "+MU_Index_Out(2))
```

```
 Maximum Coherence happens for column 13 and column 38
```

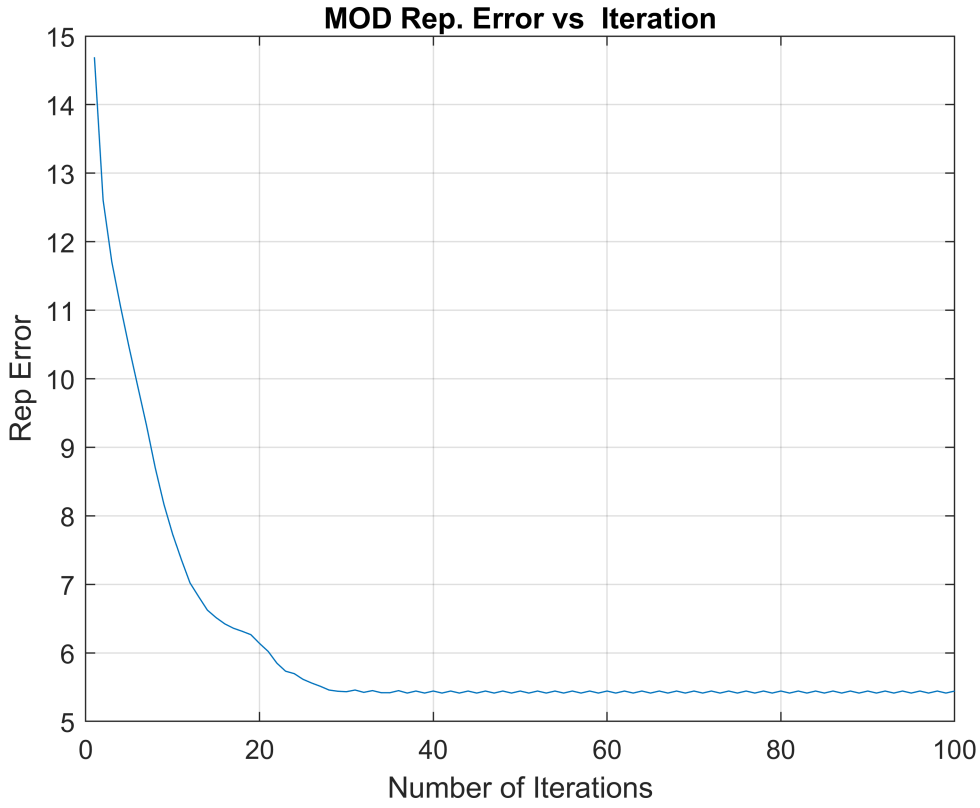
```

```

Part-2:

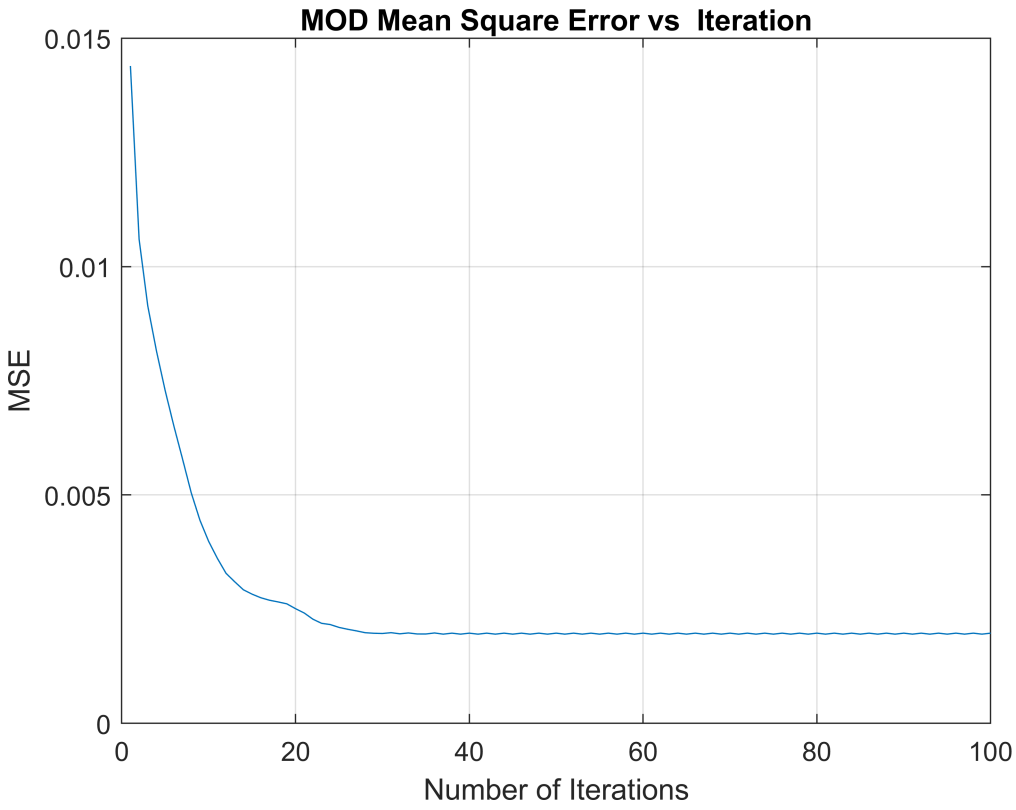
ماتریس مشاهدات را داریم، روش های MOD و K-SVD را روی ماتریس مشاهدات اعمال کنید و برای دیکشنری $\hat{D}$ و تخمین منابع $\hat{S}$ را به دست آورید. شایان ذکر است sparsity level برابر $N_0=2$ است و د از یک الگوریتم sparse recovery استفاده کنید از OMP استفاده کنید.

MOD

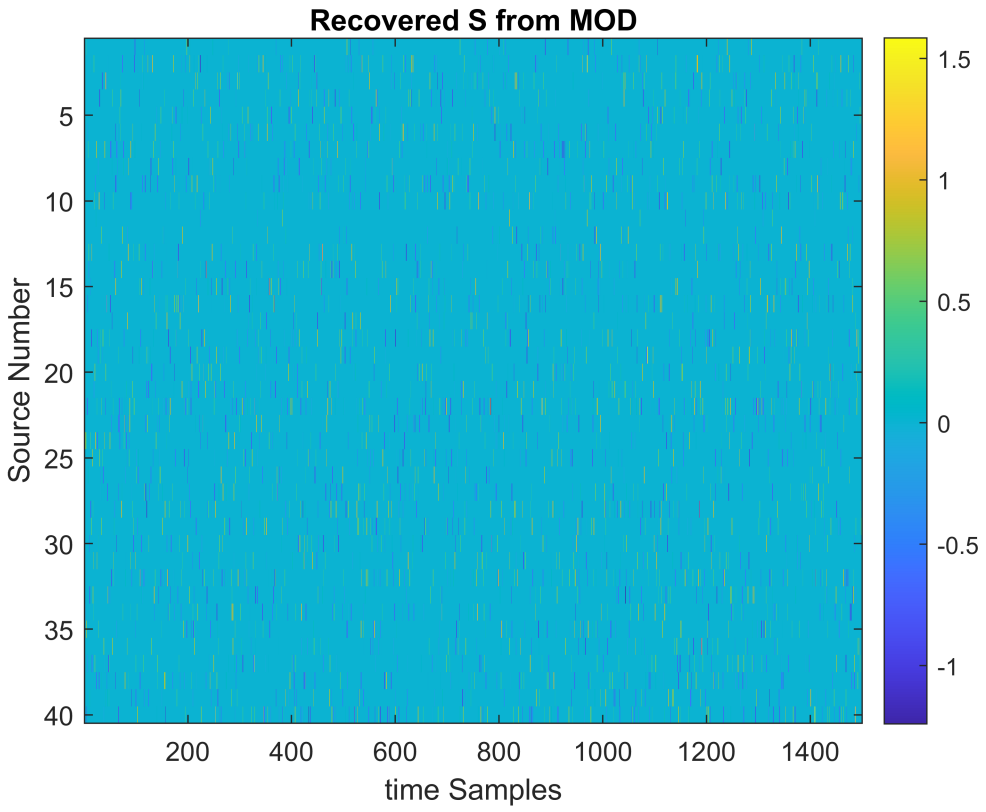
```
X_part_1   = Data_Q3.X;
S_part_1   = Data_Q3.S;
N0_Part_1  = Data_Q3.N0;

S_X_p1 = size(X_part_1);
S_S_p1 = size(S_part_1);

D_hat_MOD = randn(S_X_p1(1) , S_S_p1(1)) ;
D_hat_MOD = D_hat_MOD ./ sqrt(sum(D_hat_MOD.^2,1));

T = S_S_p1(2);
S_hat_MOD = randn(S_S_p1);

IterMax = 100;
MSE_MOD = zeros(1,IterMax);
Rep_MOD = MSE_MOD;
MOD_delta_Time = MSE_MOD;


for i=1:IterMax
    tic;
    % First Step of Alternation Minimization:
    for t=1:T
        S_hat_MOD(:,t) = OMP_Mine_CAlc( X_part_1(:,t)  , D_hat_MOD , N0_Part_1);
    end
    % Second Step of Alternation Minimization:
    D_hat_MOD = X_part_1*pinv(S_hat_MOD);                % Second Step of Altrnation Minimizati
    D_hat_MOD = D_hat_MOD ./ sqrt(sum(abs(D_hat_MOD).^2));  % Normalize Each Column of D
    % Calculate the MSE:
    MSE_MOD(i) = mse(X_part_1 , D_hat_MOD*S_hat_MOD);
    Rep_MOD(i) = norm(X_part_1 - D_hat_MOD*S_hat_MOD,"fro");
    MOD_delta_Time(i) = toc;
end

figure()
plot(Rep_MOD)
grid on
xlabel("Number of Iterations")
ylabel("Rep Error")
title("MOD Rep. Error vs  Iteration")
```




```
figure()
plot(MSE_MOD)
grid on
xlabel("Number of Iterations")
ylabel("MSE")
title("MOD Mean Square Error vs  Iteration")
```




```
figure()
imagesc(S_hat_MOD)
title("Recovered S from MOD")
xlabel("time Samples")
ylabel("Source Number")
colorbar
```

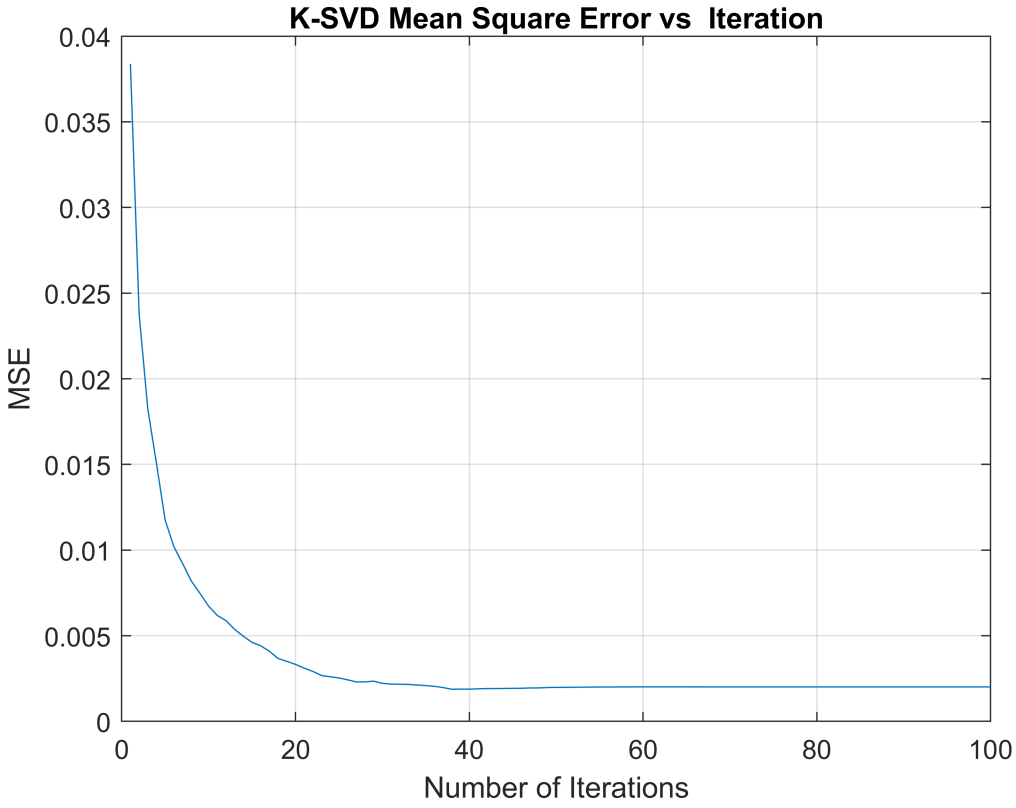


K-SVD:

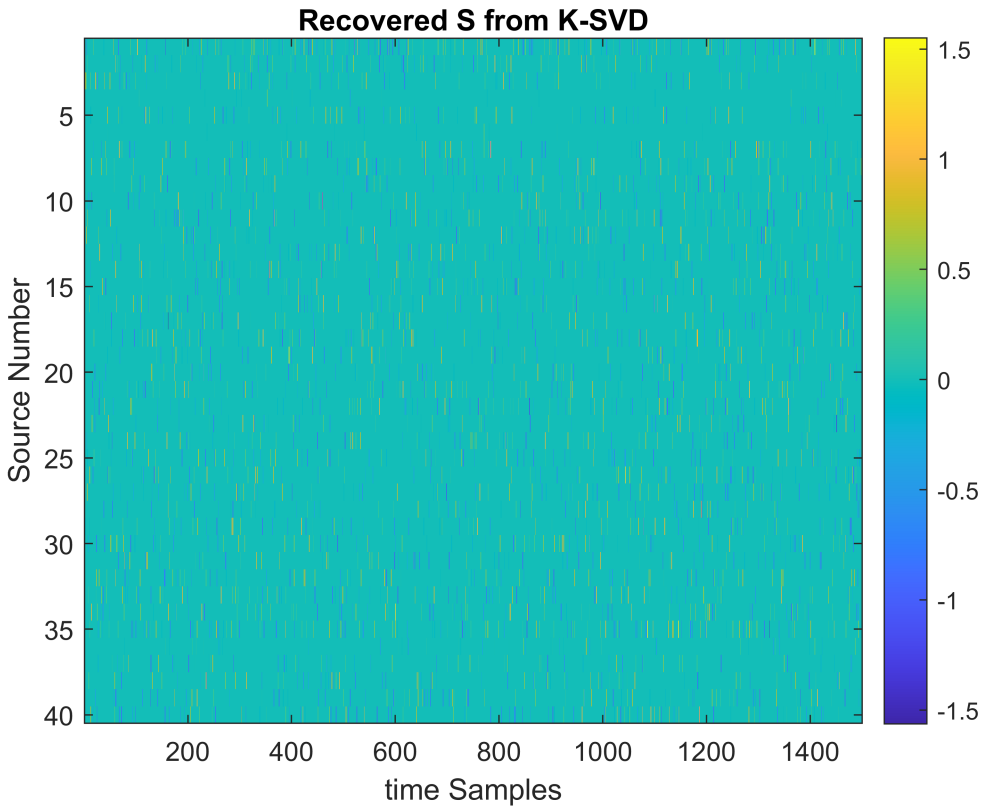
```
D_hat_K_SVD = randn(S_X_p1(1) , S_S_p1(1)) ;
T = S_S_p1(2);
S_hat_K_SVD = randn(S_S_p1);

IterMax = 100;
MSE_K_SVD = zeros(1,IterMax);
Rep_K_SVD = MSE_K_SVD;
K_SVD_delta_Time = MSE_K_SVD;


for i=1:IterMax
    tic;
    % First Step of Alternation Minimization:
    for t=1:T
        S_hat_K_SVD(:,t) = OMP_Mine_CAlc( X_part_1(:,t)  , D_hat_K_SVD , N0_Part_1);
    end
    % Second Step of Alternation Minimization:
```

```
    for n=1:S_S_p1(1)

        % Choose corresponding Index:
        D_ind_SVD = 1:S_S_p1(1);
        D_ind_SVD(n) = [];

        % Calculate the Residual X:
        X_res = X_part_1 - D_hat_K_SVD(:, D_ind_SVD)*S_hat_K_SVD(D_ind_SVD,:) ;
        [U,S,V] = svd(X_res);

        % Sort:
        [S_sorted, idx] = sort(diag(S), 'descend');

        % Reorder the columns of U and rows of V based on the sorted indices
        U_sorted = U(:, idx);
        V_sorted = V(:, idx);

        % Assign U and V to D and S:
        D_hat_K_SVD(:,n) = U_sorted(:,1);
        Zer0Index = find(abs(S_hat_K_SVD(n,:))<1e-6);
        S_hat_K_SVD(n,:) = S_sorted(1)*V_sorted(:,1);
        S_hat_K_SVD(n,Zer0Index) = 0;
    end
    % Calculate the MSE
    MSE_K_SVD(i) = mse(X_part_1 , D_hat_K_SVD*S_hat_K_SVD);
    Rep_K_SVD(i) = norm(X_part_1 - D_hat_K_SVD*S_hat_K_SVD,"fro" );
    K_SVD_delta_Time(i) = toc;
end
```

```
figure()
plot(MSE_K_SVD)
grid on
xlabel("Number of Iterations")
ylabel("MSE")
title("K-SVD Mean Square Error vs  Iteration")
```

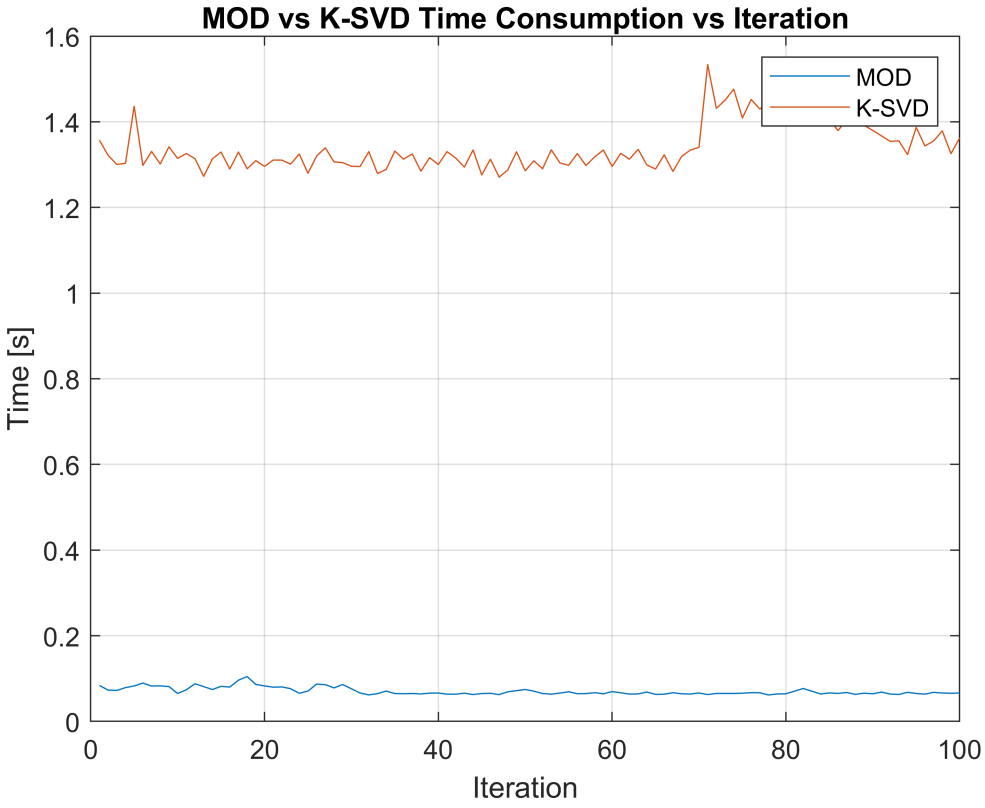



```
figure()
imagesc(S_hat_K_SVD)
title("Recovered S from K-SVD")
xlabel("time Samples")
ylabel("Source Number")
colorbar
```

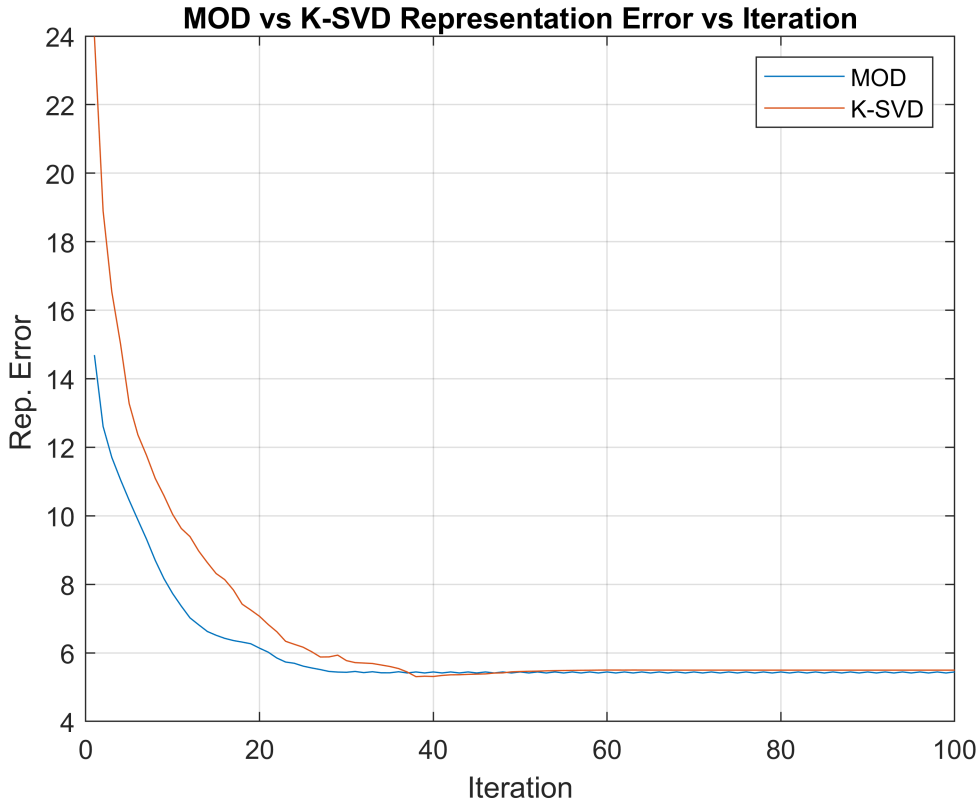


Comparison of MOD and K-SVD

```
figure()
plot(MOD_delta_Time)
hold on
plot(K_SVD_delta_Time)
title("MOD vs K-SVD Time Consumption vs Iteration")
grid on
xlabel("Iteration")
ylabel("Time [s]")
legend("MOD", "K-SVD")
hold off
```




```
figure()
plot(Rep_MOD)
hold on
plot(Rep_K_SVD)
hold off
title("MOD vs K-SVD Representation Error vs Iteration")
grid on
xlabel("Iteration")
ylabel("Rep. Error ")
legend("MOD", "K-SVD")
```



Part-5:

۵- برای مقایسه ی کیفیت یادگیری دیکشنری، برای هر روش به صورت جداگانه، به ترتیب از ستون (اتم) اول تا ستون (اتم) آخر $\widehat{D}$ ، قدر مطلق correlation هر اتم $\widehat{D}$ را با همه ی اتم های D محاسبه کنید. در صورتی که ماکزیمم این مقادیر از 0.98 بیشتر بود، فرض کنید اتمی از D که ماکزیمم correlation را داده است به درستی recover شده است. توجه داشته باشید که برای تکرار این روش و محاسبات بعدی، اتمی از D که recover شده است را حذف کنید. برای هر روش، چند درصد از ستون های ماتریس D را به درستی Recover کردید؟ به این درصد successful recovery rate می گوییم.

```
%  [index_remove_Recovered_SVD,counter_SVD] = Recovery_Calc_D(D_hat_K_SVD , D_Part_1 )
[D_hat_K_SVD,Succ_Rec_Num_K_SVD] = Recovery_Calc_D(D_hat_K_SVD , D_Part_1 )
```

```
D_hat_K_SVD = 10×40
    0.1782    0.0616    0.0067    0.1625    0.3639    0.4327    0.0122    0.1649 ···
   -0.6215   -0.1760   -0.3174    0.0084   -0.2324   -0.1279   -0.4431    0.0825
    0.2242   -0.0384   -0.3175    0.2171   -0.3800    0.1295    0.3095    0.3399
    0.2124   -0.1355   -0.5826    0.5912    0.0028    0.3417    0.2711   -0.0910
   -0.2676   -0.1822   -0.1205   -0.3646    0.2652    0.3546   -0.1804   -0.4486
    0.3716   -0.0659   -0.3122   -0.1084    0.6184    0.0031    0.0841    0.0360
   -0.3596   -0.4665   -0.1029   -0.1599    0.0002    0.1226   -0.2405    0.2894
    0.2836   -0.1813   -0.0590   -0.0752    0.1848    0.1050   -0.2951    0.2645
   -0.1171    0.7003   -0.5078    0.1144   -0.3329   -0.2997   -0.6311   -0.4214
    0.2314   -0.4085   -0.2739    0.6228   -0.2672    0.6490    0.2267    0.5553
Succ_Rec_Num_K_SVD = 35
```

```
disp(Succ_Rec_Num_K_SVD/size(D_Part_1,2)*100+" % of the Columns of D where recovered successful
```

```
87.5 % of the Columns of D where recovered successfully!
```

```
[D_hat_MOD,Succ_Rec_Num_MOD] = Recovery_Calc_D(D_hat_MOD , D_Part_1 )
```

```
D_hat_MOD = 10×40
   -0.1758   -0.0923    0.0390    0.2260   -0.3966   -0.4335    0.3994   -0.1268 ···
    0.6312    0.2277   -0.3149    0.1461    0.2639    0.1159    0.3185   -0.0980
   -0.2250   -0.0150   -0.3193    0.1594    0.3539   -0.1148   -0.1668   -0.3396
   -0.1637    0.1880   -0.5239    0.0376   -0.0423   -0.3057   -0.0217    0.1679
    0.2384    0.2389   -0.1397    0.0554   -0.2338   -0.4027    0.3025    0.4656
   -0.4267    0.0726   -0.3082    0.1902   -0.6326   -0.0306    0.2271   -0.0517
    0.3760    0.4176   -0.1242   -0.5923    0.0434   -0.1187   -0.4292   -0.3287
   -0.2806    0.1437   -0.0507    0.0940   -0.2106   -0.1346   -0.0215   -0.2818
    0.1273   -0.7189   -0.5549    0.2522    0.3271    0.3057   -0.1286    0.4117
   -0.1343    0.3604   -0.2871    0.6620    0.1946   -0.6350    0.6054   -0.5044
Succ_Rec_Num_MOD = 34
```

```
% [index_remove_Recovered_MOD,counter_MOD] = Recovery_Calc_D(D_hat_MOD, D_Part_1 )


disp(Succ_Rec_Num_MOD/size(D_Part_1,2)*100+" % of the Columns of D where recovered successfully
```

```
85 % of the Columns of D where recovered successfully!
```

Part-6:

۶- ابهام ترتیب و scale منابع را برطرف کرده و سپس عبارت زیر را برای هر دو روش گزارش کنید.

$$E = \frac{\|\hat{S} - S\|_F^2}{\|S\|_F^2}$$

```
S_hat_K_SVD_OMP = S_hat_K_SVD;
S_hat_MOD_OMP = S_hat_MOD;
for t=1:T
    S_hat_K_SVD_OMP(:,t) = OMP_Mine_CAlc( X_part_1(:,t)  , D_hat_K_SVD , N0_Part_1);
    S_hat_MOD_OMP(:,t) = OMP_Mine_CAlc( X_part_1(:,t)  , D_hat_MOD , N0_Part_1);
end
[Erroes_S_Hat_K_SVD,S_Hat_Chosen_K_SVD]  = Perm_AMP_Disamb(S_hat_K_SVD_OMP,S_part_1);

disp("Error of S_K_SVD equals to: "+ min(Erroes_S_Hat_K_SVD));
```

```
Error of S_K_SVD equals to: 1.4798
```

```
[Erroes_S_Hat_MOD,S_Hat_Chosen_MOD]  = Perm_AMP_Disamb(S_hat_MOD_OMP,S_part_1);
disp("Error of S_MOD equals to: "+ min(Erroes_S_Hat_MOD));
```

```
Error of S_MOD equals to: 1.8142
```

```

```

```
function [Erroes_B,B_Hat_Chosen]  = Perm_AMP_Disamb(B_Hat,S_Amp) %% Perm_AMP_Disamb
Perm_1 = B_Hat;
Perm_2 = -B_Hat;
Perm_3 = B_Hat([end, 1:end-1],:);
Perm_4 = -B_Hat([end, 1:end-1],:);
Perms = { Perm_1,Perm_2,Perm_3,Perm_4  };
% Estimate S:
S_hat_1 = Perm_1;
S_hat_1_Normalised = S_hat_1/norm(S_hat_1,'fro');
S_hat_2 = Perm_2;
S_hat_2_Normalised = S_hat_2/norm(S_hat_2,'fro');
S_hat_3 = Perm_3;
S_hat_3_Normalised = S_hat_3/norm(S_hat_3,'fro');
S_hat_4 = Perm_4;
S_hat_4_Normalised = S_hat_4/norm(S_hat_4,'fro');
% Calc Error:
S_Amp_Normalized = S_Amp/norm(S_Amp,'fro');

Error_1 = (norm((S_hat_1_Normalised)-(S_Amp_Normalized),'fro'))^2/norm((S_Amp_Normalized),"fro'
Error_2 = (norm((S_hat_2_Normalised)-(S_Amp_Normalized),'fro'))^2/norm((S_Amp_Normalized),"fro'
Error_3 = (norm((S_hat_3_Normalised)-(S_Amp_Normalized),'fro'))^2/norm((S_Amp_Normalized),"fro'
Error_4 = (norm((S_hat_4_Normalised)-(S_Amp_Normalized),'fro'))^2/norm((S_Amp_Normalized),"fro'

Erroes_B = [Error_1,Error_2,Error_3,Error_4];
[ ~, index] = min(Erroes_B);
B_Hat_Chosen = Perms{index};
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function  [D_Hat_sorted,Counter] = Recovery_Calc_D(D_hat , D )
%index_remove_Recovered

% D_hat = D_KSVD;
D_Hat_sorted = D_hat;
Counter = 0;
for i=1:size(D,2)
    for j = 1 : size(D_hat,2)
        if(abs(D(:,i)'*D_hat(:,j)) > 0.98)
            Counter = Counter + 1;
            D_Hat_sorted(:,i) = D_hat(:,j);
            D_hat(:,j)=[];
            break;
        end
    end
end
```

```
%     [~ , Col_D] = size(D_hat);
%     Temp_D_check = D;
%
%     index_remove_Recovered =[];
%     counter = 0;
%     for i=1:Col_D
%         Cor_Col_D =  D_hat(:,i)'*Temp_D_check ;
%         %max(abs(Cor_Col_D))
%         if(max(abs(Cor_Col_D)) > 0.98) % Check to see whether we have recovered this column o
%             [~ , index_remove_Recovered(counter+1)] = max(Cor_Col_D);
%             Temp_D_check(index_remove_Recovered) = [];
%             counter = counter+1;
%         end
%
%     end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [MU_out, MU_Index_Out] = MU_Calc_Mine(D)

Temp     = (ones(size(D'*D)) - eye(size(D'*D))).* abs( D'*D );
[MU_Vals , MU_Index] =  max(Temp)  ;
[MU_out , MU_Index2]  = max(MU_Vals);

MU_Index_Out = [MU_Index2 , MU_Index(MU_Index2)];
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function  s_OMP = OMP_Mine_CAlc( x  , D , N0)

    xr = x;
    Dr = D;
    idx_OMP = zeros(1, N0);
    for i = 1 : N0
        [~, idx_OMP(i)] = max(abs(xr'*Dr),[],'omitnan');
        xr = xr - x'*Dr(:, idx_OMP(i))*Dr(:,idx_OMP(i));
        if i > 1
            xr = xr - D(:,idx_OMP(1:i))*(pinv(D(:,idx_OMP(1:i)))*xr);
        end
        Dr(:,idx_OMP(i)) = nan;
    end

    s_OMP = zeros(size(D, 2), 1);
    s_OMP(idx_OMP) = pinv(D(:, idx_OMP))*x;
```

```
    %
    % xr = X;
    % [~ , Col_D] = size(D);
    % Chosen_IDX = inf+zeros(1,Col_D);
    % S_hat = zeros(Col_D,1);
    % for  i=1:N0
    %
    %     B = D;
    %
    %     if(i>1)
    %         B(:,Chosen_IDX(1:i-1)) = []; % Removing previously chosen Indices!
    %     end
    %
    %     Corr_matrix = abs(repmat(xr,1,Col_D-i+1).*B);
    %     Corr_sum = sum(Corr_matrix,1); % Summation for each Column -->> a Row Vector
    %     [Value,Idx] = max(Corr_sum); % Choosing Best Fitted
    %
    %     if(sum(Idx>Chosen_IDX)>0)
    %         Idx  = Idx + sum(Idx>Chosen_IDX) ; % Update the Index with respect to the Origina
    %     end
    %
    %     Chosen_IDX(i) =  Idx;
    %     xr = xr - X'*D(:,Idx)*D(:,Idx); % xr = x  - <x,di>di
    %
    %     % Update Coefficients:
    %     if(i>1)
    %         D_sub_omp = D(:,Chosen_IDX(1:i)) ;
    %         xr = xr - D(:,Chosen_IDX(1:i))*(pinv(D_sub_omp)*xr);
    %     end
    %     % Stop Criteria:
    % %     if(norm(x - D(:,Chosen_IDX_3(1:i)) ,2)<thresh )
    % %         break;
    % %     end
    %     S_hat(Chosen_IDX(i)) = pinv(D(:,i))*X;
    % end

% delta_t_OMP = toc;
% disp("Elapsed Time (OMP): "+ delta_t_OMP+"(s)");
end
```