ADAPTIVE BEAMFORMING

Assignment 3: EEL 6935 - SPRING.2018 Hector Lopez 3-2-2018

Problem 1

Use the Matrix Inversion Lemma (Woodbury's identity) to derive a recursion for the inverse of the estimated input autocorrelation matrix $\left[\hat{R}(k)\right]^{-1}$ based on the sample average recursion

$$(1) \quad \hat{R}(k) = \frac{(k-1) * \hat{R}(k-1) + r_k r_k^H}{k}$$

where $r_k, k = 1,2, \ldots$, are the array input vectors.

Let's recall the Woodbury Identity.
Given: $A_{MxM}, B_{MxM}, C_{MxL}, D_{LxL}$ assume that A,B,D, are invertible.

If $A = B^{-1} + CD^{-1}C^H$;
Then $A^{-1} = B - BC(D + C^H BC)^{-1} + C^H B$

By direct multiplication the Woodbury Lemma shows:

$(AA^{-1} = \cdots = I)$ and $(A^{-1}A = \cdots = I)$

Lets first re-write our equation (1) so that it is in the form of the Woodbury formula. We will have to move the scalar (1/k) and remember to multiply our inverted matrix by it at the end.

$$(2.1) \quad \hat{R}(k) = \frac{(k-1)}{k} * \hat{R}(k-1) + \frac{1}{k} * \vec{r}_k \vec{r}_k^H$$

$$(2.2) \quad k * \hat{R}(k) = (k-1) * \hat{R}(k-1) + \vec{r}_k \vec{r}_k^H$$

We can now define the matrices that we will use in the inversion formula.

$$(3) \quad A = \hat{R}(k), \; B^{-1} = (k-1) * \hat{R}(k-1), \; C = \vec{r}_k, \; D^{-1} = 1$$

These matrices can be used in the lemma to define our inverse A matrix.

$$(4) \quad A^{-1} = B - BC(D + C^H BC)^{-1} + C^H B$$

$$(5.1) \quad \hat{R}(k)^{-1} = inv\left[(k-1) * \hat{R}(k-1)\right] - inv[(k-1) * \hat{R}(k-1)]\vec{r}_k\left(1 + \vec{r}_k{}^H inv[(k-1) * \hat{R}(k-1)]\vec{r}_k\right)^{-1} + \vec{r}_k{}^H inv[(k-1) * \hat{R}(k-1)]$$

$$(5.2) \quad \hat{R}(k)^{-1} = \frac{1}{(k-1)} * \hat{R}(k-1)^{-1} - \frac{1}{(k-1)} * \hat{R}(k-1)^{-1} * \vec{r}_k \left(1 + \vec{r}_k{}^H * \frac{1}{(k-1)} * \hat{R}(k-1)^{-1} * \vec{r}_k\right)^{-1} + \vec{r}_k{}^H * \frac{1}{(k-1)} * \hat{R}(k-1)^{-1}$$

$$(5.3) \qquad \hat{R}(k)^{-1} = \frac{1}{(k-1)} * \left[\hat{R}(k-1)^{-1} - \frac{\hat{R}(k-1)^{-1} * \vec{r}_k \vec{r}_k{}^H * \hat{R}(k-1)^{-1}}{(k-1) + \vec{r}_k{}^H * \hat{R}(k-1)^{-1} * \vec{r}_k}\right]$$

We need to add the scalar we had before the Woodbury lemma was applied.

$$(6.1) \qquad k * \hat{R}(k)^{-1} = \frac{1}{(k-1)} * \left[\hat{R}(k-1)^{-1} - \frac{\hat{R}(k-1)^{-1} * \vec{r}_k \vec{r}_k{}^H * \hat{R}(k-1)^{-1}}{(k-1) + \vec{r}_k{}^H * \hat{R}(k-1)^{-1} * \vec{r}_k}\right]$$

$$(6.2) \qquad \hat{R}(k)^{-1} = \frac{1}{k * (k-1)} * \left[\hat{R}(k-1)^{-1} - \frac{\hat{R}(k-1)^{-1} * \vec{r}_k \vec{r}_k{}^H * \hat{R}(k-1)^{-1}}{(k-1) + \vec{r}_k{}^H * \hat{R}(k-1)^{-1} * \vec{r}_k}\right]$$

Finally, the equation (6.2) shows the final inversion as a recursive function in terms of the k , the input vector instance and the previous inversion value to get the new inversion value. This recursive formula needs to be initialized. We can initialize it with $\hat{R}(0)^{-1} = c * I$ Where I is the identity matrix and for some c << 1.

Problem 2

Show that,

$$(7) \quad \alpha_1 \triangleq \frac{\alpha_1'}{SINR_{opt} * (1 - \alpha_1') + 1}$$

where $SINR_{opt}$ is the maximum attainable SINR by the antenna array and α_1, α_1' are as defined in the lectures.

We are examining the signal present adaptive beamformer versus the signal absent adaptive beamformer by analyzing the SINR of each, versus the optimal SINR of a beamformer given some linear antenna array.

$$(1) \quad \alpha_1 = \frac{SINR_1}{SINR_{opt}} = \frac{\left(\vec{S}_\theta{}^H * \hat{R}^{-1} * \vec{S}_\theta\right)^2}{\left(\vec{S}_\theta{}^H * R_{I+N} * \vec{S}_\theta\right)\left(\vec{S}_\theta{}^H * \hat{R}^{-1} * R_{I+N} * \hat{R}^{-1} * \vec{S}_\theta\right)}$$

$$(2) \quad \alpha_2 = \frac{SINR_2}{SINR_{opt}} = \frac{\left(\vec{S}_\theta{}^H * \hat{R}_{I+N}{}^{-1} * \vec{S}_\theta\right)^2}{\left(\vec{S}_\theta{}^H * R_{I+N}{}^{-1} * \vec{S}_\theta\right)\left(\vec{S}_\theta{}^H * \hat{R}_{I+N}{}^{-1} * R_{I+N}{}^{-1} * \hat{R}_{I+N}{}^{-1} * \vec{S}_\theta\right)}$$

We notice that there is no direct way to solve for the PDF of α_1 because there are two types of matrices in the denominator, $\hat{R}^{-1}$, $\hat{R}_{I+N}{}^{-1}$ for the eq.1 . Let's define another version of α_1. by replacing the inversed sampled disturbance matrix, $\hat{R}_{I+N}{}^{-1}$ with $\hat{R}^{-1}$ in eq.1. This creates a new formula of the ratio α_1'. The resulting equation is similar to α_2. The two formulas are almost identical in form with the difference being the swap between the disturbance matrix and the input signal autocorrelation matrices. Since the two matrices, $\hat{R}^{-1}$, $\hat{R}_{I+N}{}^{-1}$, have the same distribution, the ratios of $\alpha_1{}'$ and α_2 also have the same distribution.

$$\alpha_1' \sim \alpha_2 \; (identically\ distributed)$$

$$(8) \quad E\{\alpha_1'\} = E\{\alpha_2\}$$

The denominator of the right hand side can be evaluated to always be greater than zero. We can prove this by first observing that the random variable α_1' is a ratio of the SINR's that ranges from zero to one.

$$0 \leq \alpha_1' \leq 1$$

The next observation is that the $SINR_{opt}$ is also a ratio of the signal and the disturbances, noise and interference, so it will be greater than 0.

$$SINR_{opt} > 0$$

We can prove by solving for the limits of the variables that the expression will remain as a positive value.

$$SINR_{opt} * (1 - \alpha_1') + 1 > 0$$

$$SINR_{opt} * (1 - \alpha_1') + 1 > SINR_{opt} + 1 \because \alpha_1' > 0$$

$$\cancel{0 * (1 - \alpha_1')} + 1 = 1 \because SINR_{opt} = 0$$

$$\cancel{SINR_{opt} * (0)} + 1 = 1 \because \alpha_1' = 1$$

The optimal SINR should be much larger than 0, so if the $SINR_{opt}$ value gets larger. We can also observe that as we reduce the value of α_1' the largest value α_1 can achieve is also reduced proportionally.

$$(9)\ \alpha_1 \triangleq \frac{\alpha_1'}{SINR_{opt} * (1 - \alpha_1') + 1} \ll \alpha_1'$$

In conclusion given the statement (eq.9) the random variable of α_1 will always be less than α_1' and similarly will always be less than α_2 since the mean of $E\{\alpha_1'\} = E\{\alpha_2\}$.

Problem 3

3. Design a simulation to estimate $E\{\alpha_1\}, E\{\alpha_2\}, E\{\alpha_3\}$:

Consider BPSK transmissions of one user of interest and three interferers.
Assume M = 10 and arbitrary in (-90,90) but fixed angles of arrival.

$$BPSK = \vec{r} = \sqrt{E_1} * b_1 * \vec{S}_{\theta_1} + \sum_{k=2}^{4} \sqrt{E_k} * b_k * \vec{S}_{\theta_k} + \vec{n}$$

$$where\ b = \pm 1$$

Data Generation :

Set reasonable SNR values for user and interference signals. Also set specific angles of arrival for each of these signals.

SNR	Theta
12dB	80
13dB	50
14dB	-50
15dB	-80

If we consider that the SNR of a signal is equivalent to the following expression:

$$SNR_i = \frac{P_{signal}}{P_{noise}} = \frac{{A_i}^2}{\sigma_{noise}^2}$$

We can derive the power of each of our BPSK signals given the SNR value we defined in our table. We can solve for the power of the signal and find that the E is equal to the SNR value converted from decibels to power times the variance of the noise. If we set the noise variance to 1 because it is a gaussian distributed zero mean noise. Then we can solve for the power of each signal as a function of SNR.

$$\sqrt{E_i} = \sqrt{10 * \log10(\mathrm{SNR}_i) * \sigma_{noise}^2}$$

$$E_i = 10 * \log10(\mathrm{SNR}_i)$$

SNR	E
12dB	10.7918
13dB	11.1394
14dB	11.4613
15dB	11.7609

We need to create binary data streams for each signal. Let's use the normal distribution random variable from 0 to 2 then split the values in Matlab to generate the binary +1 or -1 vectors. We would need to do something similar for the noise vector in order to get a white gaussian noise vector of M elements with zero mean and a variance of 1.

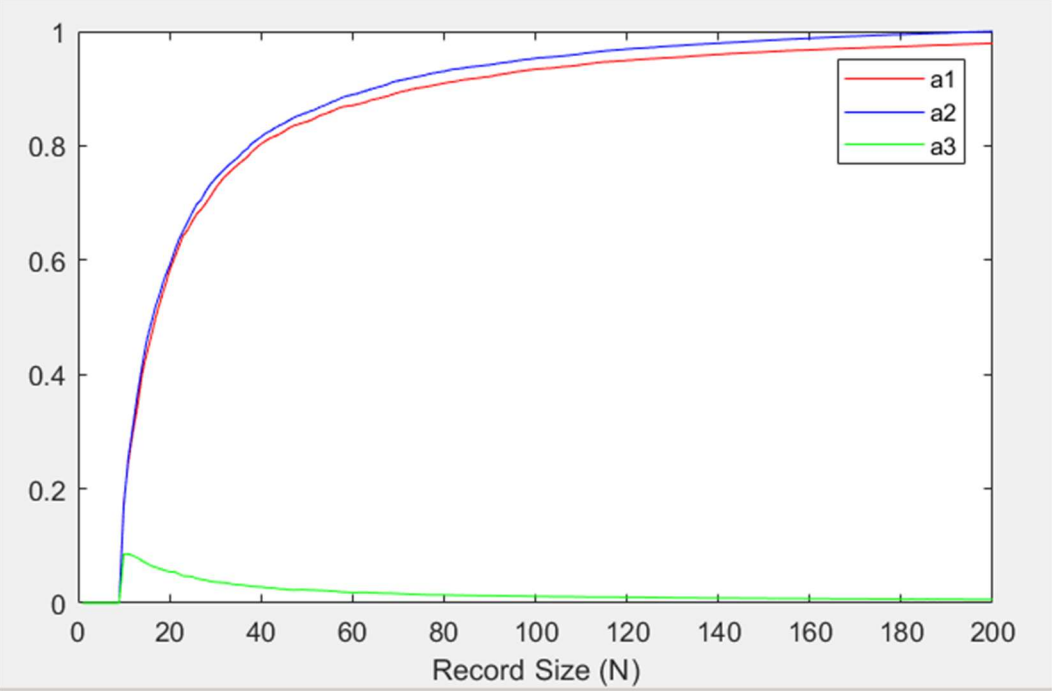
Lastly, we need to construct the array response vector for the given thetas. We will use Nyquist spacing for our antenna elements and assume an arbitrary carrier frequency. The S_theta is a vector of Mx1 of complex values.

All of the components for the received signal are derived at every iteration of N. The received signals are used as a large set to calculate the autocorrelation matrix and the disturbance matrix. The data generation will need to occur repeatedly for each sample of N then overall the samples create the variables needed to calculate the single alpha value for the set of N samples. We will re-run the entire experiment for different sized batches of N. Recording the alpha at each batch size, we will be able to trend how alpha changes based on increases in record size.

The formulas for the signal present and signal absent beam formers come from estimates of the auto-correlation of the received signals.

$$\widehat{\vec{w}}_1 = \hat{R}^{-1} * \vec{S}_\theta$$
$$\widehat{\vec{w}}_2 = \hat{R}_{I+N}{}^{-1} * \vec{S}_\theta$$

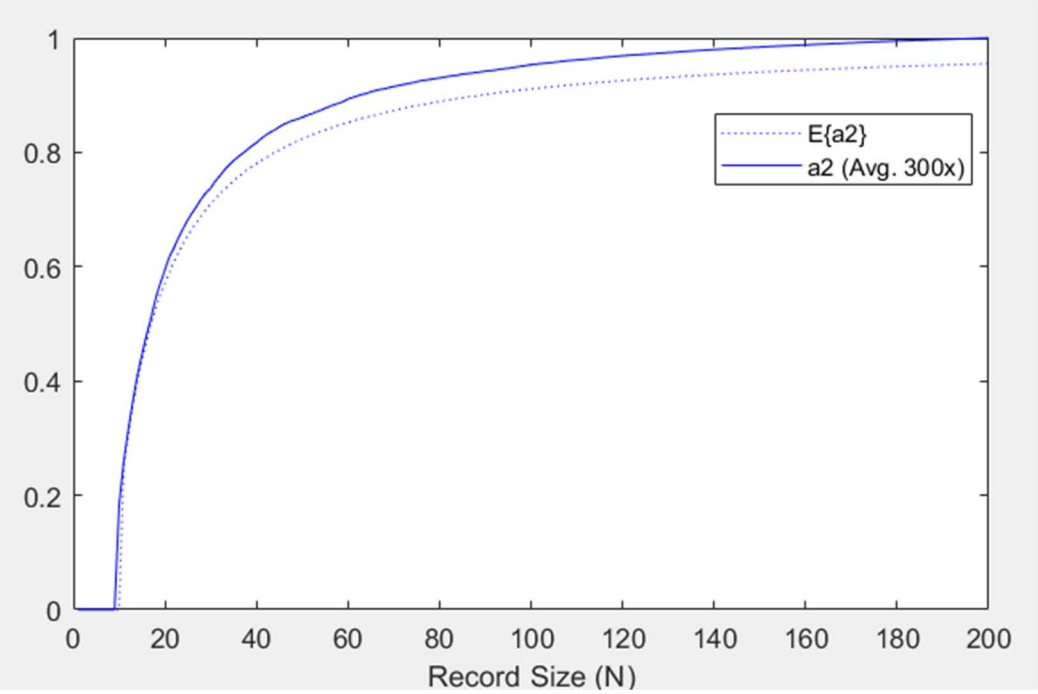
So the simulation will use the record size as the record size grows to create the $\hat{R}$ and $\hat{R}_{I+N}$ .The use of R_{I+N} and R are needed in the formulas for the alpha's so in order to create them the full sample record size of 200 will be used and all the generated received signals will be used to create the optimal auto-correlation matrix and the optimal disturbance matrix. By calculating the performance equation with the record size N a random variable for alpha would be given. Run the simulation sufficiently to get an estimate for alpha. Also this will be done at every record size between 10 and 200 . The following graph shows the result:



To verify the output of the program we can use the proof for the expected value of a2 derived from the incomplete beta distribution.

$$E\{\alpha_2\} = \frac{N+2-M}{N+1}, when\ (N > M-3)$$

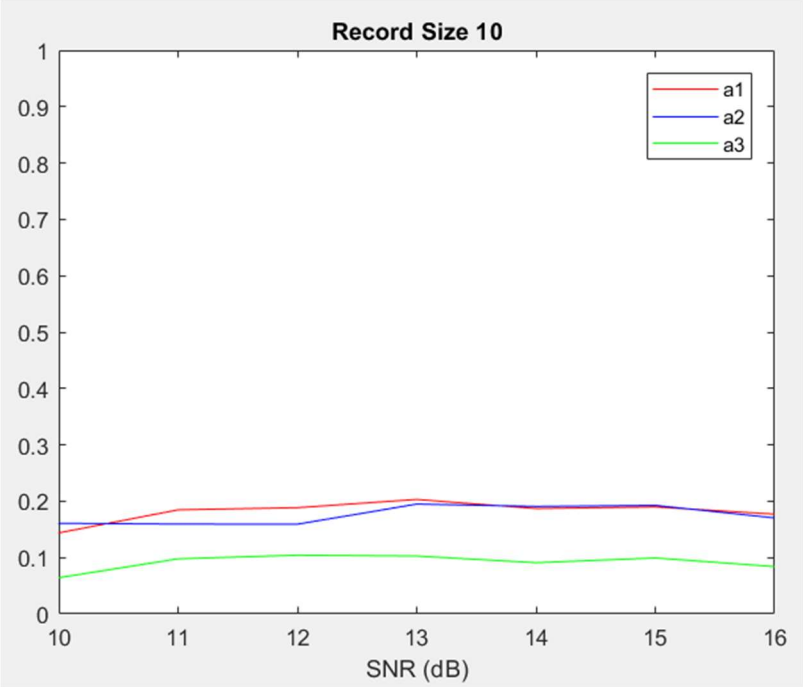
Mapping out the estimate against our simulation output we can see that the simulation is very close to our theoretical average.

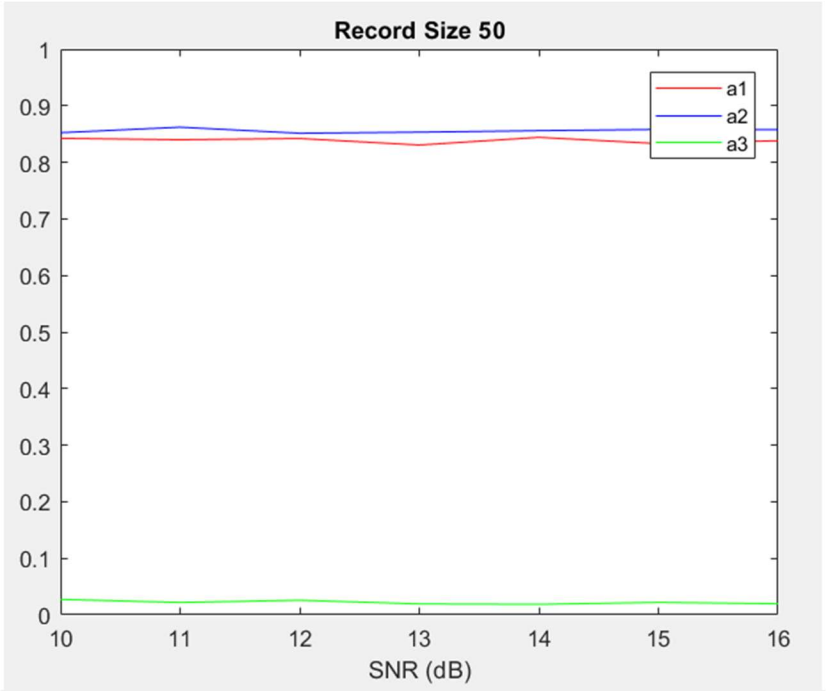


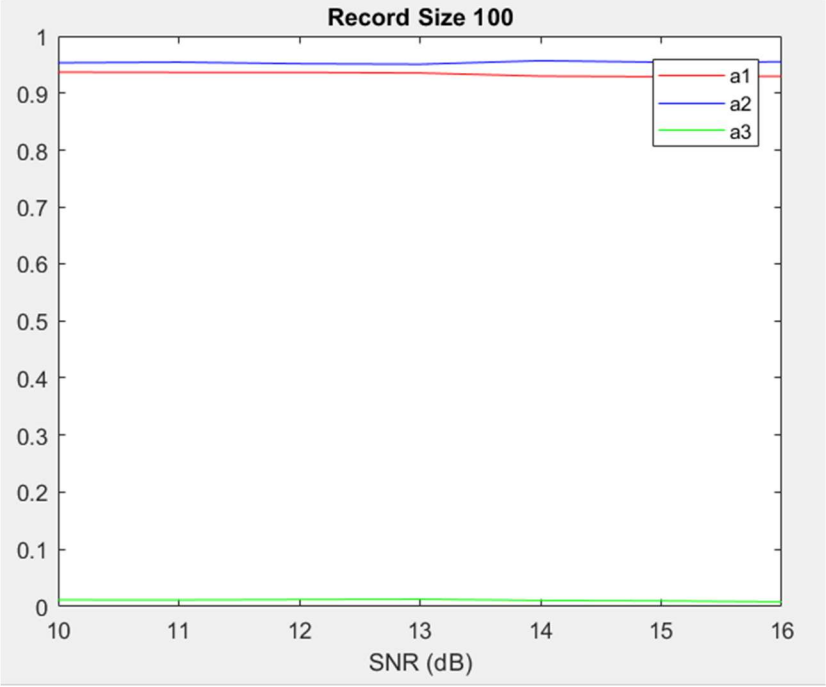
The exercise shows how the performance of alpha2 is greater than alpha1 this coincides with the mathematical proof described in part 2 of this assignment. The performance of alpha3 seemed to do poorly. The calculations for the alpha3 performance used the desired signal for each received signal. The implementation could of caused the performance calculated to be effected. The theorized performance of alpha3 would be that it performs better than alpha1 because it is capable of adjusting for error between the input and desired signals.

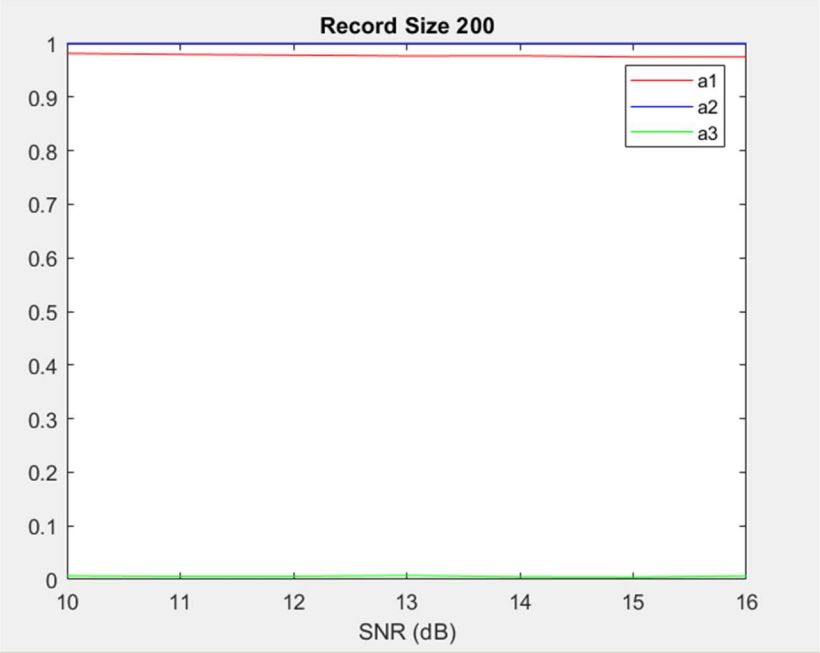
(a) Keep the SNRs of the interferences as in Part (a) and plot the estimated means of α1, α2, and α3 as a function of the SNR of the user of interest for N = 10,50,100,200.

So using the same setup as before we will change the SNR for the user to not be static but sweep between 10dB and 16dB. We will set the N samples to be static at 10, then re-run the simulation for N=50,100,200. This will generate 4 distinct graphs showing the performance across SNR of the user signal.


Record Size 10
a1
a2
a3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
11
12
13
14
15
16
SNR (dB)


Record Size 50
a1
a2
a3
SNR (dB)


Record Size 100
a1
a2
a3
SNR (dB)



Summary :

The performance of A2 is greater than the performance of A1 across a sweep of SNR of the user signals. We also see that with a greater record size the SNR ratio gets closer and closer to 1 . In our analysis A3 performed very poorly compared to the signal present and signal absent beam formers. A3 poor performance may be due to error in the programming . A3 is supposed to provide results equivalent to A2.

Appendix

Part 3.a Matlab Source Code

```
%% Signal Present Beamformer
%% Data Generation
M = 10;
SNR = [12,13,14,15];
theta = [80,50,-50,-80];
Nmax = 200;
Nmin = 10;
EstimationSize = 200;
alpha1 = zeros(1,Nmax);
alpha2 = zeros(1,Nmax);
alpha3 = zeros(1,Nmax);
r = zeros(M,Nmax);
des = zeros(M,Nmax);
ipn = zeros(M,Nmax);
```

```
%repeated for k times to estimate alphas
for k=1:EstimationSize

%% Data Generation for record size N
for t=1:Nmax
E=10*log10(SNR);
D=0:1:M-1;
S_th = exp(-1i*2*pi*(D'*.5)*sind(theta));
b= rand(M,4); b(b>0.5)=1; b(b<=0.5)=-1;
n = randn(M,1);
disturbance= E(2)*b(:,2).*S_th(:,2)+ ...
            E(3)*b(:,3).*S_th(:,3)+ ...
            E(4)*b(:,4).*S_th(:,4)+ n;
ipn(:,t) =disturbance;
desired = E(1)*b(:,1).*S_th(:,1);
des(:,t)= desired;
r(:,t) = desired+disturbance;
end
rsum = 0;ipnsum = 0;dsum=0;
for i = 1:Nmax
rsum = rsum + r(:,i)*ctranspose(r(:,i));
ipnsum = ipnsum + ipn(:,i)*ctranspose(ipn(:,i));
rr = r(:,i);
dd = conj(des(:,i));
dsum=dsum+ rr.*dd;
end
R= 1/Nmax * rsum;
R_ipn = 1/Nmax * ipnsum;


for N=Nmin:Nmax
        rsum = 0;ipnsum = 0;dsum=0;
        for i = 1:N
        rsum = rsum + r(:,i)*ctranspose(r(:,i));
        ipnsum = ipnsum + ipn(:,i)*ctranspose(ipn(:,i));
        rr = r(:,i);
        dd = conj(des(:,i));
        dsum=dsum+ rr.*dd;
        end
        Rh_ipn = 1/N * ipnsum;
        Rh= 1/N * rsum;
        w1h = inv(Rh)*S_th(:,1);
        w2h = inv(Rh_ipn)*S_th(:,1);
        w3h = inv(Rh) * ((1/N) * dsum);
        S_th_H = ctranspose(S_th(:,1));
        alpha1(k,N) = real((S_th_H*w1h)^2 / ...
            ((S_th_H*inv(R_ipn)*S_th(:,1))*(S_th_H*inv(Rh)*R_ipn*w1h)));
        alpha2(k,N) = real((S_th_H*w2h)^2 / ...
            ((S_th_H*inv(R_ipn)*S_th(:,1))*(S_th_H*inv(Rh_ipn)*R_ipn*w2h)));
        alpha3(k,N) = real(ctranspose(w3h)*S_th(:,1)*S_th_H*w3h / ...
          ((ctranspose(w3h)*R_ipn*w3h)*(S_th_H*inv(R_ipn)*S_th(:,1))));
end

end
```

```
for n=1:Nmax
   estAlpha1(n) = (1/EstimationSize) * sum(alpha1(:,n));
   estAlpha2(n) =  (1/EstimationSize) * sum(alpha2(:,n));
   estAlpha3(n) = (1/EstimationSize) * sum(alpha3(:,n));
   if(n>M)
     estCalcAlpha2(n) = (n + 2 -M) / (n + 1);
   else
       estCalcAlpha2(n) =0;
   end
end

%% Analysis
plot(estAlpha1,'color','r'); hold on;
plot(estAlpha2,'color','b');hold on;
plot(estAlpha3,'color','g');
%plot(estCalcAlpha2,':b');
```

Part 3.b Matlab Source Code

```
%% Signal Present Beamformer
%% Data Generation
M = 10;

theta = [80,50,-50,-80];
Nmax = 200;
Nmin = 10;
EstimationSize = 50;
alpha1 = zeros(1,Nmax);
alpha2 = zeros(1,Nmax);
alpha3 = zeros(1,Nmax);
r = zeros(M,Nmax);
des = zeros(M,Nmax);
ipn = zeros(M,Nmax);

Ns=[10,50,100,200];

%% Data Generation for record size N
for N = [10,50,100,200]
for sweep = 10:16
SNR = [sweep,13,14,15];
%repeated for k times to estimate alphas
for k=1:EstimationSize
%% Data Generation for largest Record Size
    for t=1:Nmax
    E=10*log10(SNR);
    D=0:1:M-1;
    S_th = exp(-1i*2*pi*(D'*.5)*sind(theta));
    b= rand(M,4); b(b>0.5)=1; b(b<=0.5)=-1;
    n = randn(M,1);
    disturbance= E(2)*b(:,2).*S_th(:,2)+ ...
                 E(3)*b(:,3).*S_th(:,3)+ ...
                 E(4)*b(:,4).*S_th(:,4)+ n;
```

```
    ipn(:,t) =disturbance;
    desired = E(1)*b(:,1).*S_th(:,1);
    des(:,t)= desired;
    r(:,t) = desired+disturbance;
    end
    rsum = 0;ipnsum = 0;dsum=0;
    for i = 1:Nmax
    rsum = rsum + r(:,i)*ctranspose(r(:,i));
    ipnsum = ipnsum + ipn(:,i)*ctranspose(ipn(:,i));
    rr = r(:,i);
    dd = conj(des(:,i));
    dsum=dsum+ rr.*dd;
    end
    R= 1/Nmax * rsum;
    R_ipn = 1/Nmax * ipnsum;

    rsum = 0;ipnsum = 0;dsum=0;
    for i = 1:N
    rsum = rsum + r(:,i)*ctranspose(r(:,i));
    ipnsum = ipnsum + ipn(:,i)*ctranspose(ipn(:,i));
    rr = r(:,i);
    dd = conj(des(:,i));
    dsum=dsum+ rr.*dd;
    end
    Rh_ipn = 1/N * ipnsum;
    Rh= 1/N * rsum;
    w1h = inv(Rh)*S_th(:,1);
    w2h = inv(Rh_ipn)*S_th(:,1);
    w3h = inv(Rh) * ((1/N) * dsum);
    S_th_H = ctranspose(S_th(:,1));
    alpha1(k) = real((S_th_H*w1h)^2 / ...
        ((S_th_H*inv(R_ipn)*S_th(:,1))*(S_th_H*inv(Rh)*R_ipn*w1h)));
    alpha2(k) = real((S_th_H*w2h)^2 / ...
        ((S_th_H*inv(R_ipn)*S_th(:,1))*(S_th_H*inv(Rh_ipn)*R_ipn*w2h)));
    alpha3(k) = real(ctranspose(w3h)*S_th(:,1)*S_th_H*w3h / ...
      ((ctranspose(w3h)*R_ipn*w3h)*(S_th_H*inv(R_ipn)*S_th(:,1))));
end
   estAlpha1(sweep) = (1/EstimationSize) * sum(alpha1);
   estAlpha2(sweep) =  (1/EstimationSize) * sum(alpha2);
   estAlpha3(sweep) = (1/EstimationSize) * sum(alpha3);
end
    figure

    %% Analysis
 plot(estAlpha1,'color','r'); hold on;
    plot(estAlpha2,'color','b');hold on;
    plot(estAlpha3,'color','g');
    legend('a1','a2','a3');
     xlabel('SNR (dB)');
    title( sprintf('Record Size %d',N));
    axis([10 16 0 1]);
end
```