به نام خدا

# Blind Source Separation (BSS)

تکلیف شماره

**6**


محمدرضا آرانی

810100511

دانشگاه تهران


1402/02/20

# BSS-HW7-2023

جدول محتویات

# بخش اول:

در این تمرین می خواهیم روش single channel sparse blind deconvolution را پیاده سازی کنیم. همان طور که در کلاس بحث شد این روش اساس multichannel sparse blind deconvolution را تشکیل می دهد.

دو داده ی تک کاناله نویزی $x_1$ و $x_2$ و یک داده ی دو کاناله ی X در ماتریس hw7.mat در اختیار شما قرار داده شده است.

## قسمت-1:

الگوریتم مورد نظر برای این قسمت، پیاده‌سازی Sparse Blind Deconvolution در حوزه‌ی زمان است. مراحل این پیاده‌سازی و مدل کردن مسئله به صورت زیر است:

$$[\hat{s}, \hat{\text{Ш}}] = argmin \sum_t \big(x(t) - s(t) * \text{Ш}(t)\big)^2$$

$$S.t. \qquad \text{Ш}(t) = \sum_k \big(\alpha_k \delta(t - \tau_k)\big)$$

$$||s||_2 = 1$$

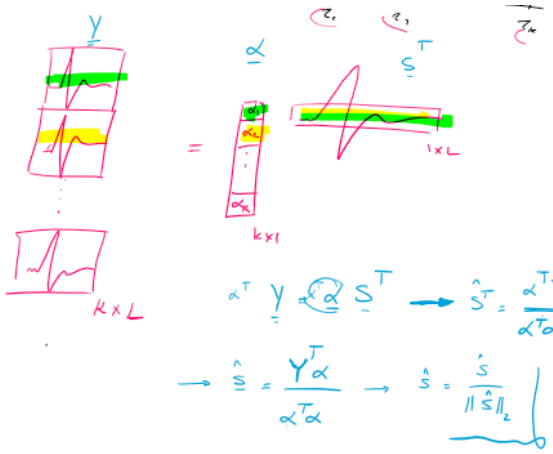
$$|\tau_i - \tau_j| > L$$

$$\alpha_k > 0$$

برای حل این مسئله که در بالا نشان داده شده است، بازهم از Alternation Minimization استفاده می‌کنیم:

Step-1:

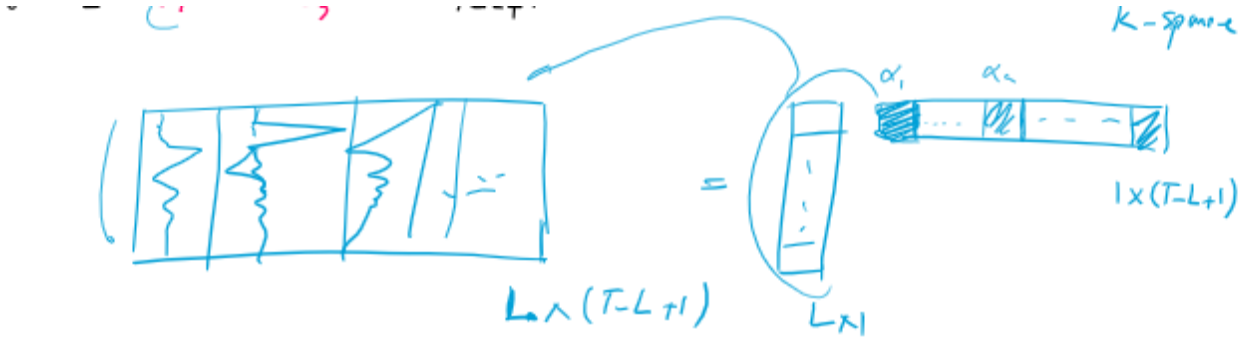
$\text{Ш}(t)$ is fixed → obtain $\hat{s}$ by $\hat{s} = \frac{\alpha^T Y}{\alpha^T \alpha}$; and then $\hat{s} = \frac{\hat{s}}{||\hat{s}||_2}$



شکل 1

Step-2:

$\hat{s}$ is fixed → obtain $\hat{\text{Ш}}(t)$ by $\hat{b} = \frac{\hat{s}^T Z}{\hat{s}^T \hat{s}}$;



شکل 2

در واقع ماتریس Y از کنار هم قرار دادن K تا ستون که هر یک L تا نمونه از قسمت‌های محتمل وقوع سیگنال در ماتریس مشاهدات هستند می‌باشد.

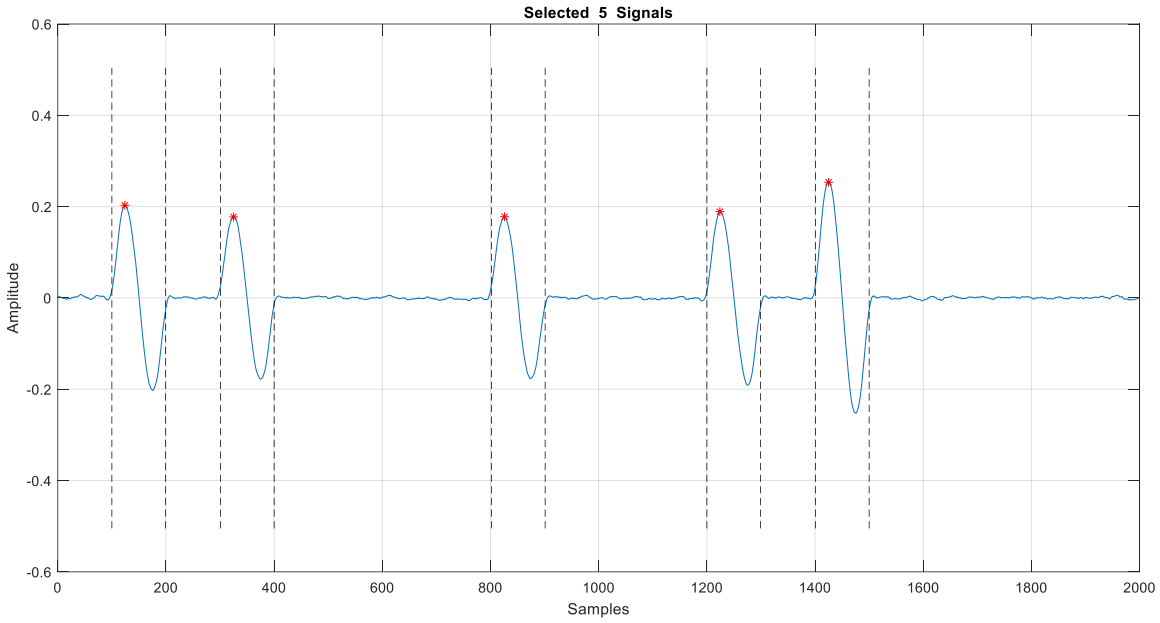
همچنین ماتریس Z نیز از کنارهم قراردادن نمونه‌های با طول L به صورت شیفت یافته از روی سیگنال مشاهدات است. به این ترتیب یک ماتریس با تعداد ردیف‌های L و تعداد ستون‌های T-L داریم.

پیاده‌سازی این روش در متلب به صورت زیر است:

```
disp(char(hex2dec('0428'))+"(t) is fixed & then S is Fixed");
Ш(t) is fixed & then S is Fixed
IterMax=50;
rng(1);
Alpha_vec = randn(K,1);
Z = buffer(y1 ,L, L-1 ,'nodelay');
T = length(y1);
y_hat = zeros(size(y1));
for i=1:IterMax
    % Step-1: Shai is fixed
    S_hat_time = Signal_Selected_k*pinv(Alpha_vec');
    % Normalize S hat
    S_hat_time = S_hat_time/norm(S_hat_time);
    % Step-2:  S is fixed
    Chosen_Index_Z = zeros(1,K);
    temp_Z = Z;
    b = zeros(1,T);
    for k=1:K
       [~,index_max]  = max(   abs( S_hat_time'*temp_Z  ));
       Chosen_Index_Z(1,k) = index_max;
       if(index_max-L/4+1<0)
        Index_to_Change = [1:L];
       else
        Index_to_Change = [index_max-L/4+1:index_max+3*L/4];
       end
       b(1,index_max) =  S_hat_time'*temp_Z(:,index_max)/( S_hat_time'*S_hat_time  );
       Alpha_vec(k,1) = b(1,index_max);
       Signal_Selected_k(:,k) = y1(Index_to_Change)  ;%Chosen_Index_Z(1,k)-
L/4+1:Chosen_Index_Z(1,k)+3*L/4
       y_hat(1,Index_to_Change) = Signal_Selected_k(:,k);
       temp_Z(:, Index_to_Change ) = 0;
    end
end


figure()
plot(y_hat);
grid on
```

```
hold on
plot(y1,'r--');
title("x^ vs x")
xlabel(" Samples")
legend("x hat" , "x");
hold off
```

در واقع برای انتخاب مکان‌های اولیه، ابتدا سیگنال را smooth کرده و سپس پیک‌های آن را انتخاب کردیم و در نتیجه مکان‌های نسبی اولیه انتخاب به دست آمدند:



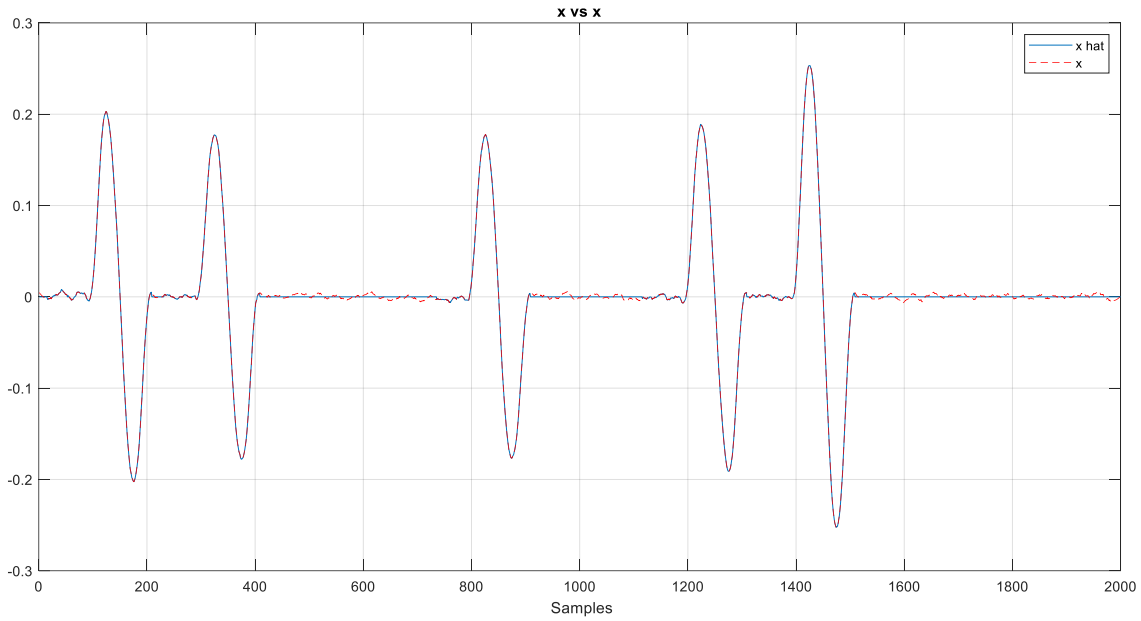

شکل 3

```
windowSize = 31;
polyOrder = 2;
y1 = sgolayfilt(x1, polyOrder, windowSize);
figure()
plot(y1)
[Peaks_y1 , Locs_y1] = findpeaks(y1);


Good_Indices_Peaks =   find( ((Peaks_y1)>mean(abs(y1))) );

plot(Locs_y1(Good_Indices_Peaks),Peaks_y1(Good_Indices_Peaks),'r*');


% Loc_target_k = zeros(1,K);
Loc_target_k = Locs_y1(Good_Indices_Peaks);
Selected_Indices = zeros(2,K) ;
```

```
Signal_Selected_k = zeros(L,K);


for k=1:K
    Signal_Selected_k(:,k) = x1(Loc_target_k(1,k)-L/4+1:Loc_target_k(1,k)+3*L/4);
    Selected_Indices(:,k) = [ Loc_target_k(1,k)-L/4+1 ; Loc_target_k(1,k)+3*L/4 ];
    plot(Selected_Indices(1,k)*ones(1,10),linspace(-2*abs(min(y1)),2*max(abs(y1)),10)
,'black--' );
    plot(Selected_Indices(2,k)*ones(1,10),linspace(-2*abs(min(y1)),2*max(abs(y1)),10)
,'black--' );


end
grid on
title("Selected  "+K+"  Signals");
xlabel("Samples")
ylabel("Amplitude")
hold off
```

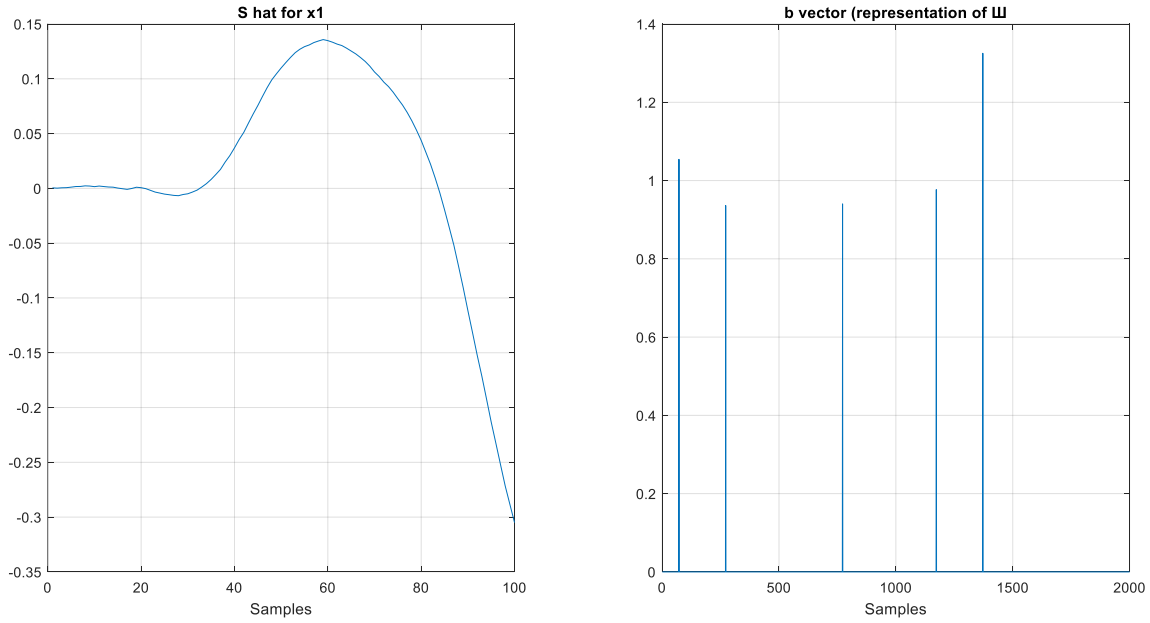
در بالا برای smooth کردن یک سیگنال، از بین روش‌های موجود بسیار، روش فیلتر کردن توسط فیلتر **savitky-Golai** را انتخاب کردیم چرا که نتایج بهتری میداد و پارامترهای قابل تنظیم آن دم دست تر بودند. انتخاب‌های دیگر مانند انتخاب فیلتر Lowpass و یا استفاده از Moving Average فیلتر نیز مطرح بودند.

پس از همگرایی در 50 تکرار به خروجی زیر می‌رسیم:




شکل 4

مشاهده می‌شود به خوبی سیگنال بازیابی و بازسازی شده است و مسئله‌ی Blind Deconvolution در حوزه‌ی زمان برای یک تک کانال حل شده است.

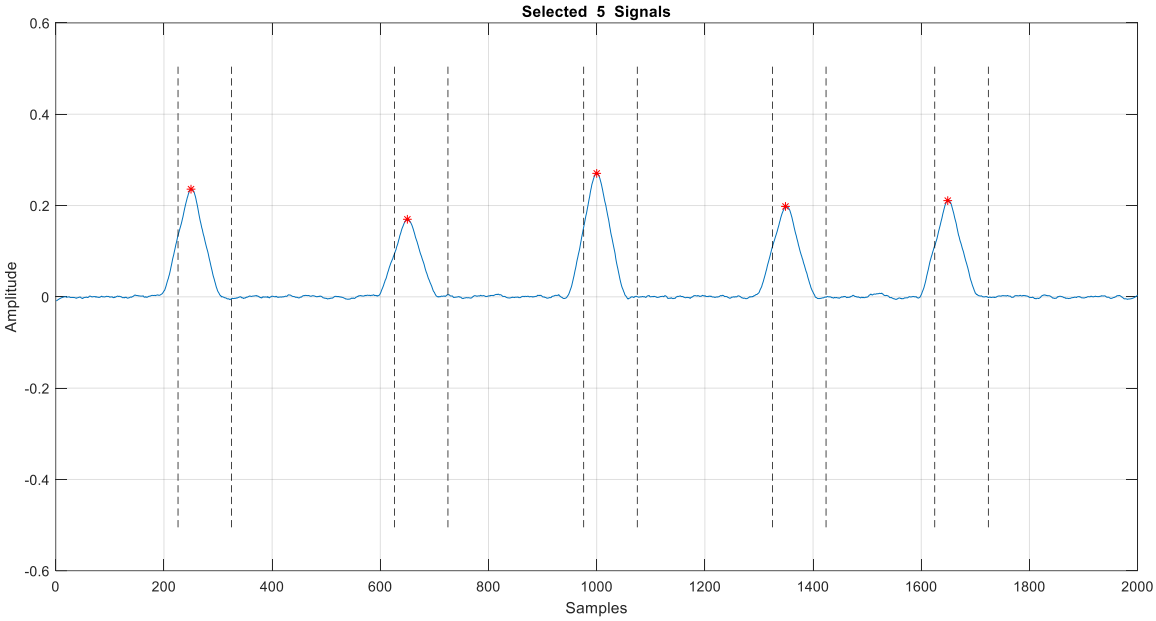



شکل 5

## قسمت-2:

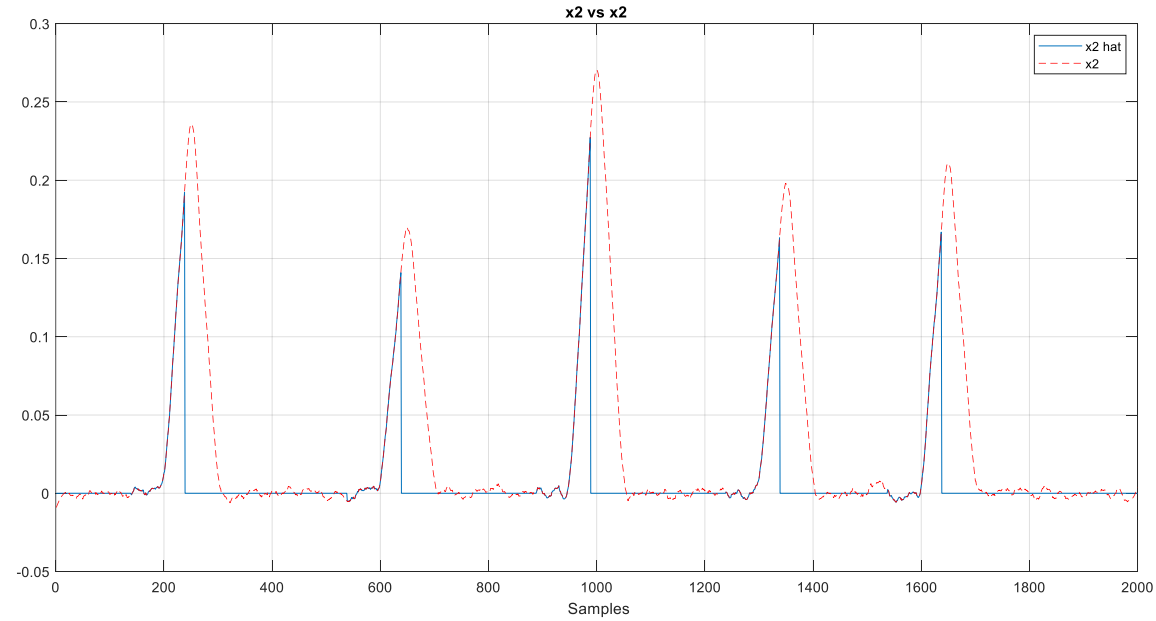
با استفاده از همین روش، برای داده‌ی سیگنال x2 نیز داریم:

```
x2 = Data_hw7.x2;


windowSize = 35;
polyOrder = 2;
y2 = sgolayfilt(x2, polyOrder, windowSize);


figure()
plot(y2)
[Peaks_y2 , Locs_y2] = findpeaks(y2);


Good_Indices_Peaks =   find( ((Peaks_y2)>mean(abs(y2))) );
hold on
plot(Locs_y2(Good_Indices_Peaks),Peaks_y2(Good_Indices_Peaks),'r*');


% Loc_target_k = zeros(1,K);
Loc_target_k = Locs_y2(Good_Indices_Peaks);
Selected_Indices = zeros(2,K) ;
Signal_Selected_k = zeros(L,K);


for k=1:K
    Signal_Selected_k(:,k) = x2(Loc_target_k(1,k)-L/4+1:Loc_target_k(1,k)+3*L/4);
    Selected_Indices(:,k) = [ Loc_target_k(1,k)-L/4+1 ; Loc_target_k(1,k)+3*L/4 ];
    plot(Selected_Indices(1,k)*ones(1,10),linspace(-2*abs(min(y1)),2*max(abs(y1)),10)
,'black--' );
    plot(Selected_Indices(2,k)*ones(1,10),linspace(-2*abs(min(y1)),2*max(abs(y1)),10)
,'black--' );


end
grid on
title("Selected  "+K+"  Signals");
xlabel("Samples")
ylabel("Amplitude")
hold off
```



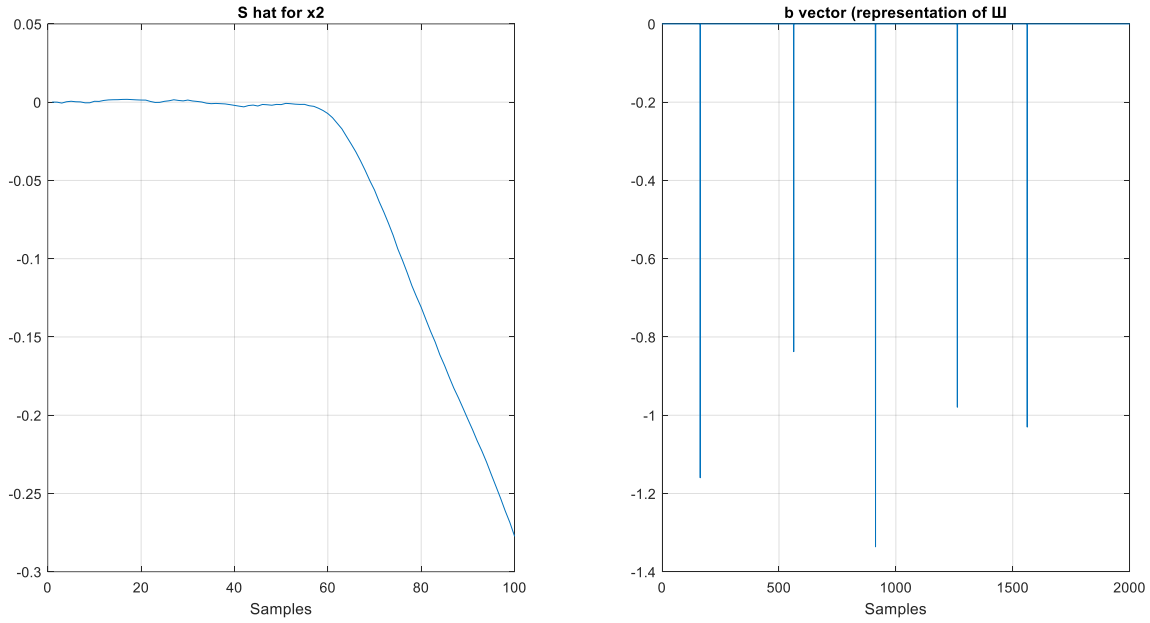

شکل 6

برای داده‌ی x2 هم به همین ترتیب داریم:




شکل 7

مشاهده می‌شود به خوبی سیگنال بازیابی و بازسازی شده است و مسئله‌ی Blind Deconvolution در حوزه‌ی زمان برای یک تک کانال حل شده است.



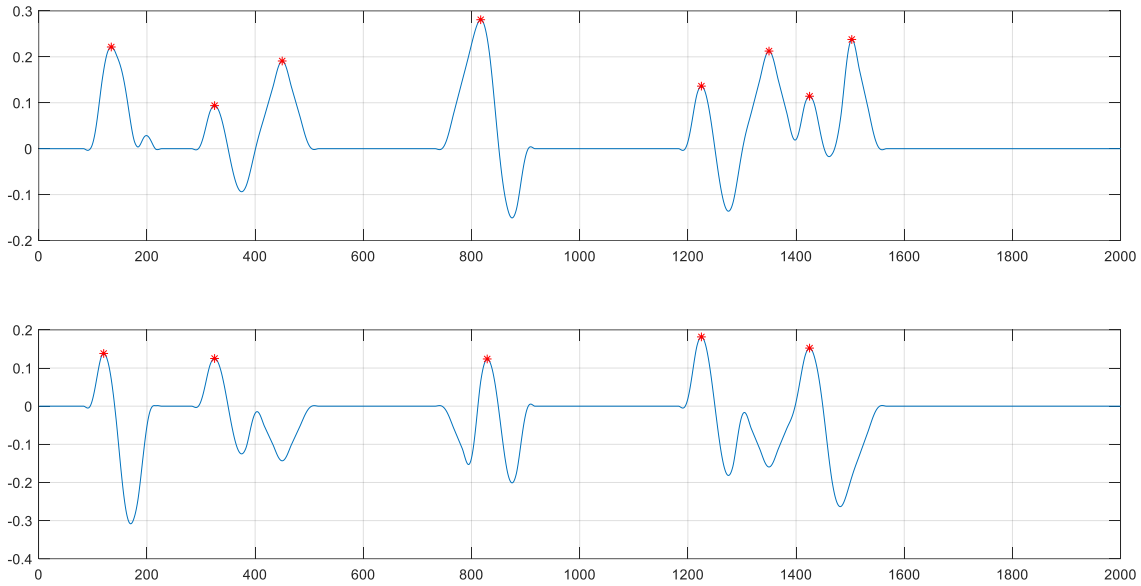

شکل 8

## قسمت-3:

۳- ابتدا داده ی دو کاناله ی X را رسم کرده و به صورت چشمی آن را بررسی کنید. روش multichannel sparse blind deconvolution را روی آن اعمال کرده و دو شکل موج اسپایکی و دو سری زمانی که این داده را به وجود آورده اند به دست آورید. طول شکل موج ها را برابر $L = 100$ سمپل و تعداد رخداد ها را برای هر کدام برابر $K = 5$ در نظر بگیرید.

نکته‌ی مهم در این قسمت، تخمین درست ماتریس میکس برای مانال‌ها در کنار انجام درست *Single Channel Blind Deconvolution* است.

بررسی چشمی داده‌های این سیگنال به صورت زیر انجام شده است:



شکل 9

در کانال اول، از بین این 8 عدد پیک شناخته شده، 5 تای آنها باید انتخاب شود.

این فرآیند در تابع متناسب با *single channel blind deconvolution* اتفاق می‌افتد. این تابع به صورت زیر است:

```
function  [ y_hat , b, S_hat_time ]  = Single_Ch_Time_Dec(x,IterMax,K,L)


T = length(x);
% Smooth the Input:
windowSize = 35;
polyOrder = 2;
y = sgolayfilt(x, polyOrder, windowSize);


% figure()
% plot(y)
[Peaks_y , Locs_y] = findpeaks(y);


Good_Indices_Peaks =   find( ((Peaks_y)>mean(abs(y))) );
% hold on
% plot(Locs_y(Good_Indices_Peaks),Peaks_y(Good_Indices_Peaks),'r*');
```

```
% Loc_target_k = zeros(1,K);
Loc_target_k = Locs_y(Good_Indices_Peaks);
Signal_Selected_k = zeros(L,K);
cntr = 1;
Indexes_to_remove =[];


for i=2:length(Loc_target_k)
    if(Loc_target_k(1,i) - Loc_target_k(1,i-1)<L)
        Indexes_to_remove(cntr) = i;
        cntr = cntr+1;
    end
end
Loc_target_k(Indexes_to_remove) = [];
Indexes_to_remove =[];
% CHoose Randomly K of found occurences:
if (length(Loc_target_k)>K)
    Indexes_to_remove = randperm(length(Loc_target_k),length(Loc_target_k)-K);
    Loc_target_k(Indexes_to_remove) = [];
else

    if(Loc_target_k(1,end)+1.5*L < T )
        Loc_target_k(1,end+1) =  Loc_target_k(1,end)+1.5*L ;
    elseif( Loc_target_k(1,end)-1.5*L - Loc_target_k(1,end-1) > L   )
        Loc_target_k(1,end+1) = Loc_target_k(1,end)-1.5*L;
    elseif (Loc_target_k(1,1)>2*L)
        Loc_target_k(1,end+1) = Loc_target_k(1,1)-L;
    end
    Loc_target_k = sort(Loc_target_k);
    num_to_add = K - length(Loc_target_k);


    if(num_to_add>0)
        for i=1:num_to_add
            diff_vector =  diff(Loc_target_k) ;
            Possible_Candidate_to_Insert_Next_To =  find(diff_vector>L);
            Loc_target_k(1,end+1) =
Loc_target_k(Possible_Candidate_to_Insert_Next_To(1,end))-L;
            Loc_target_k = sort(Loc_target_k);
        end
    end
end


for k=1:K
    Signal_Selected_k(:,k) = x(Loc_target_k(1,k)-L/4+1:Loc_target_k(1,k)+3*L/4);
end
```

```
% IterMax=10;
Alpha_vec = randn(K,1);
Z = buffer(y ,L, L-1 ,'nodelay');
T = length(y);
y_hat= zeros(size(y));


for i=1:IterMax
    % Step-1: Shai is fixed
    S_hat_time = Signal_Selected_k*pinv(Alpha_vec');
    % Normalize S hat
    S_hat_time = S_hat_time/norm(S_hat_time);
    % Step-2:  S is fixed
    Chosen_Index_Z = zeros(1,K);
    temp_Z = Z;
    b = zeros(1,T);
    for k=1:K
       [~,index_max]  = max(   abs( S_hat_time'*temp_Z  ));
       Chosen_Index_Z(1,k) = index_max;
       if(index_max-L/4+1<0)
        Index_to_Change = 1:L;
       else
        Index_to_Change = index_max-L/4+1:index_max+3*L/4;
       end
       b(1,index_max) =  S_hat_time'*temp_Z(:,index_max)/( S_hat_time'*S_hat_time  );
       Alpha_vec(k,1) = b(1,index_max);
       Signal_Selected_k(:,k) = y(Index_to_Change)  ;%Chosen_Index_Z(1,k)-
L/4+1:Chosen_Index_Z(1,k)+3*L/4
       y_hat(1,Index_to_Change) = Signal_Selected_k(:,k);
       temp_Z(:, Index_to_Change ) = 0;
    end
end



end
```

همچنین فرض شده است که اگر از $K$ تا نقطه کمتر در پردازش اولیه پیدا شد، به تعداد کمبودها نقطه‌ی جدید پیدا شود. نقاط جدید پیدا شده دارای قید حداقل فاصله با عناصر مجاور خود می‌باشند.

همچنین قدم‌های برداشته شده برای رسیدن به خروجی مناسب به صورت زیر است:

```
IterMax = 2;
X = Data_hw7.X;
[Row_X , Col_x] = size(X);
K = 5;
L=100;
% Generate Random A Matrix:


A   = randn(Row_X);
A   = normalize(A, 'norm', 2) ;  % Normalize Columns of A


IterMax_2 = 50;
Rep_Error = zeros(1,IterMax_2);
best_Rep_Err = inf;
for i=1:IterMax_2


    S_hat =  A\X;


    y_hat_Xj = zeros(Row_X,T) ;
    b_Xj     = zeros(Row_X,T) ;
    S_hat_time_Xj = zeros(Row_X,L);







    for j=1:Row_X
            flip = 0;
            if ( abs(min(S_hat(j,:)))>abs(max(S_hat(j,:)))  )
                S_hat(j,:) = -S_hat(j,:);
                flip = 1;
            end
            [ y_hat_Xj(j,:) , b_Xj(j,:) , S_hat_time_Xj(j,:) ]  =
Single_Ch_Time_Dec(S_hat(j,:),IterMax,K,L);
            if(flip)
```

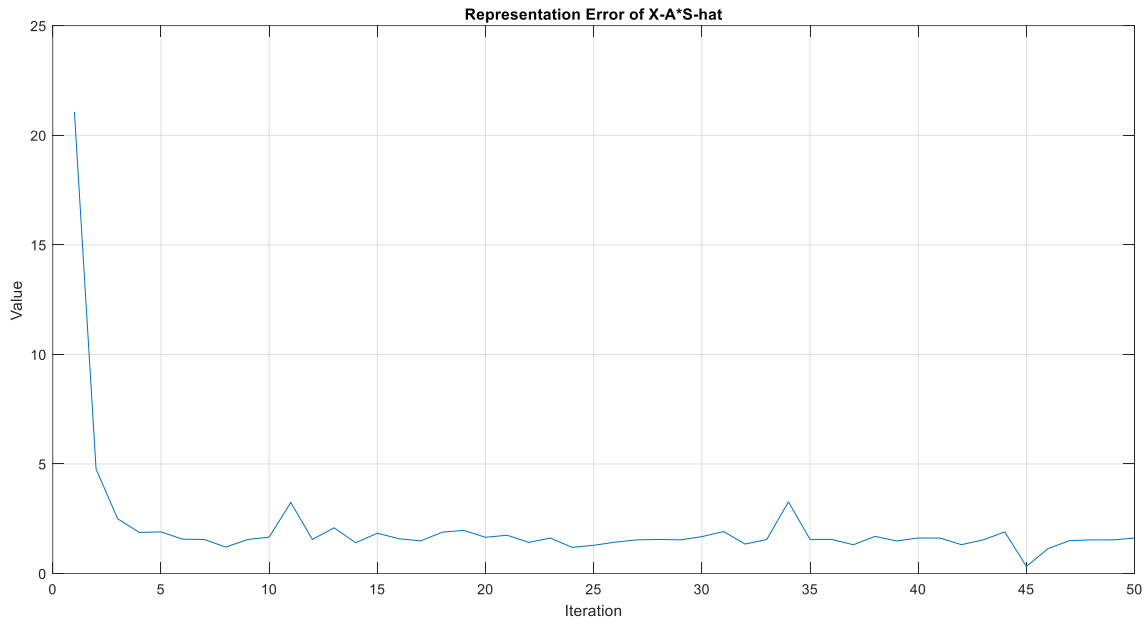
```
                y_hat_Xj(j,:)  = -y_hat_Xj(j,:) ;
                b_Xj(j,:)  = -b_Xj(j,:) ;
                S_hat_time_Xj(j,:)  = -S_hat_time_Xj(j,:) ;
            end
        % S_hat(j,:)  = conv( b_Xj(j,:) , S_hat_time_Xj,"same" )  ;
    end
    % S_hat = [conv(Si1,S1_hat,'same');conv(Si2,S2_hat,'same')];
    S_hat = y_hat_Xj;


    % Step-2: S is fixed --> FInd A:
    A = X*pinv(S_hat);
    A   = normalize(A, 'norm', 2) ;


    % Calculate the Representation Error:
    Rep_Error(1,i) = norm( X-A*S_hat ,"fro" );
    if (abs(Rep_Error(1,i) - mean(Rep_Error(1,1:i))) < 1e-3 ) && (i>5) % Convergence
Happens
        break;
    elseif( Rep_Error(1,i)< best_Rep_Err )
        best_Rep_Err = Rep_Error(1,i);
        Best_Index_Rep_Err = i;
        Best_S_hat = S_hat;
        Best_b_vec = b_Xj ;
        Best_S_hat_k = S_hat_time_Xj;
    end
end
```

در ابتدا باید ماتریس رندوم $A$ را تولید و سپس نرمالایز کنیم. در ادامه باید ماتریس $S$ را به دست آورده که هر سطر این ماتریس را برایش یکبار *Single Channel Deconvolution* انجام می‌دهیم.

پس از این قسمت، باید به سراغ *update* کردن $A$ برویم، برای اینکار، از رابطه‌ی $A = X * pinv(S)$ استفاده می‌کنیم.

- توجه شود قبل از استفاده از تابع *Single channel blind deconv.* ابتدا یک پیش پردازش صورت می‌گیرد تا سعی شود *spike* ها به سمت مثبت باشند و سپس برای این سیگنال پردازش انجام شود.

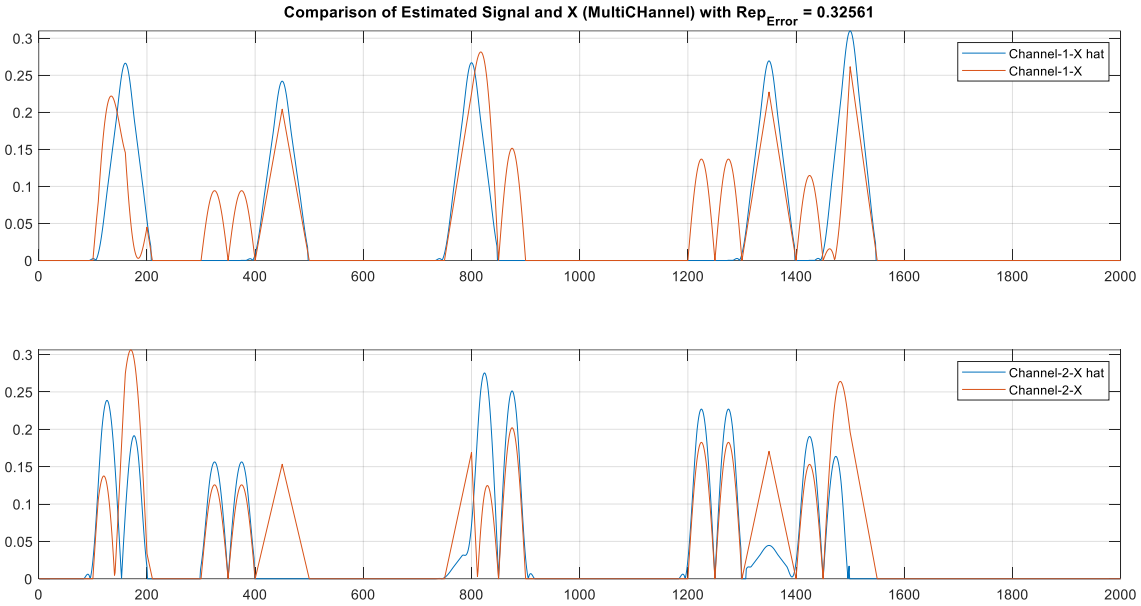
- در هر تکرار، بهترین‌ها تا آن تکرار را نگه داشته و ذخیره می‌کنیم. در نهایت پس از همگرایی، نمودار خطا متناسب با تعداد تکرار را رسم کرده ایم:




شکل 10

نمودار خطای نمایش بر حسب تعداد تکرار

تقریبا خطای نمایش به مقدار *0.3* رسیده است که بسیار مناسب است:




شکل 11

با کمی دقت متوجه می‌شویم که ماتریس $A$ تقریبا به درستی تخمین زده شده است ولی همچنان قسمت‌های کمی از کانال 1 در کانال 2 دیده می‌شوند و بالعکس.

## قسمت-4:

۴- قسمت ۱ را مجدداً در نظر بگیرید. فرض کنید داده ی $x_1$ را به حوزه ی فرکانس می بریم. با پردازشی در حوزه ی فرکانس، مساله ی قسمت ۱ را حل کنید.

برای این قسمت ابتدا باید فرمول‌بندی مسئله انجام شود:

$$in\ Time\ we\ had:$$

$$[\hat{s}, \hat{\text{Ш}}] = argmin \sum_t \left(x(t) - s(t) * \text{Ш}(t)\right)^2$$

$$in\ Frequency\ we\ have:$$

$$\hat{x}(f) = \hat{s}(f).\hat{\text{Ш}}(f)$$

$$Where\ \hat{\text{Ш}}(f) = \sum_{k=1\ to\ K}\left(\alpha_k e^{-j2\pi f \tau_k}\right)$$

برای ادامه ابتدا متناسب با طیف موجود، قسمتی که سیگنال اصلی در آن طیف فرکانسی نیست را از $s$ و $\text{Ш}$ در حوزه‌ی فرکانس حذف می‌کنیم.

از این قسمت به بعد دوباره رویکرد ما استفاده از *Alternation Minimization* است:

به ترتیب که ابتدا بردار $\hat{\text{Ш}}(f)$ را ثابت گرفته و سپس $\hat{s}(f)$ را به دست می‌آوریم. نکته‌ی مهم آن است که در حوزه‌ی فرکانس باید سیگنال $\hat{s}(f)$ را متقارن در نظر بگیریم چرا که باید یک سیگنال زمانی حقیقی در حوزه‌ی زمان به ما بدهد.

پس به این صورت به دست خواهد آمد:

$$\hat{s}(f) = \frac{1}{2}\left(\frac{\left(\hat{x}(f)\right)}{\widehat{\amalg}(f)} + \frac{\left(\hat{x}(-f)\right)^*}{\widehat{\amalg}(-f)^*}\right)$$

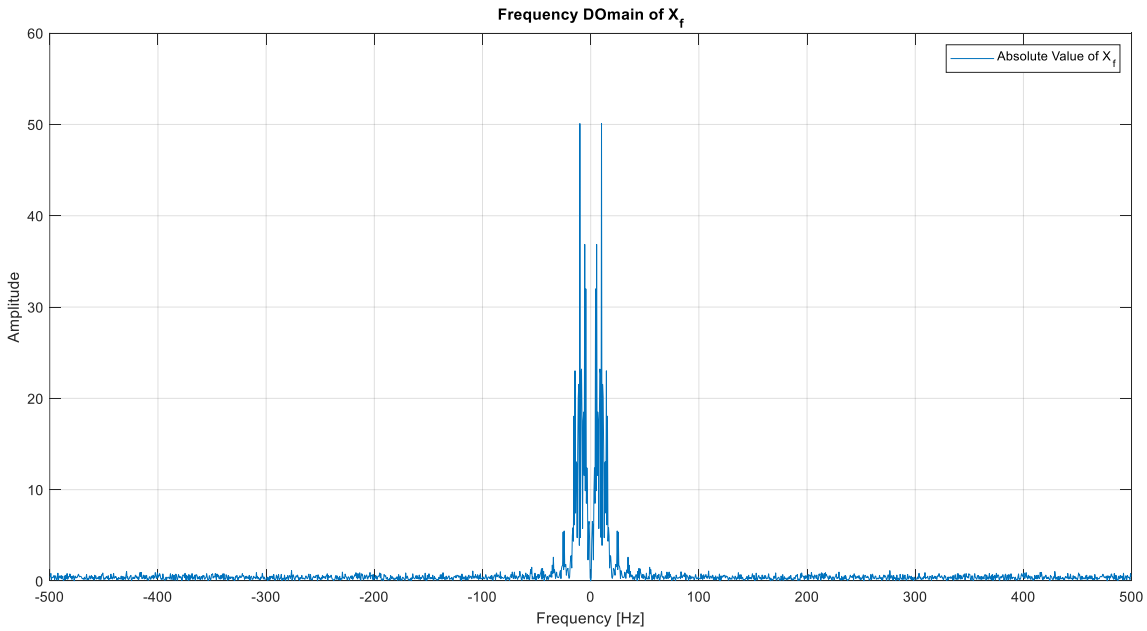
با این فرم، قدم اول الگوریتم به پایان می‌رسد. در قدم بعدی باید با دانسته فرض کردن $\hat{s}(f)$ به سراغ تعیین $\widehat{\amalg}(f)$ برویم.

برای قدم بعدی از تکنیک *ifft* با تعداد سمپل برابر $T$ استفاده می‌کنیم.در واقع داریم:

$$\hat{y}(f) = \frac{\hat{x}(f)}{\hat{s}(f)} = \sum_k \left(\alpha_k e^{-j2\pi f \tau_k}\right)$$

به این ترتیب با گرفتن یک *ifft* از $\hat{y}(f)$ می‌توان مکان و اندازه‌ی سیگنال را به دست آورد که به ترتیب همان $\tau_k$ها و $\alpha_k$ های ما هستند.

با توجه به مراحل بالا داریم:

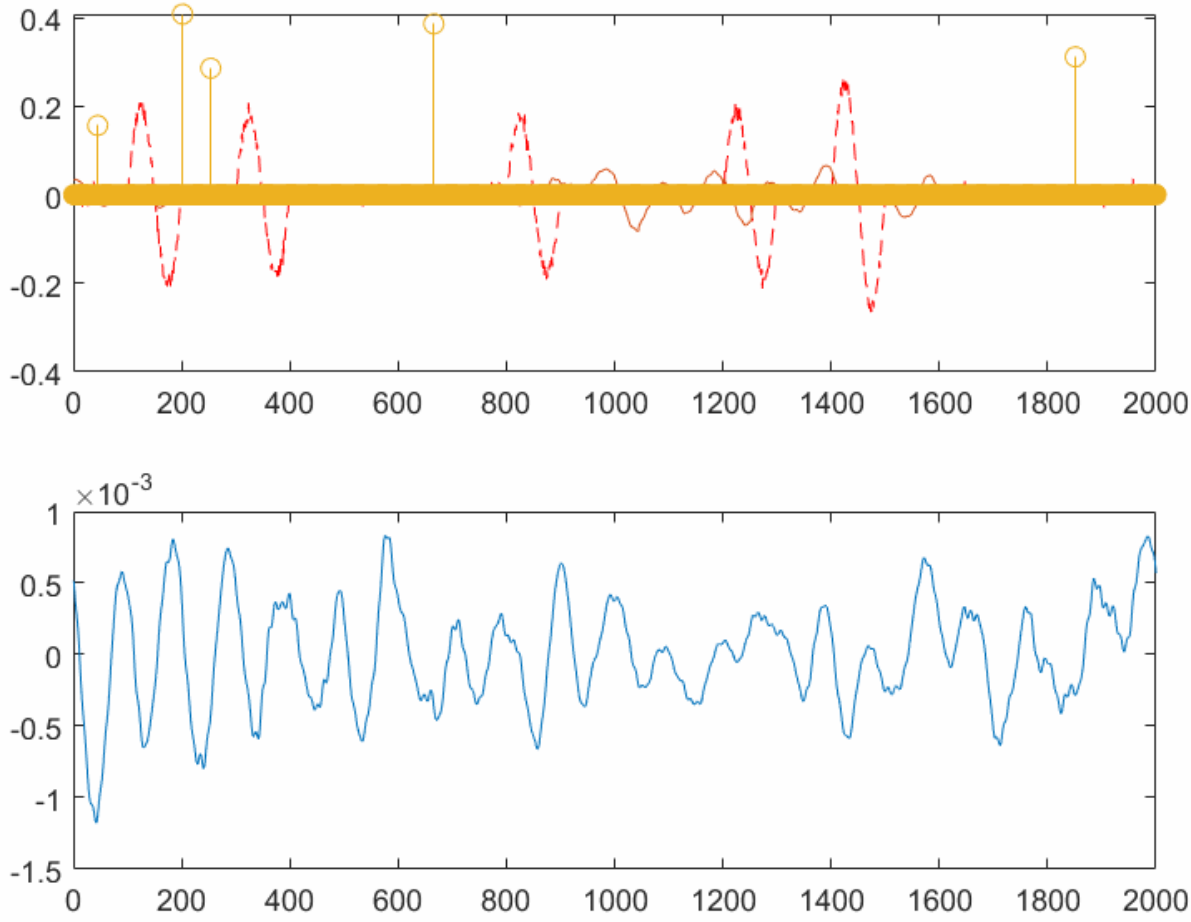



شکل 12

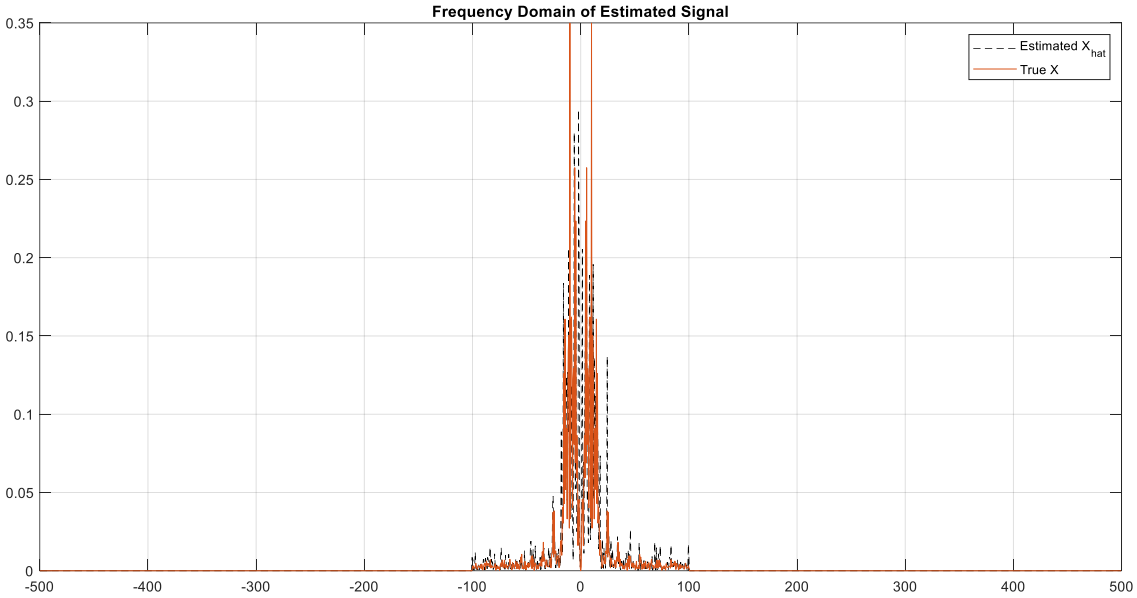
در حوزه‌ی فرکانس به نظر می‌رسد از 100 هرتز به بعد فرکانس نداریم. پس آنها را برابر0 هم در $\hat{s}$ و هم در $\widehat{\text{Ш}}(f)$ می‌گیریم.

```
Indices_TO_Remove = find(abs(freq)>100)  ;
Y_F = X_f;
Y_F(Indices_TO_Remove) = 0;
T = length(Y_F);
Y_F = fftshift(Y_F)/norm(abs(Y_F));
% Random Generation of SHAI:
SHAI_f_hat = randn(size(Y_F))+1j*randn(size(Y_F)) ;
close all;
figure()
Rep_Error = zeros(1,IterMax);
Mean_Error =inf;
IterMax=150;
for i=1:IterMax
    % Step-1: S_f Update based on fixed SHAI_hat

    S_f_hat    = 0.5*( (Y_F)./(SHAI_f_hat) + conj( flip(Y_F) )./conj( flip(SHAI_f_hat) )
);
    % Step-2: Obtain "SHAI hat"
    YY_F_hat =zeros(size(Y_F));
    NonZero_Indexes = find(S_f_hat ~= 0);
    YY_F_hat(NonZero_Indexes) = Y_F(NonZero_Indexes)./S_f_hat(NonZero_Indexes);
    SHAI_hat = ifft((YY_F_hat),T);

    % Choose with deflation:
    TAU_vec = zeros(1,K);
    Temp_SHAI_HAT = SHAI_hat;


    for tau_ind=1:K
        [~,cache_Tau] = sort(abs(Temp_SHAI_HAT));
        TAU_vec(1,tau_ind) =  cache_Tau(1,end);
        if(cache_Tau(1,end)-L/2<1)
            Temp_SHAI_HAT(1,1:L) = 0;
        elseif(cache_Tau(1,end)+L/2>length(Temp_SHAI_HAT))
            Temp_SHAI_HAT(1,end:end-L) = 0;
        else
            Temp_SHAI_HAT(1,cache_Tau(1,end)-L/2:cache_Tau(1,end)+L/2) = 0;
        end
    end
%     [~ , TAU_vec ]  =  sort(abs(SHAI_hat));
%     TAU_vec   =   sort(TAU_vec(1,1:K));
```

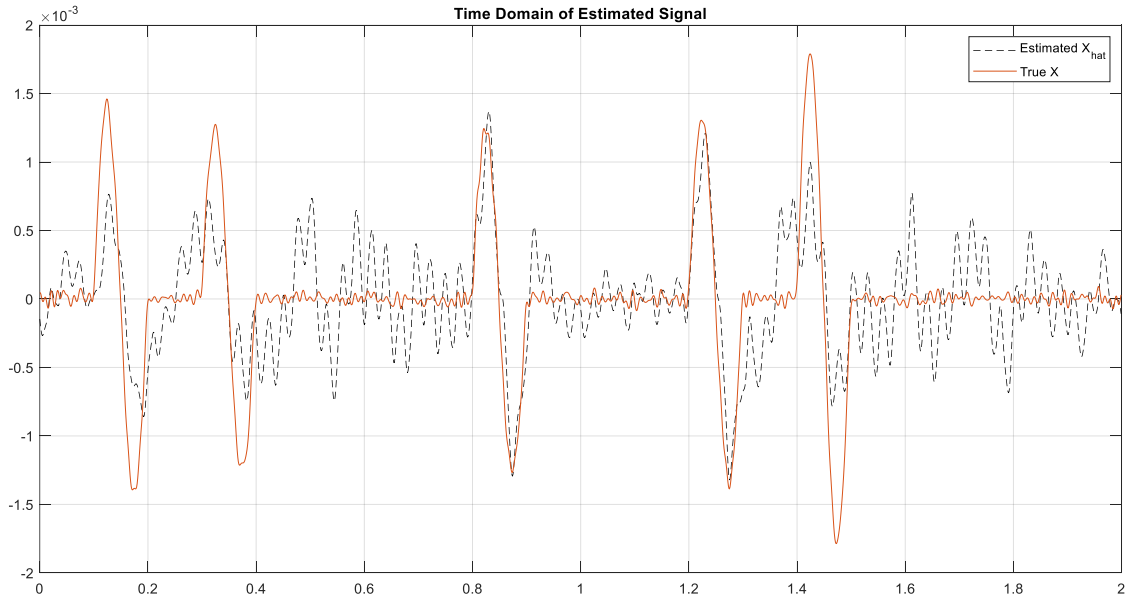
```
    Alpha_Vec =   SHAI_hat(TAU_vec);
    Alpha_Vec = Alpha_Vec/norm(Alpha_Vec);


    SHAI_time = zeros(1,T);
    SHAI_time(TAU_vec) = abs(real(Alpha_Vec));

    SHAI_f_hat  = (fft(SHAI_time));
    SHAI_f_hat  = SHAI_f_hat/norm(abs(SHAI_f_hat),"fro");  % Normalize
    S_hat_Time  = real(ifft((S_f_hat)));
    X_hat_F = SHAI_f_hat.*S_f_hat;
    X_hat_F = X_hat_F/norm(abs(X_hat_F) , "fro" ); % Normalize
    X_hat_Time = conv(S_hat_Time,SHAI_time , "same");
    X_hat_Time = X_hat_Time/norm(X_hat_Time,"fro"); % Normalize
    % Error Calculation:
    Rep_Error(1,i) = norm( ifft(Y_F)/max(ifft(Y_F)) -
ifft(SHAI_f_hat.*S_f_hat)/max(ifft(SHAI_f_hat.*S_f_hat))   , "fro") ;
    if(Mean_Error >Rep_Error(1,i) )
        Mean_Error = Rep_Error(1,i);
        Best_Indice = i;
        Best_S_hat_Time = S_hat_Time;
        Best_SHAI_f_hat = SHAI_f_hat;
        Best_SHAI_time = SHAI_time;
        Best_TAU_vec = TAU_vec;
        Best_Alpha_Vec = Alpha_Vec;
        Best_X_hat_F = X_hat_F;
    end
    %close all;
    drawnow
    subplot(2,1,1)
    plot(x1,'r--');
    hold on
    plot(X_hat_Time);
    stem(SHAI_time);
    hold off;
    subplot(2,1,2)
    plot(real(ifft(X_hat_F)));

end
```


0.4
0.2
0
-0.2
-0.4
0
200
400
600
800
1000
1200
1400
1600
1800
2000
×10^-3
1
0.5
0
-0.5
-1
-1.5
0
200
400
600
800
1000
1200
1400
1600
1800
2000

شکل 13

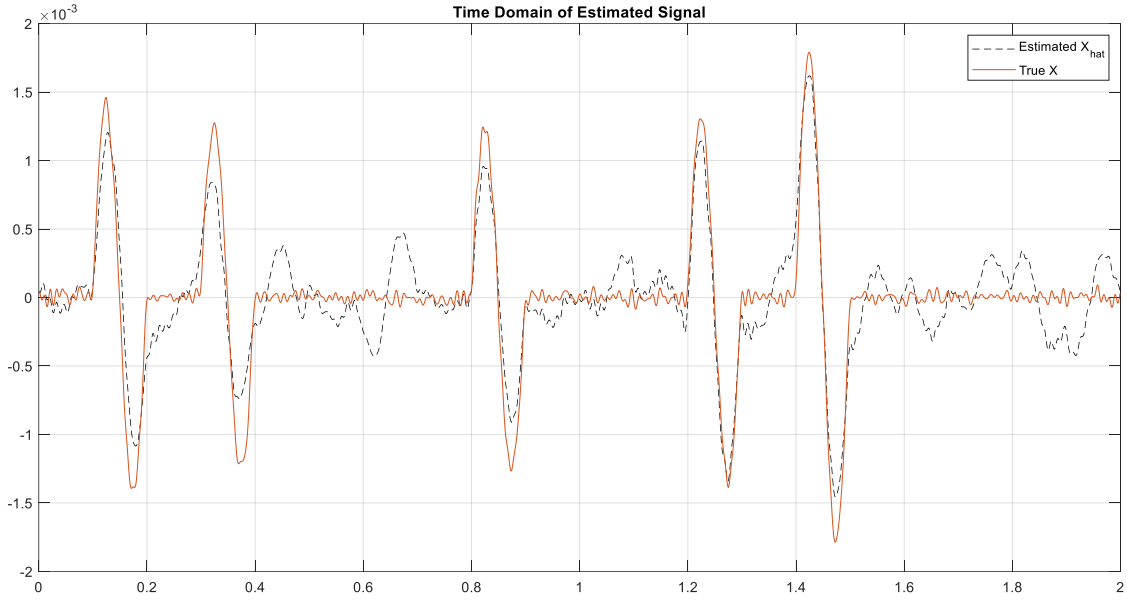

Frequency Domain of Estimated Signal
Estimated X_hat
True X
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-500
-400
-300
-200
-100
0
100
200
300
400
500

شکل 14


Time Domain of Estimated Signal
×10^-3
Estimated X_hat
True X
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

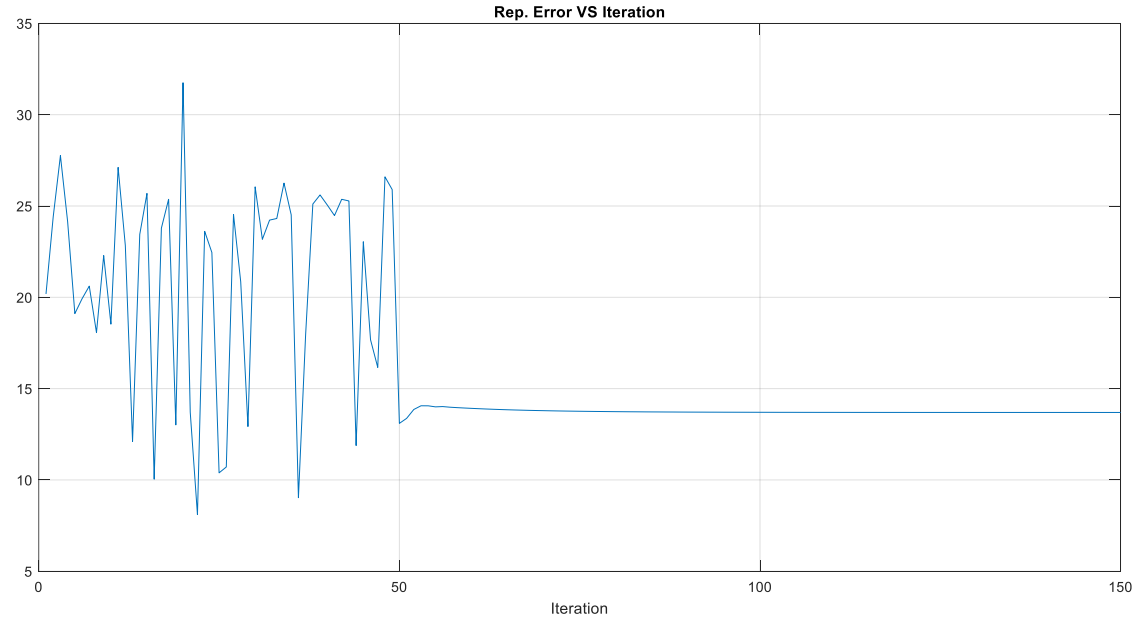
شکل 15

**بهترین نتیجه** برای تمامی تکرار ها به صورت زیر است:




شکل 16

***به خوبی همگرا شده است!***




شکل 17

همگرایی نیز در شکل نمودار خطا بر حسب تعداد iteration نیز مشهود است!

# پایان