ADSP Final Project

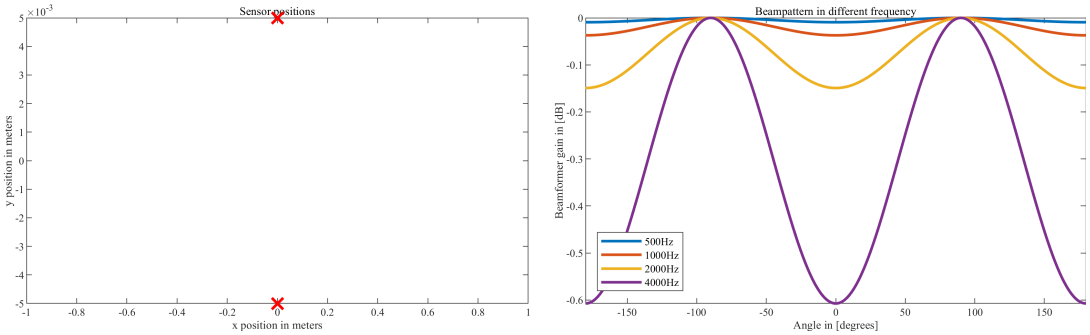
Gengjianhua and Song Ruibo

December 17, 2019

Assignment B: scenario 1: Analyse hearing aid array configuration

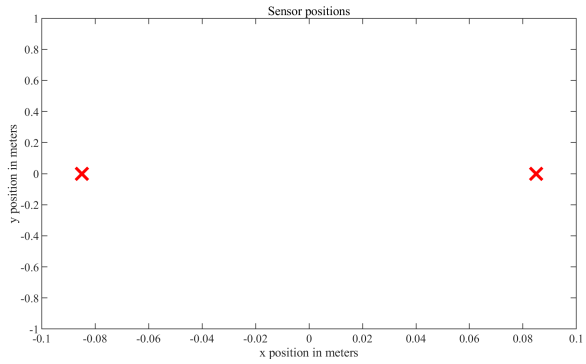
a) Fig.1 $\sim$ Fig.6 show very intuitive description of beampatterns with all $\frac{1}{J}$ weights in different frequencies. Before null-steering the beampattern is greatly affected by frequency because that the changes in frequency leads to a change in ratio $\frac{d}{\lambda}$. We can simply estimate $\frac{d}{\lambda} = \frac{c}{\lambda} \times \frac{d}{c} = f \times \frac{d}{c}$. When we use left two microphones $d = 0.01m$, the ratio $\frac{d}{\lambda} \approx 0.015$ to $0.12 \ll \frac{1}{2}$ with the frequency from 500Hz to 4kHz. Which means that no exact cancelling at $\theta = 0°$ and $180°$. The polar plots Fig.4 support our opinions. For the reason that the ration $\frac{d}{\lambda} \ll \frac{1}{2}$ and the sensor number J is two, their polar plots are almost round. In the case that only the upper two microphones are used, $d = 0.17m$ and the ratio $\frac{d}{\lambda} \approx \frac{1}{4}$ to 2. We get a perfect $\frac{d}{\lambda} = \frac{1}{2}$ when frequency $f = 1000$Hz, so Fig.5(a) shows the oversampling beampattern while Fig.5(c) and Fig.5(d) give the undersampling beampatterns with patterns repeat at $\theta = \arcsin \frac{\lambda}{d}$. Apparently, Fig.5(b) shows a perfect pattern, but it doesn't have a good resolution with only two sensors. Fig.6 and Fig.5 are approximately the same due to $dy = 0.01m \ll dx = 0.17m$ but we could see some differences in high frequencies such as Fig.5(d) and Fig.6(d) which have different gains in $0°$ and other spatial aliasing repeated pattern due to the role of tiny dy.

b) Fig.7 $\sim$ Fig.15 show the null-steering beampatterns in different frequencies with $0°$ desired source and $135°$ or $45°$ interferences under different array configurations. Obviously, Fig.7 $\sim$ Fig.15 meet the basics requirements that set unit gain to desired source at $0°$ while suppress interference to zero gain. Unfortunately, for null-steering the gain of other degrees can also bigger than one, which means bad SNR. We can clearly see that Fig.8 and Fig.9 are entirely different but Fig.11 and Fig.12 are the same. Which means that when we using left two microphones, the null-steering beampatterns are different at $135°$ and $45°$ interference sources, while the patterns are the same with different interference when utilize upper two sensors. The reason is that the left two sensor configuration leads to a cosine type beampattern $\cos\theta = \cos(-\theta)$ so they are different with $135°$ and $45°$ interference but they are even symmetric about $0°$. What's more, the upper two sensor configuration results in a sine type beampattern $\sin\theta = \sin(180° - \theta)$ so they are exactly the same with $135°$ and $45°$ interference. The subfigures under different frequencies in Fig.8 are similar but not the same. It may not be intuitive to distinguish them, but they're not exactly the same. Because $dy = 0.01m$ is very small which leads to $\frac{dy}{\lambda} \ll \frac{1}{2}$ and the change in frequency doesn't change $\frac{dy}{\lambda}$ much. Which means that giving a very large increase in frequency to break $\frac{dy}{\lambda}$ such as 40kHz we can get a quite different null-steering beampattern. Of course, a large increase in microphone spacing such as $dy = 0.1m$ can also get different subfigures due to breaking the ratio $\frac{dy}{\lambda}$. The quite different subfigures in Fig.11 validate our points with $\frac{dx}{\lambda} \approx \frac{1}{4} \sim 2$. Fig.14 and Fig.12 are approximately similar due to $dy = 0.01m \ll dx = 0.17m$ and we could see some differences in high frequencies such as Fig.14(d) and Fig.15(d) which have different gains in some degrees due to the role of tiny dy and the complex beampattern function types.

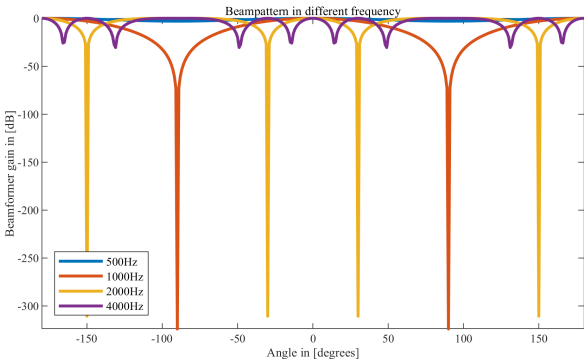


(a) Array configuration (b) Narrow-band beampattern

Figure 1: Beampattern of left two sensors before null-steering

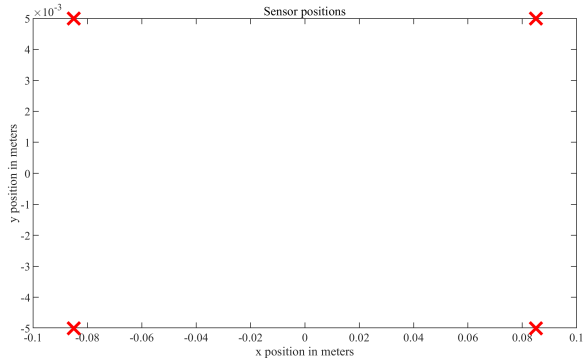


(a) Array configuration

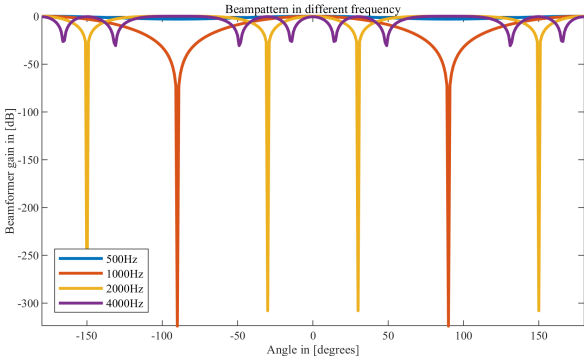


(b) Narrow-band beampattern

Figure 2: Beampattern of upper two sensors before Null-steering

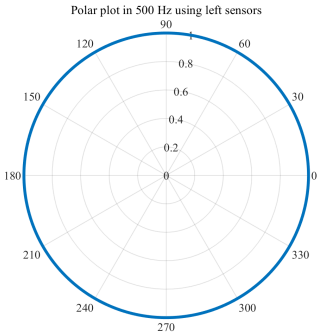


(a) Array configuration

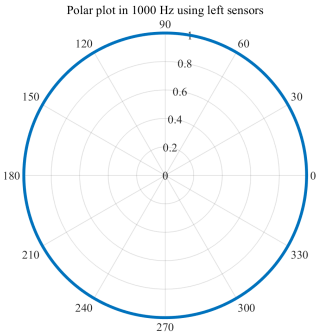


(b) Narrow-band beampattern

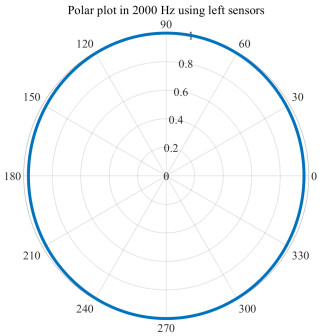
Figure 3: Beampattern of all four sensors before Null-steering



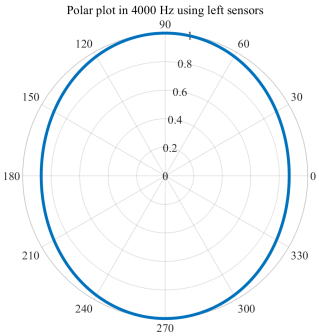
(a) Polar plot in 500Hz



(b) Polar plot in 1000Hz

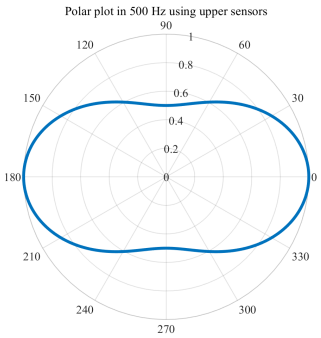


(c) Polar plot in 2000Hz

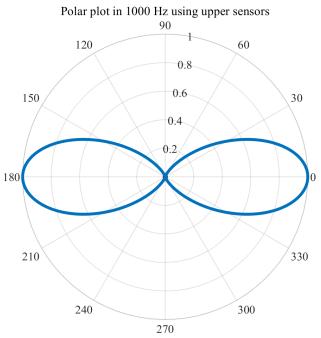


(d) Polar plot in 4000Hz

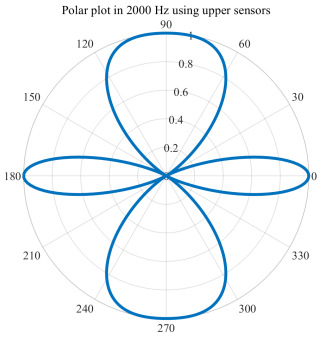
Figure 4: Polar plots in different frequencies using left two sensors before null-steering



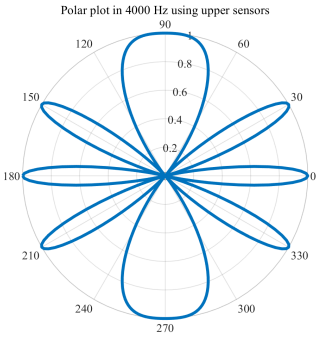
(a) Polar plot in 500Hz



(b) Polar plot in 1000Hz

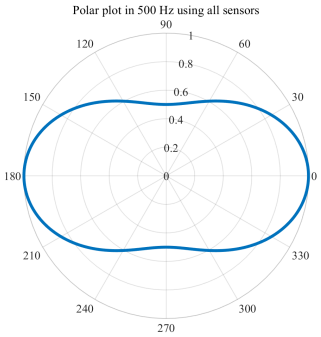


(c) Polar plot in 2000Hz

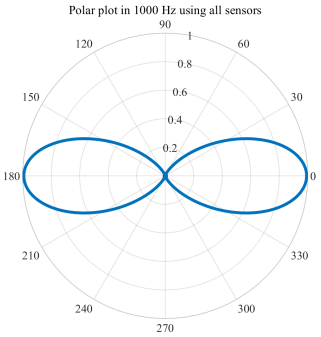


(d) Polar plot in 4000Hz

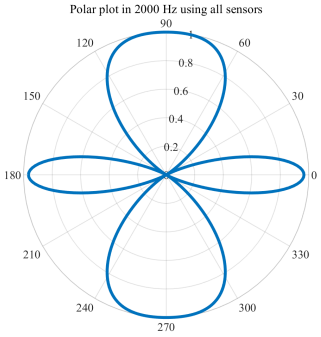
Figure 5: Polar plots in different frequencies using upper two sensors before null-steering



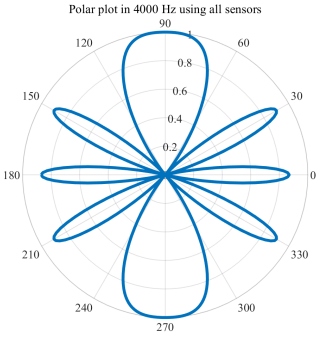
(a) Polar plot in 500Hz



(b) Polar plot in 1000Hz

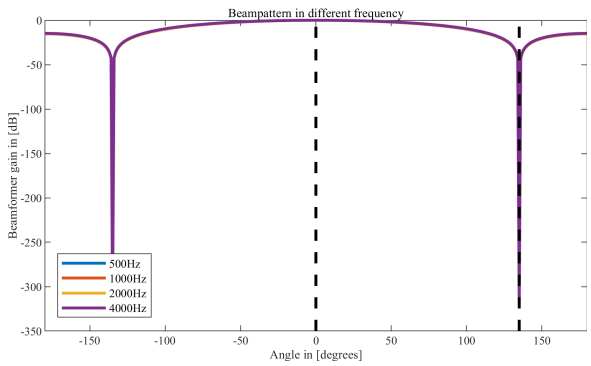


(c) Polar plot in 2000Hz

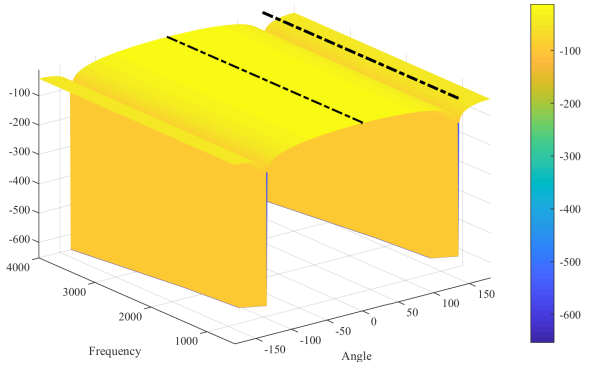


(d) Polar plot in 4000Hz

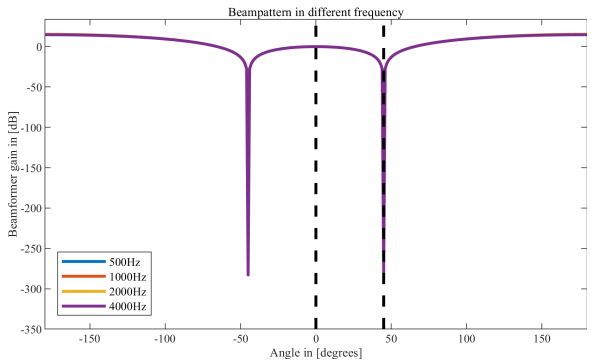
Figure 6: Polar plots in different frequencies using all four sensors before null-steering



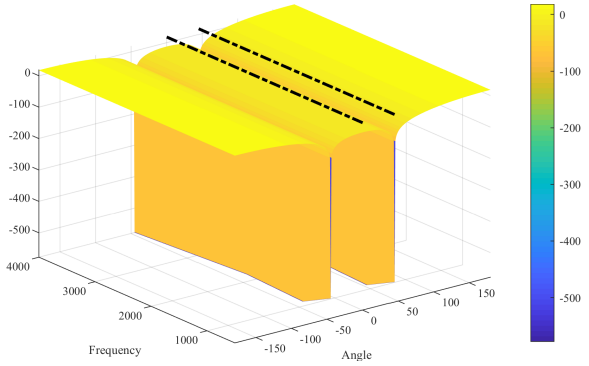
(a) Narrow-band beampattern with 135° interference



(b) Multi-tone beampattern with 135° interference

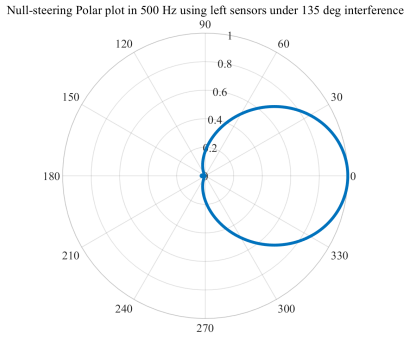


(c) Narrow-band beampattern with 45° interference

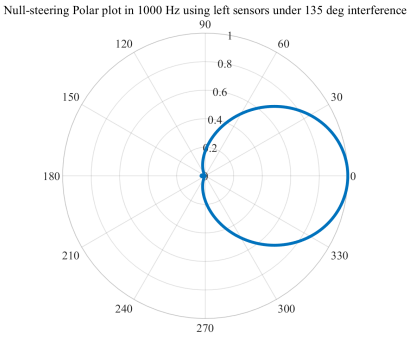


(d) Multi-tone beampattern with 45° interference

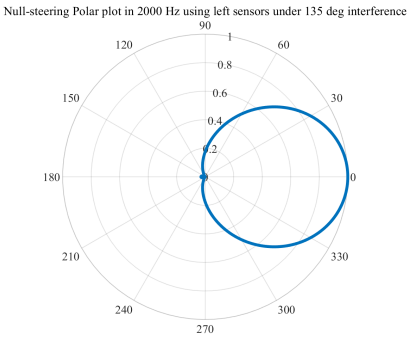
Figure 7: Null-steering beampattern using left two sensors under different interferences and the same 0° desired source



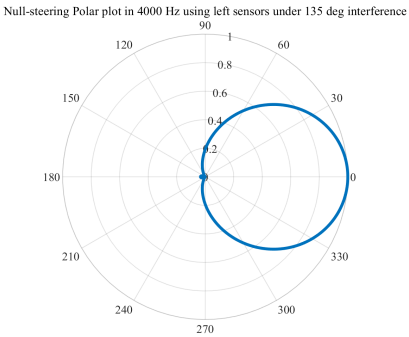
(a) Null-steering polar plot in 500Hz



(b) Null-steering polar plot in 1000Hz

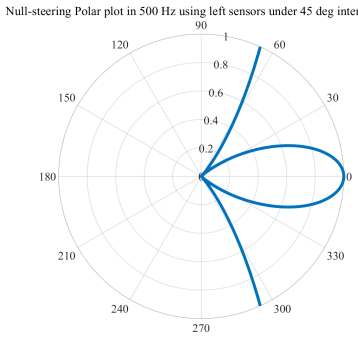


(c) Null-steering polar plot in 2000Hz

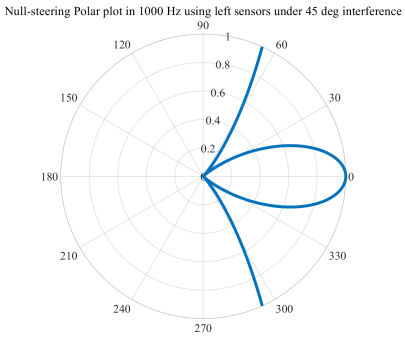


(d) Null-steering polar plot in 4000Hz

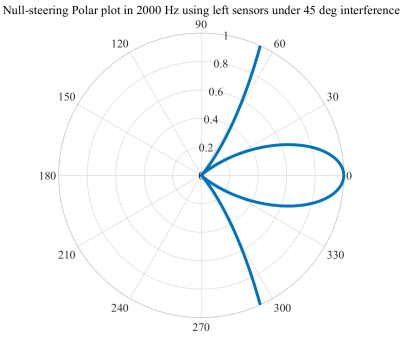
Figure 8: Null-steering polar plots in different frequencies using left two sensors under 135° interference and 0° desired source



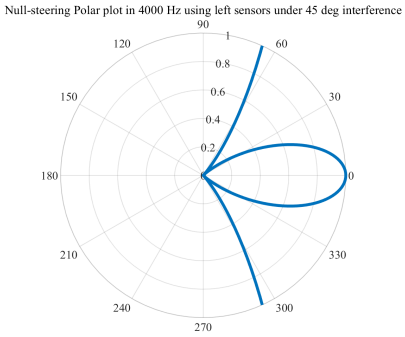
(a) Null-steering polar plot in 500Hz



(b) Null-steering polar plot in 1000Hz

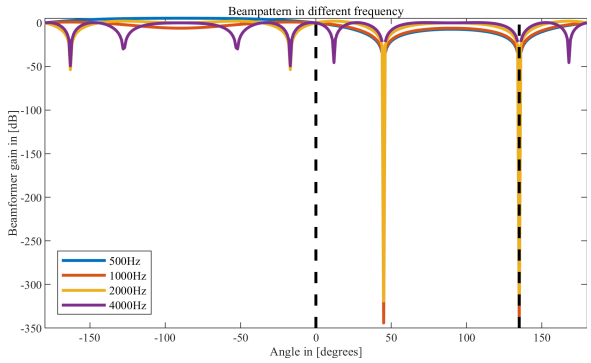


(c) Null-steering polar plot in 2000Hz

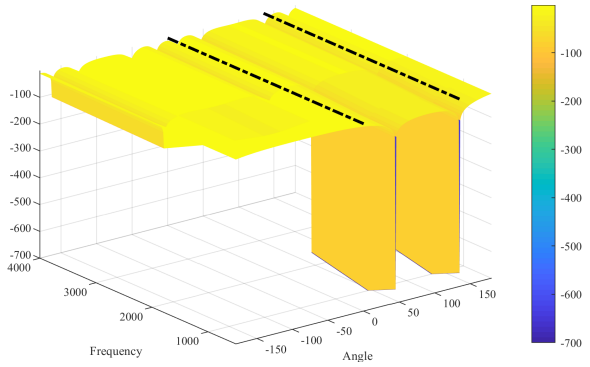


(d) Null-steering polar plot in 4000Hz

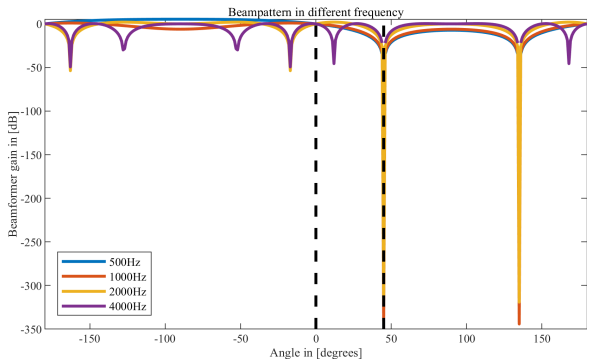
Figure 9: Null-steering polar plots in different frequencies using left two sensors under 45° interference and 0° desired source



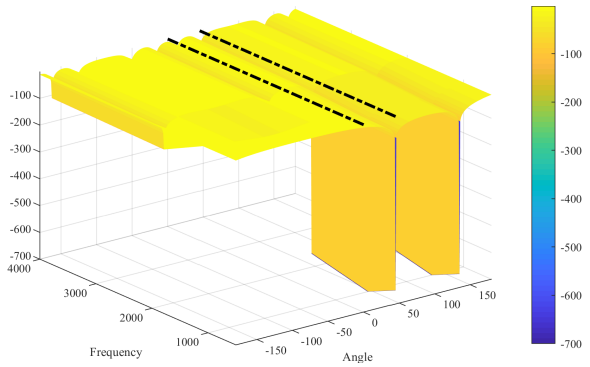
(a) Narrow-band beampattern with 135° interference



(b) Multi-tone beampattern with 135° interference

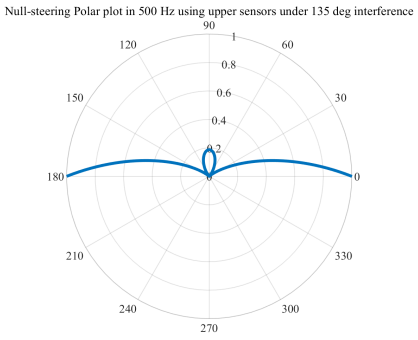


(c) Narrow-band beampattern with 45° interference

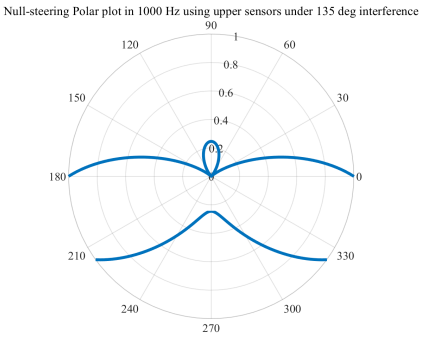


(d) Multi-tone beampattern with 45° interference

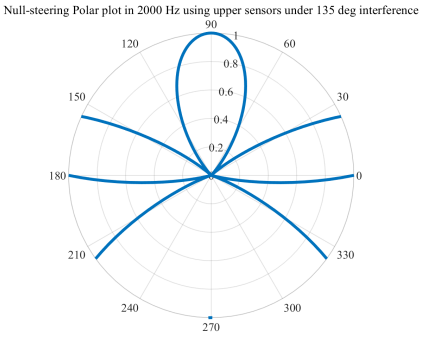
Figure 10: Null-steering beampattern using upper two sensors under different interferences and the same 0° desired source



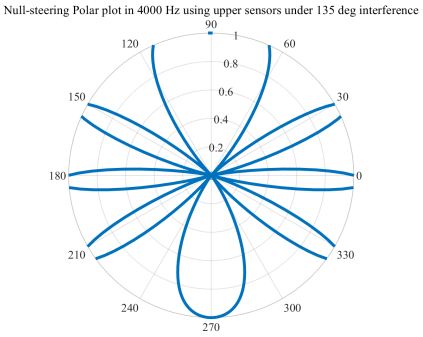
(a) Null-steering polar plot in 500Hz



(b) Null-steering polar plot in 1000Hz

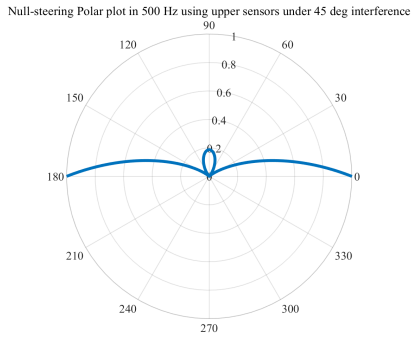


(c) Null-steering polar plot in 2000Hz

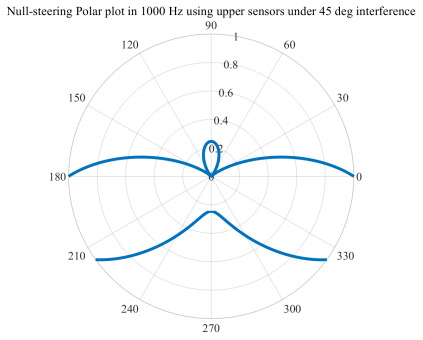


(d) Null-steering polar plot in 4000Hz

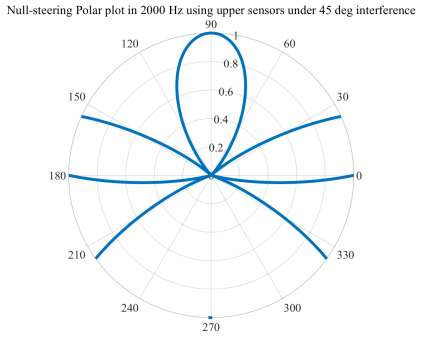
Figure 11: Null-steering polar plots in different frequencies using upper two sensors under 135° interference and 0° desired source



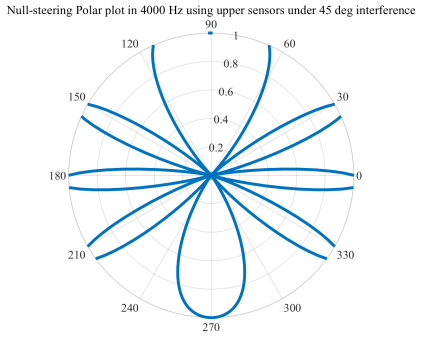
(a) Null-steering polar plot in 500Hz



(b) Null-steering polar plot in 1000Hz

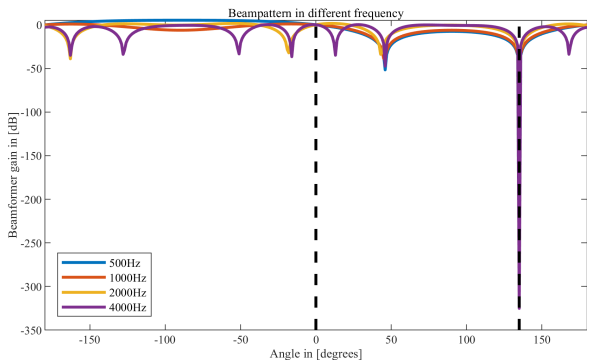


(c) Null-steering polar plot in 2000Hz

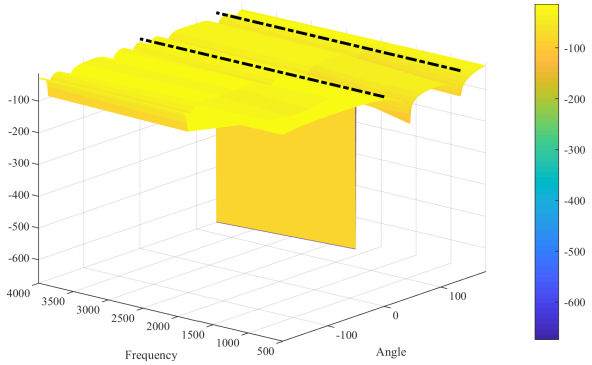


(d) Null-steering polar plot in 4000Hz

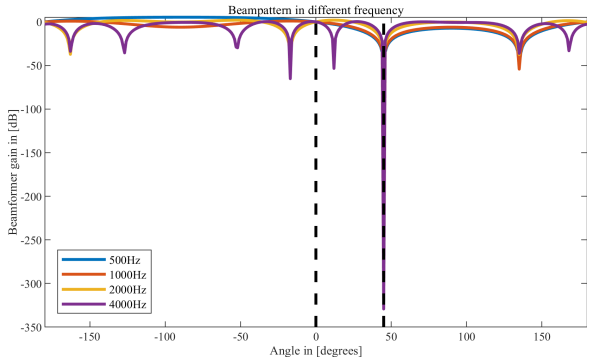
Figure 12: Null-steering polar plots in different frequencies using upper two sensors under 45° interference and 0° desired source



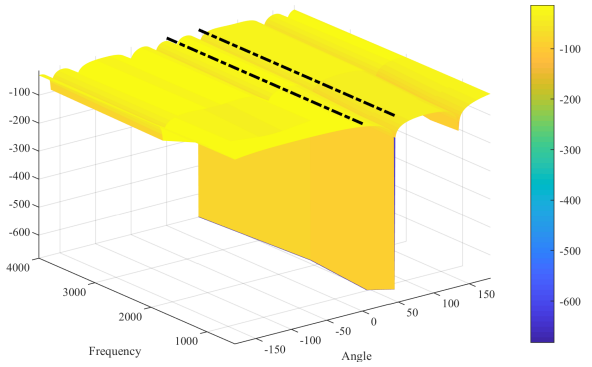
(a) Narrow-band beampattern with 135° interference



(b) Multi-tone beampattern with 135° interference

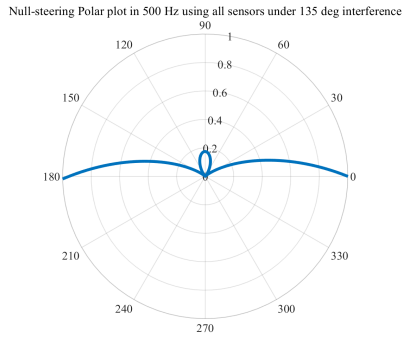


(c) Narrow-band beampattern with 45° interference

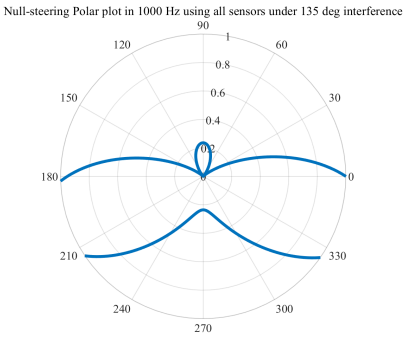


(d) Multi-tone beampattern with 45° interference

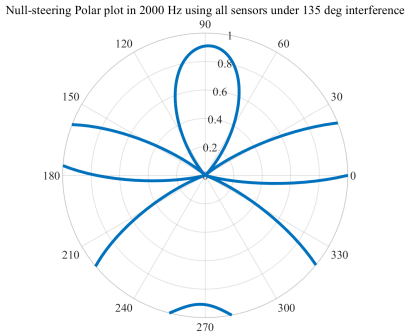
Figure 13: Null-steering beampattern using all four sensors under different interferences and the same 0° desired source



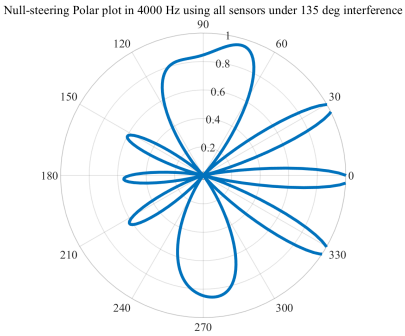
(a) Null-steering polar plot in 500Hz



(b) Null-steering polar plot in 1000Hz

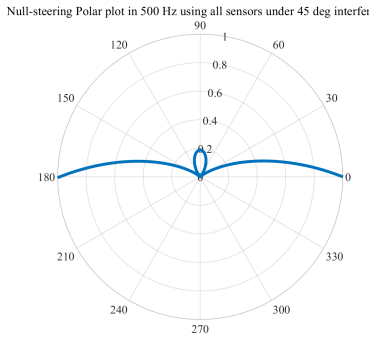


(c) Null-steering polar plot in 2000Hz

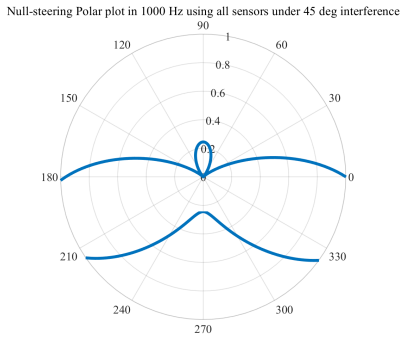


(d) Null-steering polar plot in 4000Hz

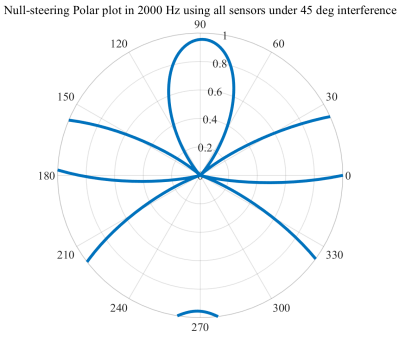
Figure 14: Null-steering polar plots in different frequencies using all four sensors under 135° interference and 0° desired source



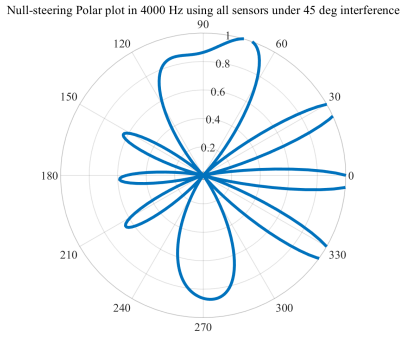
(a) Null-steering polar plot in 500Hz



(b) Null-steering polar plot in 1000Hz



(c) Null-steering polar plot in 2000Hz



(d) Null-steering polar plot in 4000Hz

Figure 15: Null-steering polar plots in different frequencies using all four sensors under 45° interference and 0° desired source

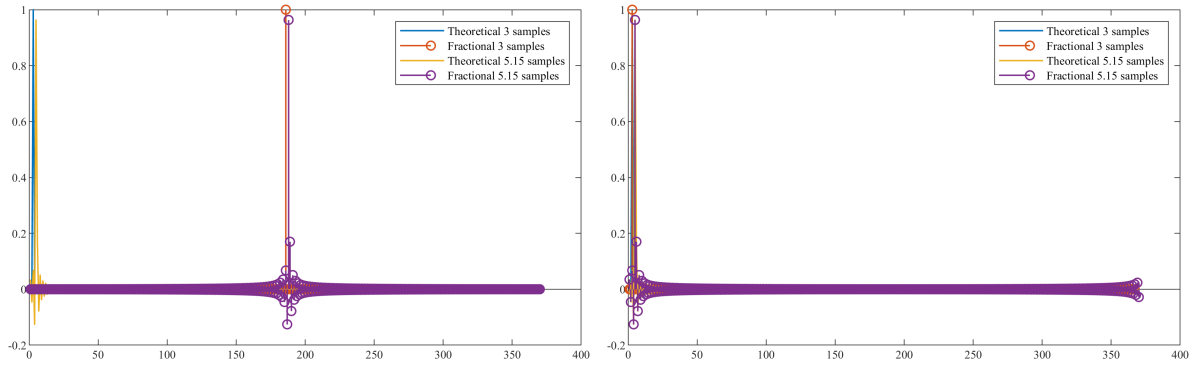
Assignment B: scenario 1: Fractional delay

a) After doing the inverse Fourier transform, we get the impulse response $d[n]$:

$$\begin{aligned} d[n] =& \frac{1}{2\pi}\int_{-\pi}^{\pi} D(e^{j\theta})e^{jn\theta}\,d\theta = \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{-j\tau\theta}e^{jn\theta}\,d\theta \\ =& \frac{1}{2\pi}\int_{-\pi}^{\pi} e^{j(n-\tau)\theta}\,d\theta = \frac{e^{j(n-\tau)\pi} - e^{-j(n-\tau)\pi}}{2\pi j(n-\tau)} \\ =& \frac{\cos\left(\pi(n-\tau)\right) + j\sin\left(\pi(n-\tau)\right) - \left(\cos\left(\pi(n-\tau)\right) - j\sin\left(\pi(n-\tau)\right)\right)}{2\pi j(n-\tau)} \\ =& \frac{\sin\left(\pi(n-\tau)\right)}{\pi(n-\tau)} = Sa(\pi(n-\tau)) = sinc(n-\tau) \end{aligned}$$

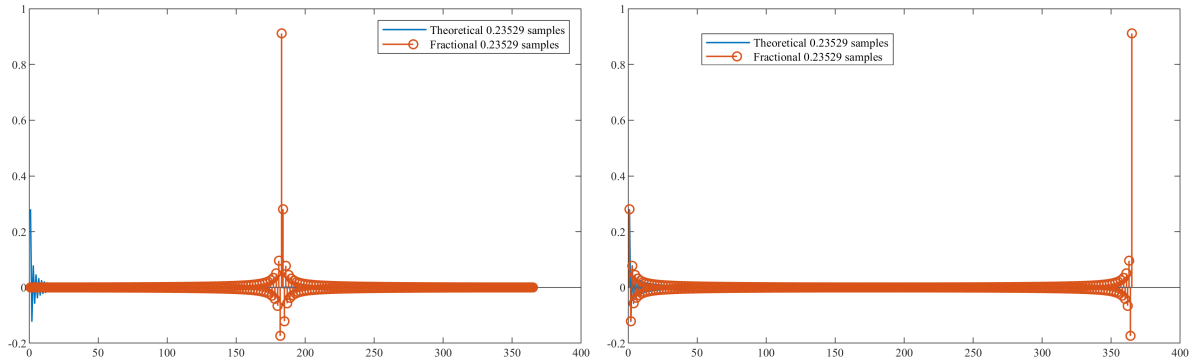
b) Fig.16(a) shows the analytical expressions of impulse responses $d[n]$ and delay function h. From assignment a) we know that $d[n] = sinc(n-\tau)$, when τ is an integer, MATLAB gives a NaN to that, so we shall replace NaN with 1. Fig.16(c) shows the theoretical calculation and delay function h of real time-align delay caused by dy.

c) Fig.16(b) and Fig.16(d) give the plots of aligning delay function h with theoretical values $d[n]$ under different delay samples τ. We find a magic number 183, the difference of the peaks in x-value, which is independent with the length of h. By cyclic shift 183 samples, we plot Fig.16(b) and Fig.16(d). Some of the problems occurred in the tail of Fig.16(b) and Fig.16(d) after cyclic shift. When τ less than 1, such as $\tau = 0.23529$ the tail is quite obvious as we can see in Fig.16(d). To solve this problem, we just drop the first 183 samples in fractional delay h. In the later assignment, we would prove that our approach works and better than cyclic shift. Fig.18 and Fig.19 support our opinions. when we using RLS adaptive algorithm, the filter coefficients are showed in Fig.18 and Fig.19. Obviously, Fig.18(c) and Fig.19(c) are similar with Fig.18(d) and Fig.19(d) respectively which means that drop the first 183 samples are better than cyclic shift align.



(a) Theoretical calculation and delay function

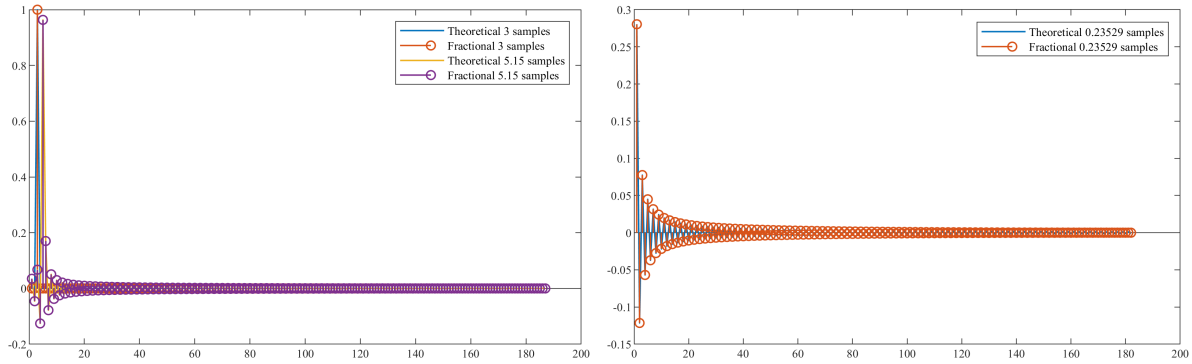
(b) Cyclic shift align



(c) Theoretical calculation and delay function

(d) Cyclic shift align

Figure 16: Fractional delay of different τ



(a) Dropping align

(b) Dropping align

Figure 17: Fractional delay of different τ by dropping the first 183 samples

Assignment B: scenario 2: GSC applied to monaural hearing aid

a) From the output of the upper part we can hear the speech of the desired source and the interference and the background noise, the overall speech doesn't sound very stereoscopic. The delay and sum beamformer is attached in our MATLAB codes.

b) The blocking matrix $\mathbf{B} = [-1, 1]$ or $\mathbf{B} = [1, -1]$ after time aligning for 0° desired source. Time aligning means to compensate the phase shift caused by time delay between sensors. After the blocking of matrix $\mathbf{B}$, we remain only interference signal and noise due to no time aligning for them.

c) Fig.18 and Fig.20 show RLS and NLMS adaptive algorithms using left two microphones. It is obvious from the comparison between Fig.18(c) and Fig.20(c), We choose RLS algorithm with faster convergence and less disturbance. The iterative of adaptive filtering coefficients in Fig.19 and Fig.21 which are using all four sensors also verifies our idea of choosing RLS algorithm.

d) To get the transfer function, we transform the input and output expressions in the time domain to the frequency domain. In the time domain $u_1 = x_1 * h$, $u_2 = x_2$, after delay and sum $v_0 = \frac{1}{2}(u_1 + u_2) \rightarrow d = v_0 * h_\Delta$, blocked by $\mathbf{B}$ we have $v_1 = u_1 - u_2$, to do the adaptive $e = g = v_1 * w$, and then $y = d - e$. So the time-domain y and frequency-domain $\mathbf{Y}$ can be expressed as:

$$\begin{aligned}
y =& \frac{1}{2}(x_1 * h + x_2) * h_\Delta - (x_1 * h - x_2) * w \\
\mathbf{Y} =& \frac{1}{2}(\mathbf{X_1 H} + \mathbf{X_2})\mathbf{H_\Delta} - (\mathbf{X_1 H} - \mathbf{X_2})\mathbf{W} \\
=& (\frac{1}{2}\mathbf{H H_\Delta} - \mathbf{H W})\mathbf{X_1} + (\frac{1}{2}\mathbf{H_\Delta} + \mathbf{W})\mathbf{X_2} \\
=& (\frac{1}{2}e^{-j\tau\theta}e^{-j\theta} - e^{-j\tau\theta}\mathbf{W})\mathbf{X_1} + (\frac{1}{2}e^{-j\theta} + \mathbf{W})\mathbf{X_2}
\end{aligned}$$

So the transfer function under monaural hearing aid scenario is given that:

$$\begin{aligned}
\frac{\mathbf{Y}}{\mathbf{X_1}} =& \frac{1}{2}e^{-j(\tau+1)\theta} - e^{-j\tau\theta}\mathbf{W} \\
\frac{\mathbf{Y}}{\mathbf{X_2}} =& \frac{1}{2}e^{-j\theta} + \mathbf{W}
\end{aligned}$$

We give the detailed analysis of null-steering narrow-band and multi-tone beampattern in scenario 1. And Fig.7, Fig.10 and Fig.13 show the narrow-band and multi-tone beampattern under different interference sources using left two sensors, upper two microphones and all four sensors respectively. Fig.22 shows that use transfer function as filter weights.

Assignment B: scenario 3: GSC applied to binaural hearing aids

a) The detailed analysis of all four microphones is presented in the previous assignments and scenarios. Fig.3(b) shows the array configuration and beampattern before null-steering and Fig.6 gives the polar plots in different frequencies using all four microphones without null-steering. In addition, Fig.13, Fig.14 and Fig.15 describe the null-steering beampattern in narrow-band, multi-tone and polar plots with different interference sources, respectively. What's more, Fig.19 and Fig.21 show the adaptive filter coefficients in iteration using RLS and NLMS algorithms, respectively. And the our blocking matrix is given :

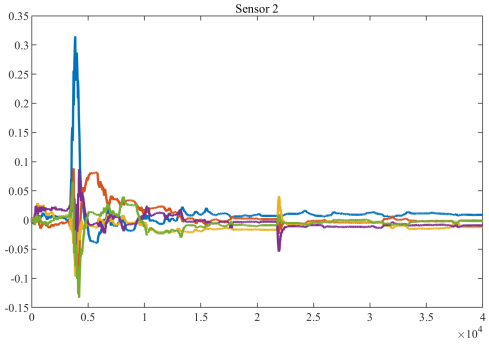
$$\mathbf{B} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \quad or \quad \mathbf{B} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \end{bmatrix}$$

Clearly, other matrices which compensate the phase shift of the desired source caused by time delay between sensors can also work as blocking matrix $\mathbf{B}$. In the same way, we can get the transfer function under binaural hearing aids scenario. In the time domain $u_1 = x_1 * h$, $u_2 = x_2$, $u_3 = x_3$ and $u_4 = x_4 * h$ so after delay and sum we have $v_0 = \frac{1}{4}(u_1 + u_2 + u_3 + u_4)$ and $d = v_0 * h_\Delta$, after blocking matrix, $v_1 = u_1 - u_2$, $v_2 = u_1 - u_3$, $v_3 = u_1 - u_4$ and filtered by adaptive FIR filters we get $g_1 = v_1 * w_1$, $g_2 = v_2 * w_2$, $g_3 = v_3 * w_3 \rightarrow e = \frac{1}{3}(g_1 + g_2 + g_3)$ and then $y = d - e = \frac{1}{4}(x_1 * h + x_2 + x_3 + x_4 * h) * h_\Delta - \frac{1}{3}[(x_1 * h - x_2) * w_1 + (x_1 * h - x_3) * w_2 + (x_1 * h - x_4 * h) * w_3]$. So the frequency-domain $\mathbf{Y}$:

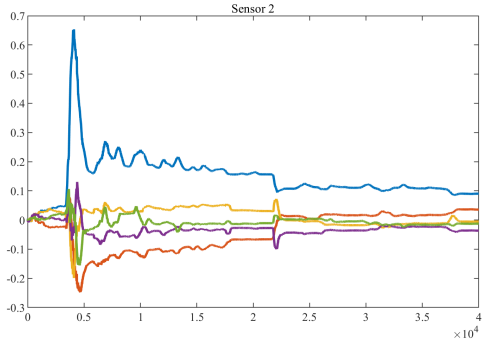
$$\begin{aligned}
\mathbf{Y} =& \frac{1}{4}(\mathbf{X_1 H} + \mathbf{X_2} + \mathbf{X_3} + \mathbf{X_4 H})\mathbf{H_\Delta} - \frac{1}{3}[(\mathbf{X_1 H} - \mathbf{X_2})\mathbf{W_1} + (\mathbf{X_1 H} - \mathbf{X_3})\mathbf{W_2} + (\mathbf{X_1 H} - \mathbf{X_4 H})\mathbf{W_3}] \\
=& [\frac{1}{4}\mathbf{H H_\Delta} - \frac{1}{3}\mathbf{H}(\mathbf{W_1} + \mathbf{W_2} + \mathbf{W_3})]\mathbf{X_1} + (\frac{1}{4}\mathbf{H_\Delta} + \frac{1}{3}\mathbf{W_1})\mathbf{X_2} \\
&+ (\frac{1}{4}\mathbf{H_\Delta} + \frac{1}{3}\mathbf{W_2})\mathbf{X_3} + (\frac{1}{4}\mathbf{H H_\Delta} + \frac{1}{3}\mathbf{H W_3})\mathbf{X_4}
\end{aligned}$$

So the transfer function under binaural hearing aids scenario is given that:

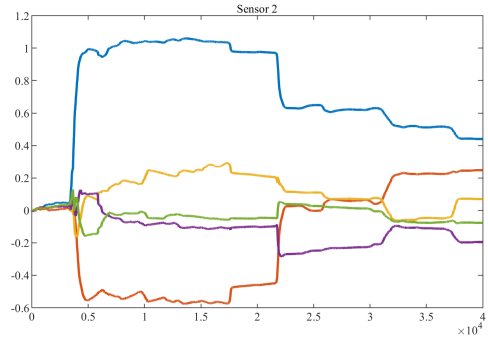
$$\begin{aligned}
\frac{\mathbf{Y}}{\mathbf{X_1}} =& \frac{1}{4}\mathbf{H H_\Delta} - \frac{1}{3}\mathbf{H}(\mathbf{W_1} + \mathbf{W_2} + \mathbf{W_3}) = \frac{1}{4}e^{-j(\tau+1)\theta} - \frac{1}{3}e^{-j\tau\theta}(\mathbf{W_1} + \mathbf{W_2} + \mathbf{W_3}) \\
\frac{\mathbf{Y}}{\mathbf{X_2}} =& \frac{1}{4}\mathbf{H_\Delta} + \frac{1}{3}\mathbf{W_1} = \frac{1}{4}e^{-j\theta} + \frac{1}{3}\mathbf{W_1} \\
\frac{\mathbf{Y}}{\mathbf{X_3}} =& \frac{1}{4}\mathbf{H_\Delta} + \frac{1}{3}\mathbf{W_2} = \frac{1}{4}e^{-j\theta} + \frac{1}{3}\mathbf{W_2} \\
\frac{\mathbf{Y}}{\mathbf{X_4}} =& \frac{1}{4}\mathbf{H H_\Delta} + \frac{1}{3}\mathbf{H W_3} = \frac{1}{4}e^{-j(\tau+1)\theta} - \frac{1}{3}e^{-j\tau\theta}\mathbf{W_3}
\end{aligned}$$



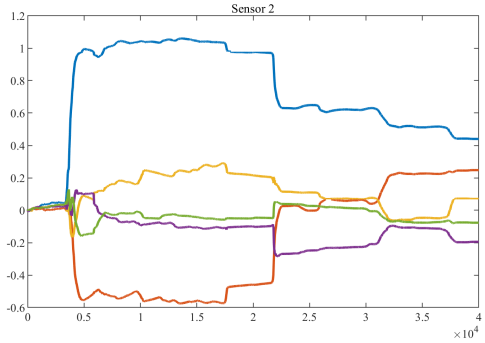
(a) Original delay function as delay filter



(b) Cyclic shift align as delay filter

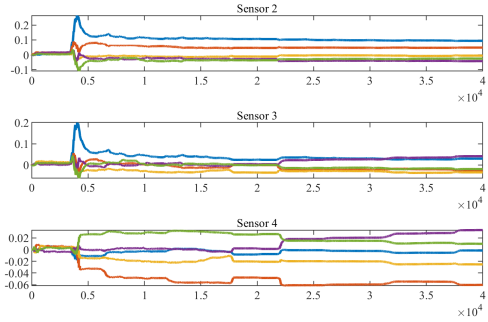


(c) Dropping align as delay filter

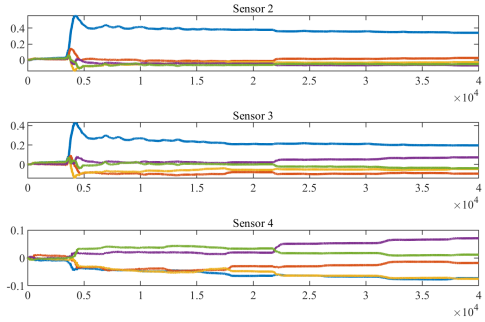


(d) Theoretical calculation as delay filter

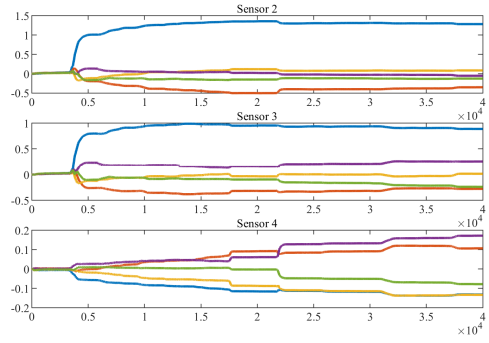
Figure 18: RLS adaptive filter coefficient using different delay filter under left two sensors



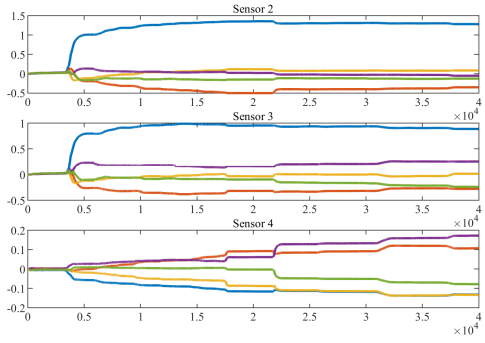
(a) Original delay function as delay filter



(b) Cyclic shift align as delay filter

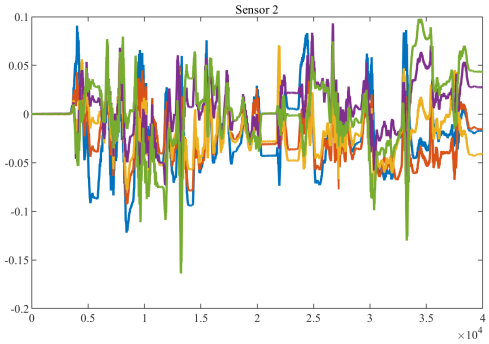


(c) Dropping align as delay filter

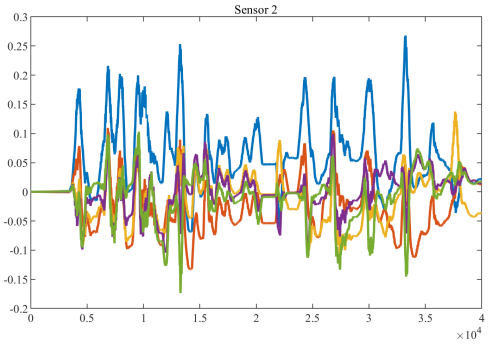


(d) Theoretical calculation as delay filter

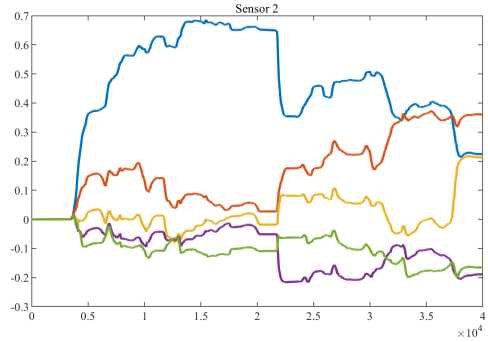
Figure 19: RLS adaptive filter coefficient using different delay filter under all four sensors



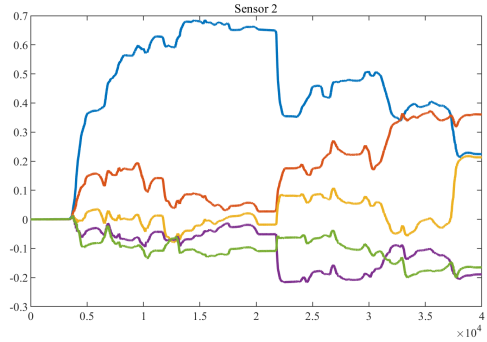
(a) Original delay function as delay filter



(b) Cyclic shift align as delay filter

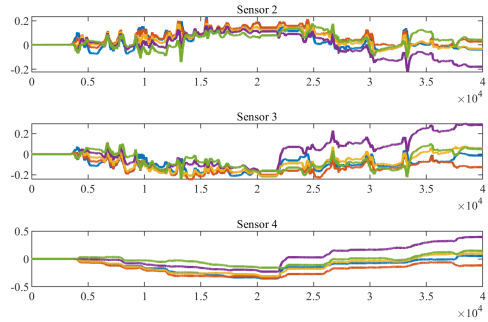


(c) Dropping align as delay filter

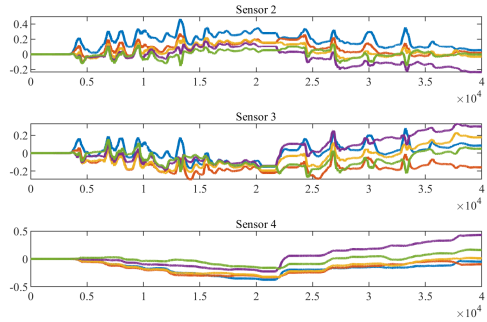


(d) Theoretical calculation as delay filter

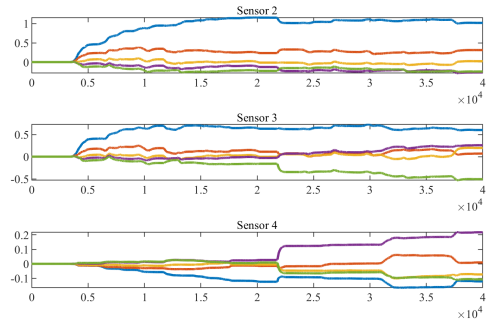
Figure 20: NLMS adaptive filter coefficient using different delay filter under left two sensors



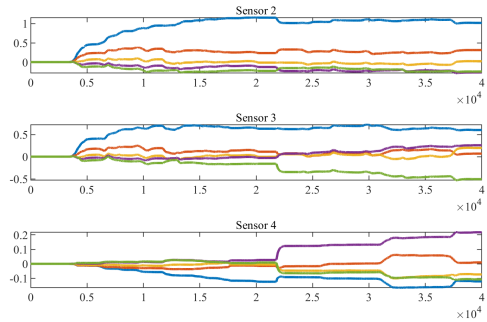
(a) Original delay function as delay filter



(b) Cyclic shift align as delay filter

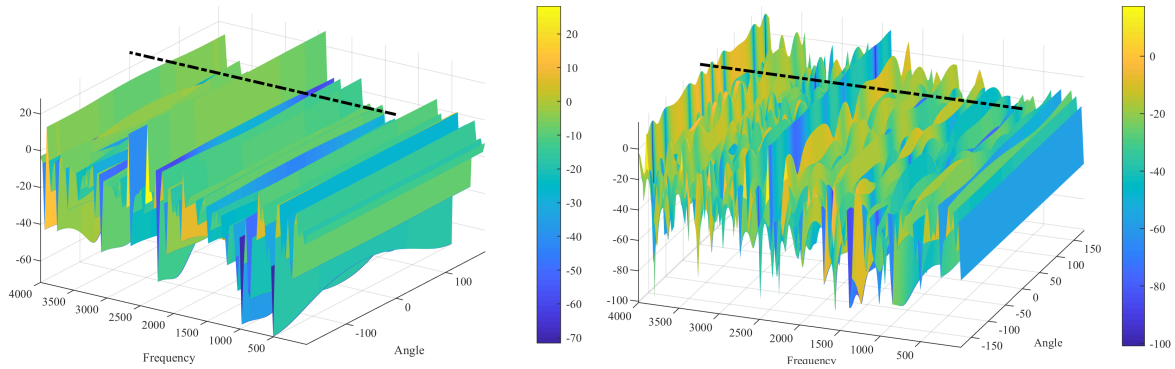


(c) Dropping align as delay filter



(d) Theoretical calculation as delay filter

Figure 21: NLMS adaptive filter coefficient using different delay filter under all four sensors



(a) Transfer function as filter under left two sensors

(b) Transfer function as filter under all four sensors

Figure 22: The multi-tone beampattern of using transfer function as filter weights

Assignment B: scenario 4: Wireless binaural hearing aids

a) To reduce the amount of data exchanged, we choose to pre-process the raw data before conducting wireless transmission by antennas. This can reduce the power consumption of digital signal processing chips which will execute our GSC adaptive algorithms. Although we reduce the power consumption of adaptive processing, the power consumption of other chips will increase correspondingly. This is a trade-off.

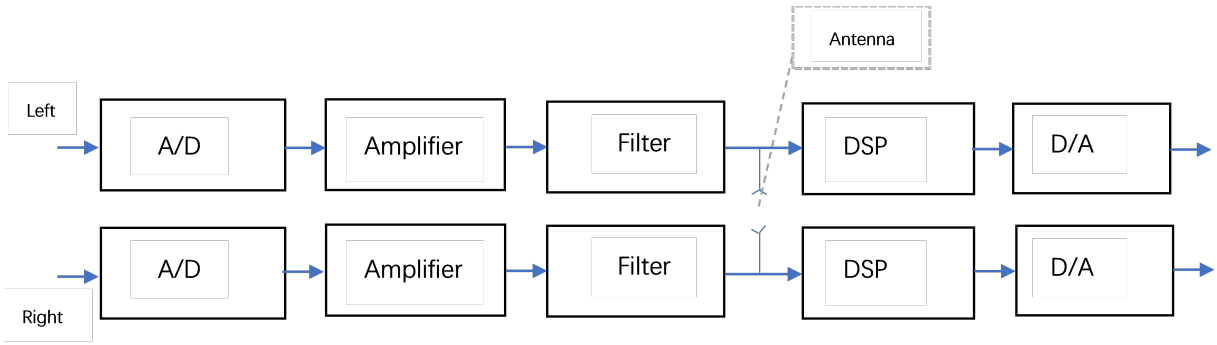


Figure 23: Data exchange flow diagram

The MATLAB codes is attached below:

```
clear all;
close all;
clear classes;
clc;
%% Array Configuration
Scenario = 'all';
switch Scenario
    case 'left'
        dx = 0;
        dy = 0.01;
        J = 2;
        my_array = arrays.ULA(J,dx,dy);
        disp('The left two microphones are ready !');
        figure(1);
        my_array.plot();
        flag = 0;
    case 'upper'
        dx = 0.17;
        dy = 0;
        J = 2;
        my_array = arrays.ULA(J,dx,dy);
        disp('The upper two microphones are ready !');
        figure(1);
        my_array.plot();
        flag = 2;
    case 'all'
        dx = 0.17;
```

```
        dy = 0.01;
        J = 4;
        my_array = arrays.square_array(J,dx,dy);
        disp('All four microphones are ready !');
        figure(1);
        my_array.plot();
        flag = 1;
end


%% Beamformer
mt_f = [500 1000 2000 4000];            % multiple tones in Hz
c = 340;                                % the sound speed
theta = [0,45];                         % row vector containing angles for which ...
    constraints hold
target = [1,0];                         % row vector containing target values for the ...
    beampattern


b = beamformer;
set(b, 'array',         my_array);
set(b, 'angles',        -180:1:180);
set(b, 'n_sensor',      J);
set(b, 'sound_speed',   c);
set(b, 'mt_frequency',  mt_f);

%% Pattern with Weights 1/J
figure(2)
for i = 1:length(mt_f)
nb_f = mt_f(i);
set(b, 'nb_frequency',  nb_f);
b.nb_weights = 1/J*ones(J,1);
b.calc_nb_beampattern;
Beam = b.nb_beampattern;
plot(b.angles, 10*log10(Beam),'LineWidth',2);
%b.plot_nb;
hold on;
end
legend('500Hz','1000Hz','2000Hz','4000Hz','Location','southwest')
axis tight
title('Beampattern in different frequency')
xlabel('Angle in [degrees]');
ylabel('Beamformer gain in [dB]');

b.mt_weights = 1/J*ones(J,length(mt_f));
b.calc_mt_beampattern;
figure(3)
b.plot_mt;

for i = 1:length(mt_f)
nb_f = mt_f(i);
set(b, 'nb_frequency',  nb_f);
b.nb_weights = 1/J*ones(J,1);
b.calc_nb_beampattern;
Beam = b.nb_beampattern;
figure(100+i)
polarplot(b.angles.'*pi/180,Beam,'LineWidth',2);
rlim([0 1])
title(['Polar plot in ',num2str(mt_f(i)),' Hz using ', Scenario, ' sensors']);
end


%% Null-steering pattern
figure(4)
for i = 1:length(mt_f)
nb_f = mt_f(i);
set(b, 'nb_frequency',  nb_f);
b.beam_steering_nb(theta, target);
b.calc_nb_beampattern;
Beam = b.nb_beampattern;
plot(b.angles, 10*log10(Beam),'LineWidth',2);
hold on;
end
plot([theta;theta], [-350 max(10*log(Beam))]'* ones(1,length(theta)),'k--','LineWidth',2);
legend('500Hz','1000Hz','2000Hz','4000Hz','Location','southwest')
axis tight
title('Beampattern in different frequency')
```

```
xlabel('Angle in [degrees]');
ylabel('Beamformer gain in [dB]');


b.beam_steering_mt(theta, target);
b.calc_mt_beampattern;
figure(5);
b.plot_mt(theta, {'k-.','LineWidth',2});

for i = 1:length(mt_f)
nb_f = mt_f(i);
set(b, 'nb_frequency',  nb_f);
b.beam_steering_nb(theta, target);
b.calc_nb_beampattern;
Beam = b.nb_beampattern;
figure(200+i)
polarplot(b.angles.'*pi/180,Beam,'LineWidth',2);
rlim([0 1])
title(['Null-steering Polar plot in ',num2str(mt_f(i)),' Hz using ', Scenario, ' ...
    sensors under ',num2str(theta(2)),' deg interference']);
end


%% Fractional Delay
if flag ~= 2
Dataset= load('DatasetAssignBs2.mat');
fs = Dataset.fs;
time_align = dy/c * fs;
tau = time_align;
%tau = [3,5.15];
%tau = [15.551,42.443];
disp(['Time align = ',num2str(tau),' samples !']);

legend_the = cell(1,length(tau));
legend_fra = cell(1,length(tau));
for i = 1:length(tau)
[h,my_h] = Fractional_delay(tau(i));
my_h(isnan(my_h)) = 1;
legend_the{i} = ['Theoretical ' num2str(tau(i)) ' samples'];
legend_fra{i} = ['Fractional ' num2str(tau(i)) ' samples'];
figure(50)
plot(my_h,'LineWidth',1)
hold on;
stem(h,'LineWidth',1)
hold on;
figure(51)
plot(my_h(1:end-183),'LineWidth',1)
hold on;
%stem(circshift(h,length(h)-183),'LineWidth',1)
stem(h(184:end),'LineWidth',1)
hold on;
end
%legend({legend_the});
if length(tau) == 1
    figure(50)
    legend([legend_the, legend_fra]);
    figure(51)
    legend([legend_the, legend_fra]);
else
figure(50)
legend(legend_the{1}, legend_fra{1},legend_the{length(tau)},legend_fra{length(tau)});
figure(51)
legend(legend_the{1}, legend_fra{1},legend_the{length(tau)},legend_fra{length(tau)});
end
h = circshift(h,length(h)-183);
%h = h(184:end);

%% Time Align
x = Dataset.x(1:J,:);
u = x;
u(1,:) = filter(h,1,x(1,:).').';
%u(1,:) = filter(my_h,1,x(1,:).').';
if flag == 1
    u(J,:) = filter(h,1,x(J,:).').';
    %u(J,:) = filter(my_h,1,x(J,:).').';
end
v0 = 1/J * sum(u);
```

```
d = v0;
d(:,1:end-1) = v0(:,2:end);
d(:,end) = v0(:,1);    % d(:,end) = 0;

%d(:,1:end-183) = v0(:,184:end);
%d(:,end-182:end) = v0(:,1:183);


%v0 = (v0 - sum(v0)./length(v0))./std(v0);

soundsc(v0,fs);
disp('Listen,the upper branch audio is playing !');

%audiowrite('GSC_upper_branch.wav',v0,fs);
%v = u(1,:) - u(J,:);
%soundsc(v,fs);
%B = [-1 1];
B(:,1) = -1*ones(J-1,1);
B(:,2:J) = eye(J-1);
v = B*u;

%% Adaptive
%w_plot = [];
state = Init_A1s2(J);
w_plot = zeros(state.nw,J-1,length(d));
for i = 1:length(d)
    state = Update_A1s2(state,d(i),v(:,i).');
    %w_plot = [w_plot,state.w];
    w_plot(:,:,i) = state.w;
end
w_a = state.w;
g_temp = zeros(J-1,length(d));
for i = 1:J-1
    g_temp(i,:) = filter(w_a(:,i),1,v(i,:));
end
%g = filter(w_a,1,v);
g = 1/(J-1) * sum(g_temp,1);
y = d - g;

%soundsc(y,fs);
disp('Listen,the output audio is playing !');

figure
%for i = 1:state.nw
%    plot(w_plot(i,:));
%    hold on;
%end
for i = 1:J-1
subplot(J-1,1,i);
    for j = 1:state.nw
        plot(squeeze(w_plot(j,i,:)),'LineWidth',1.5);
        hold on;
    end
title(['Sensor ' num2str(i+1)]);
end

else
   disp('Warning!! There is no delay!');
end

%% Trans function
theta = 0;                      % desired source angle
n_fft = 256;
mt_f = linspace(1,fs/2,n_fft/2);

set(b, 'mt_frequency',  mt_f);

Y = fft(y,n_fft);
X = fft(x.',n_fft).';
Trans = Y./X;
b.mt_weights = Trans(:,1:n_fft/2);

%for i = 1:J
%    b.mt_weights(i,:) = fft(y)./fft(x(i,:));
%end
%b.mt_weights = b.mt_weights(:,1:end/2);
%for i = 1:J
```

```
%    b.mt_weights(i,:) = ifft(fft(y)./fft(x(i,:)));
%end
%for i =1:J
%b.mt_weights(i,:) = y./x(i,:);
%end

b.calc_mt_beampattern;
figure(1000);
b.plot_mt(theta, {'k-.','LineWidth',2});


%% Function
function [h,my_h] = Fractional_delay(tau)
dmax = ceil(tau);
d_int = fix(tau);
index_dot = find(num2str(tau)== '.');
if isempty(index_dot)
    L = 10;
    fd = 0;
else
    L =10^(length(num2str(tau)) - index_dot);
    fd = fix((tau - d_int)*L);
end
h = delay(d_int,fd,L,dmax);
%H =10*log10(abs(fft(h)));
%% Analytical expression
len_filter = length(h);
%half = fix((len_filter-1)/2);
%n = linspace(-half,half,len_filter);
n = 1:len_filter;
%n = 1:len_filter-183;
my_h = sin(pi*(n - tau))./(pi*(n - tau));
end

function [ state ] = Init_A1s2(J)
% Initialize all variables that you want to use:
deta = 0.1;        %0.1
state.J = J;
state.alpha =0.00008;  %0.0008
state.nw =5;
state.var = 0.5*ones(1,state.J-1);
state.w = zeros(state.nw,state.J-1);
state.x = zeros(state.nw,state.J-1);
state.R_inv = deta * eye(state.nw);
end

function state = Update_A1s2( state, x1, x2 )
alpha = state.alpha;
e = x1;
state.x(2:end,:) = state.x(1:end-1,:);     % this is a vector
state.x(1,:) = x2;
%r = e - state.w.' * state.x;
r = e - (1/(state.J-1)) * sum(diag(state.w.' * state.x));

%rls;
nlms;
%% RLS
function rls
lamda = 0.9999999;             %deta = 0.1
%lamda = 1;
gain = (state.R_inv * state.x)./(lamda^2 + diag(state.x.' * state.R_inv * state.x)).';
state.R_inv = (state.R_inv - gain * state.x.' * state.R_inv)./(lamda^2);
state.w = state.w + r * gain;
end

%% NLMS
function nlms
beta = 0.9999999;  %0.5
%beta = 1;
state.var = beta*state.var + (1-beta)*((diag(state.x.'*state.x)).'./state.nw);
state.w = state.w + 2*alpha*r*state.x./state.var;
end
end
```