

# ADSP Final Project

Gengjianhua and Song Ruibo

December 17, 2019

## Assignment B: scenario 1: Analyse hearing aid array configuration

a) Fig.1 ~ 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  $\theta = 0^\circ$  and  $180^\circ$ . The polar plots Fig.4 support our opinions. For the reason that the ratio  $\frac{d}{\lambda} \ll \frac{1}{2}$  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  $f = 1000\text{Hz}$ , 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  $0^\circ$  and other spatial aliasing repeated pattern due to the role of tiny  $dy$ .

b) Fig.7 ~ Fig.15 show the null-steering beampatterns in different frequencies with  $0^\circ$  desired source and  $135^\circ$  or  $45^\circ$  interferences under different array configurations. Obviously, Fig.7 ~ Fig.15 meet the basics requirements that set unit gain to desired source at  $0^\circ$  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  $135^\circ$  and  $45^\circ$  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  $135^\circ$  and  $45^\circ$  interference but they are even symmetric about  $0^\circ$ . What's more, the upper two sensor configuration results in a sine type beampattern  $\sin \theta = \sin(180^\circ - \theta)$  so they are exactly the same with  $135^\circ$  and  $45^\circ$  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.

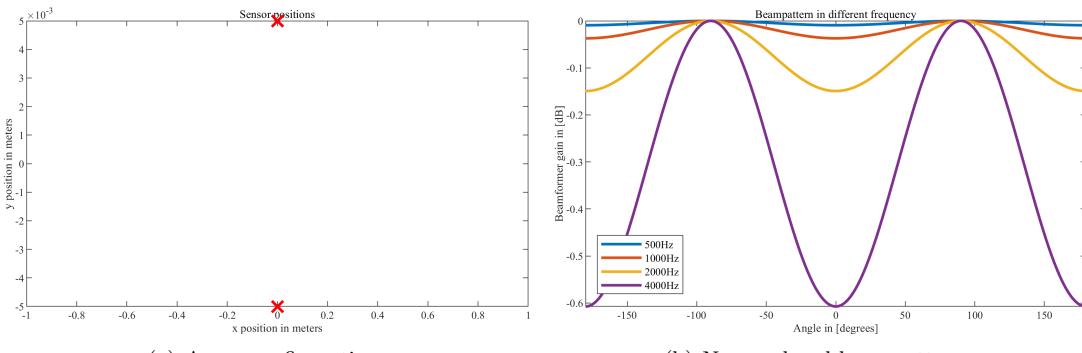


Figure 1: Beampattern of left two sensors before null-steering

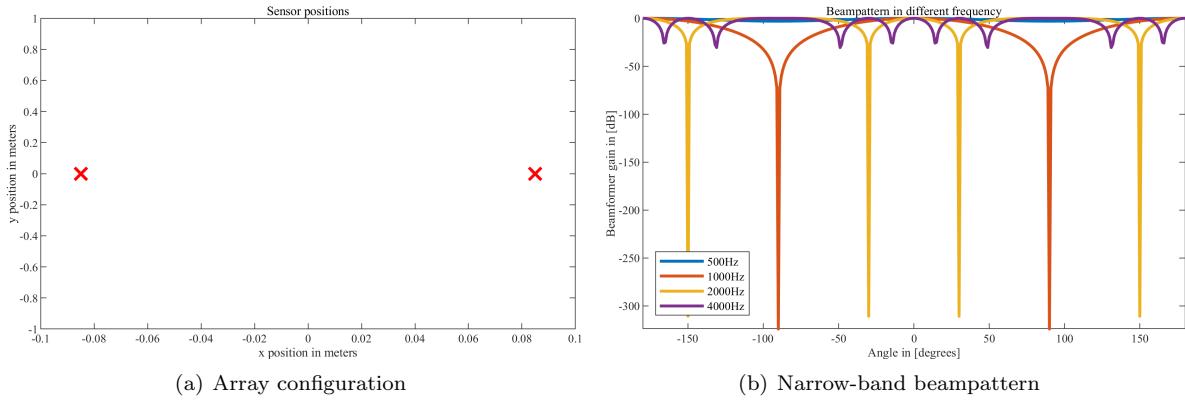


Figure 2: Beampattern of upper two sensors before Null-steering

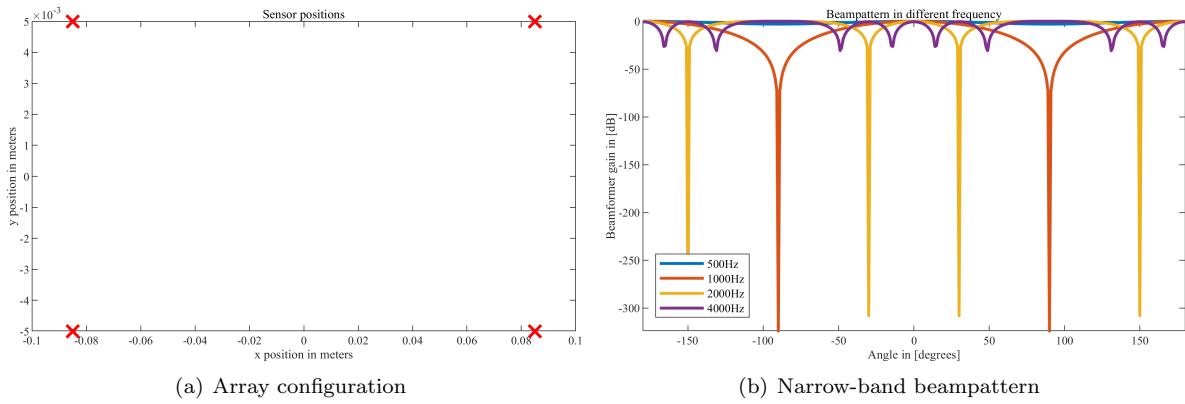


Figure 3: Beampattern of all four sensors before Null-steering

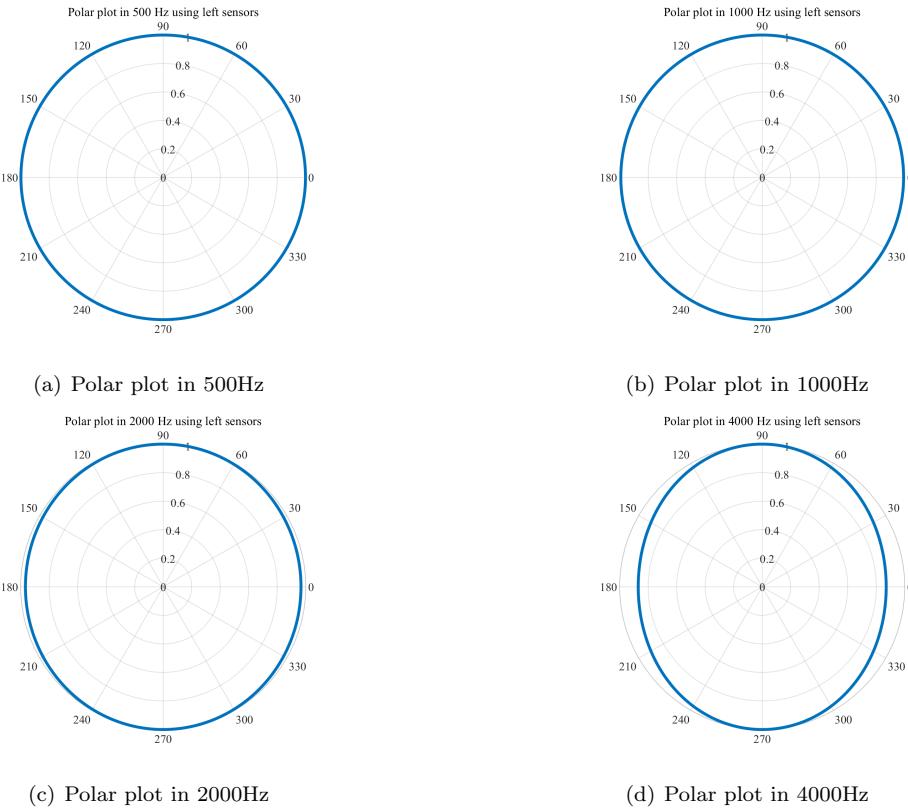


Figure 4: Polar plots in different frequencies using left two sensors before null-steering

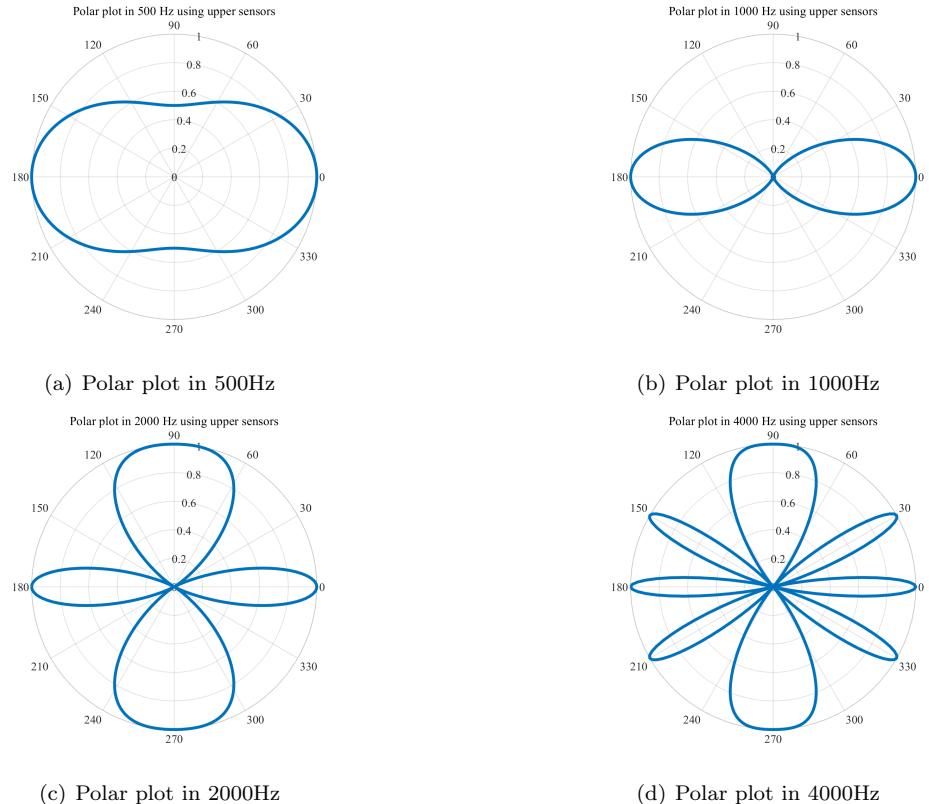


Figure 5: Polar plots in different frequencies using upper two sensors before null-steering

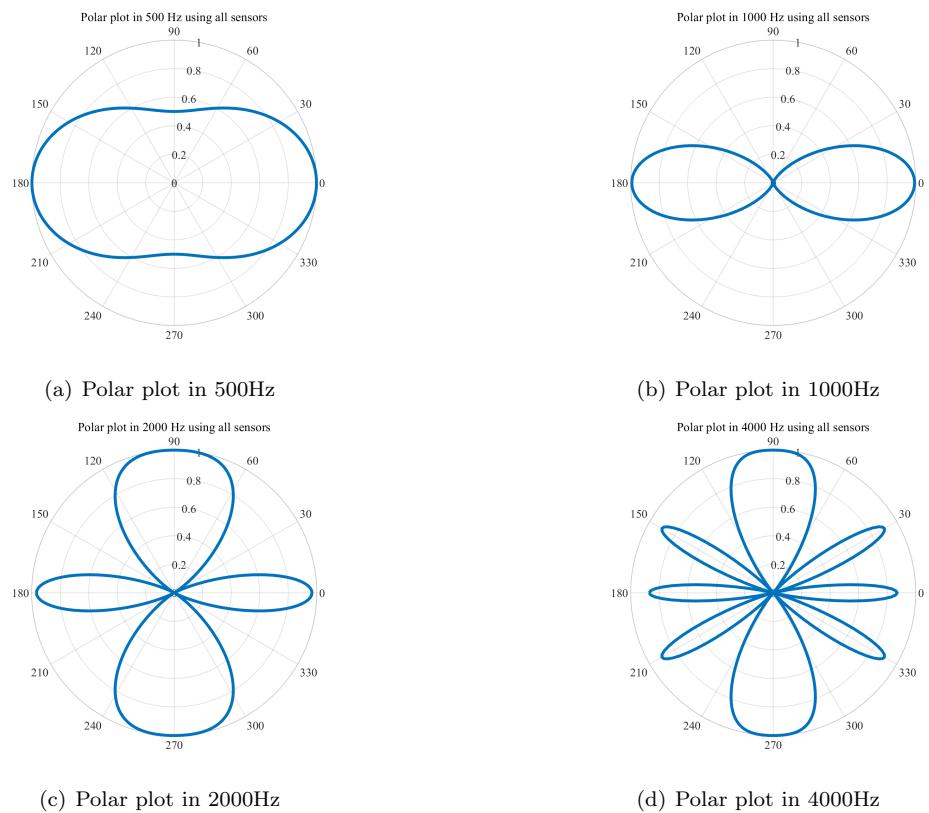


Figure 6: Polar plots in different frequencies using all four sensors before null-steering

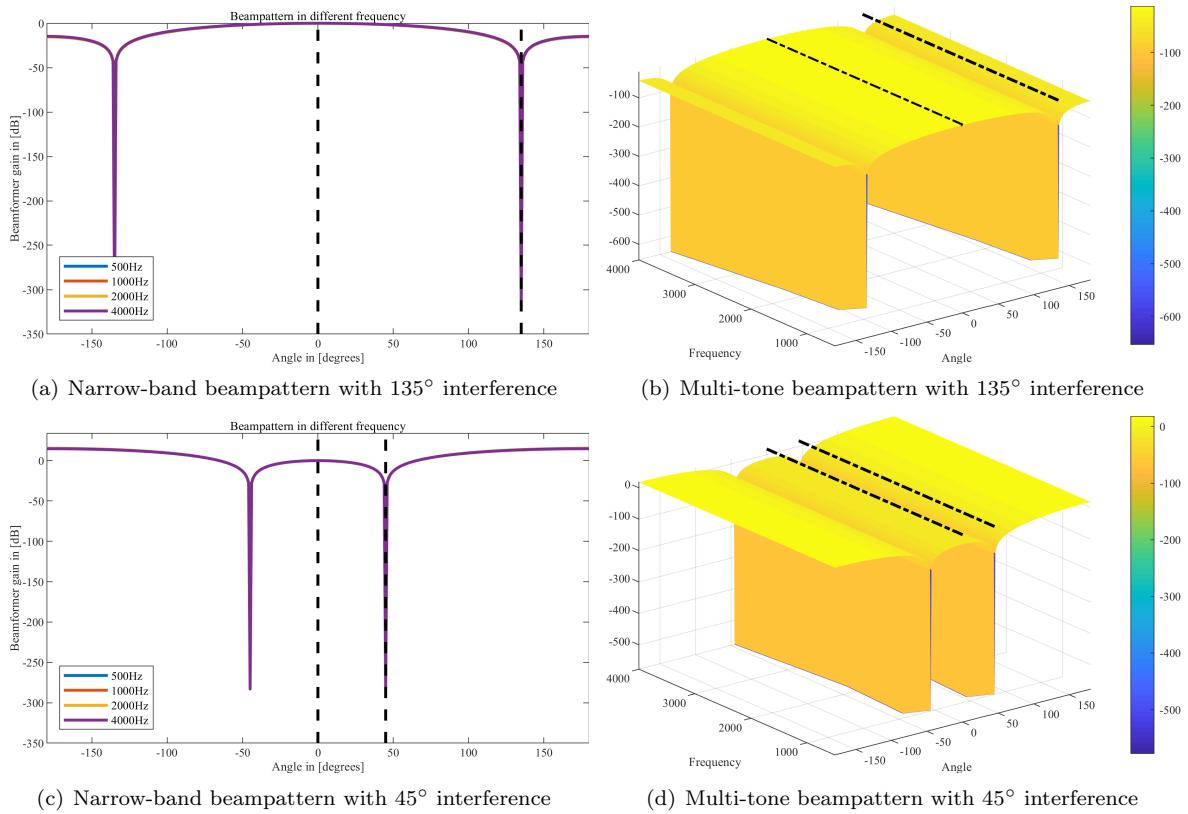


Figure 7: Null-steering beampattern using left two sensors under different interferences and the same  $0^\circ$  desired source

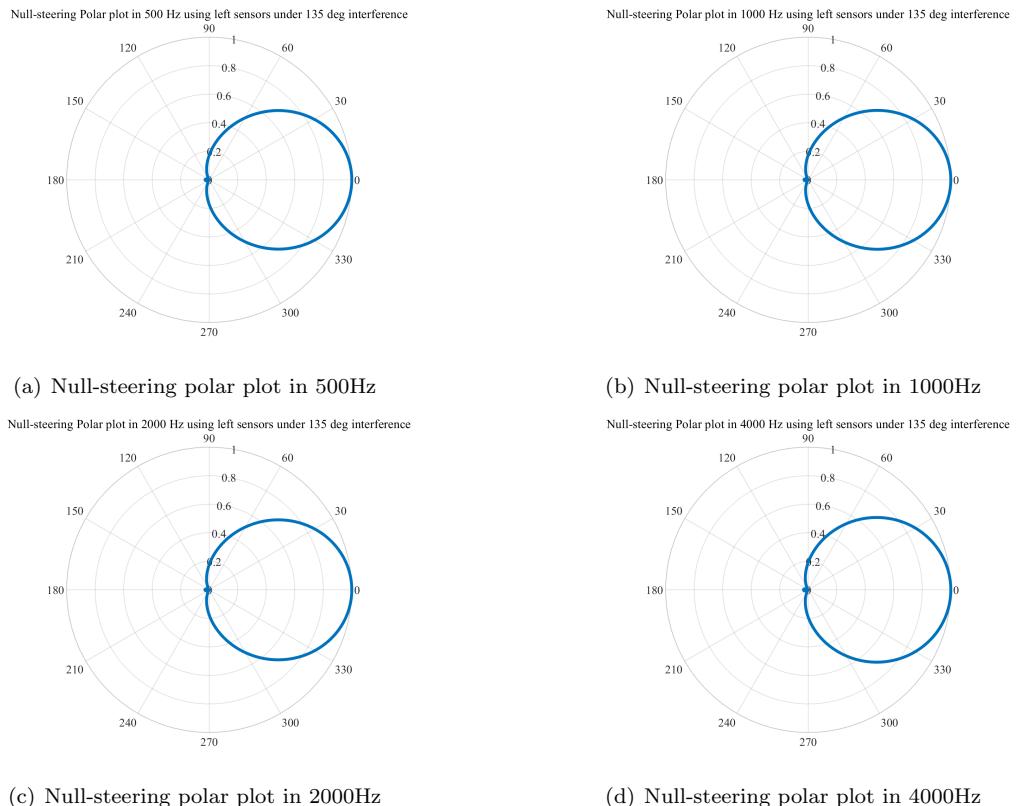
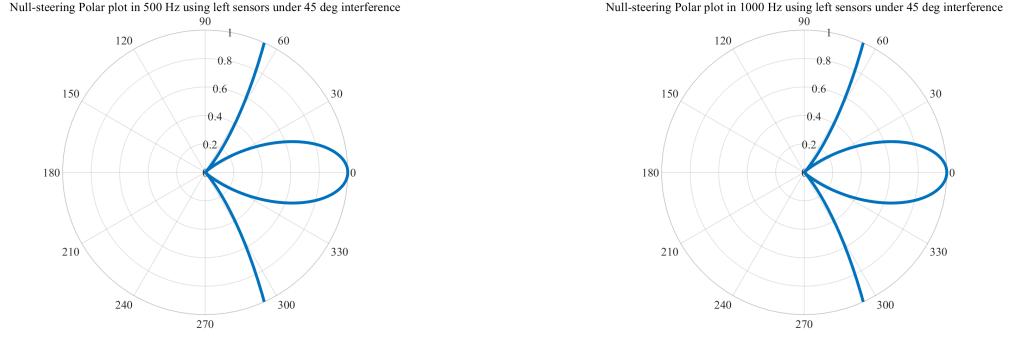
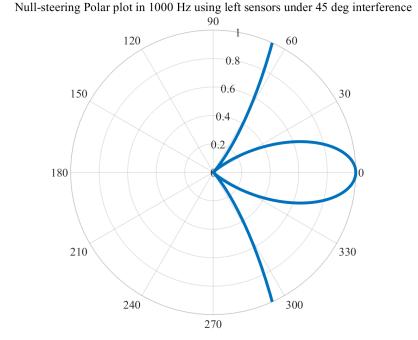


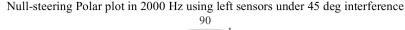
Figure 8: Null-steering polar plots in different frequencies using left two sensors under  $135^\circ$  interference and  $0^\circ$  desired source



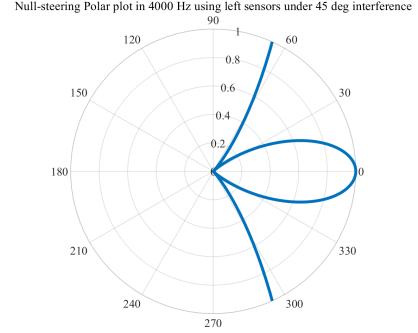
(a) Null-steering polar plot in 500Hz



(b) Null-steering polar plot in 1000Hz

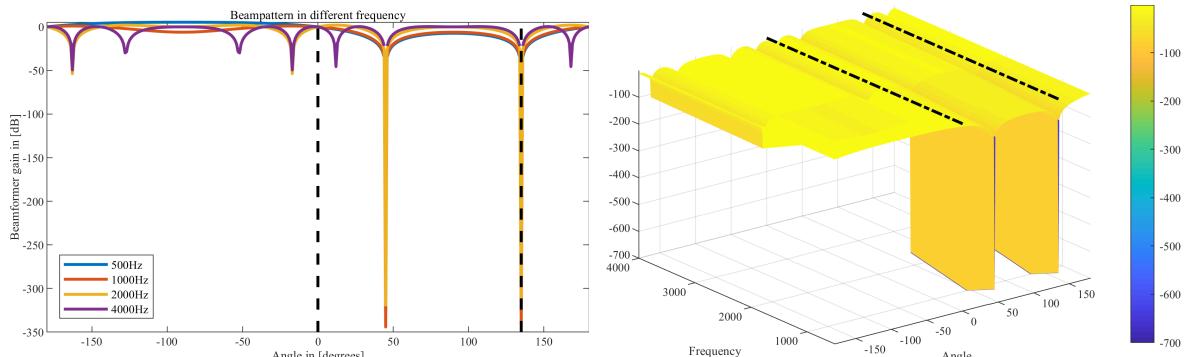


(c) Null-steering polar plot in 2000Hz

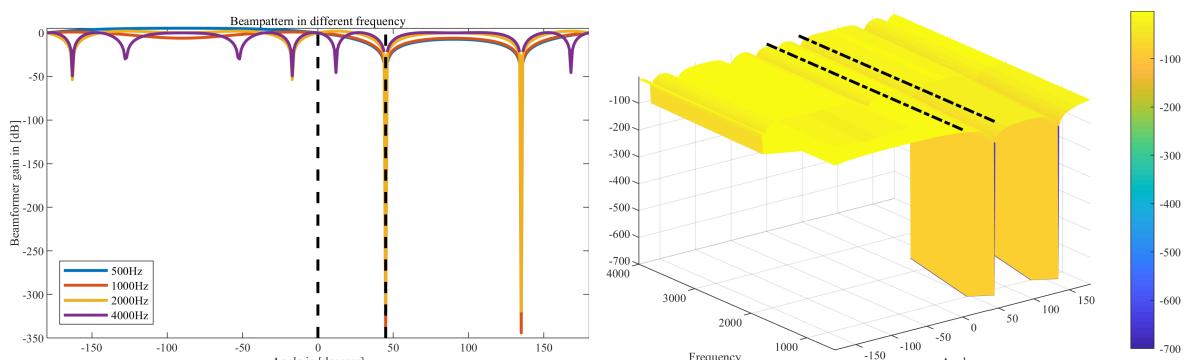


(d) Null-steering polar plot in 4000Hz

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



(b) Multi-tone beampattern with  $135^\circ$  interference



(d) Multi-tone beampattern with  $45^\circ$  interference

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

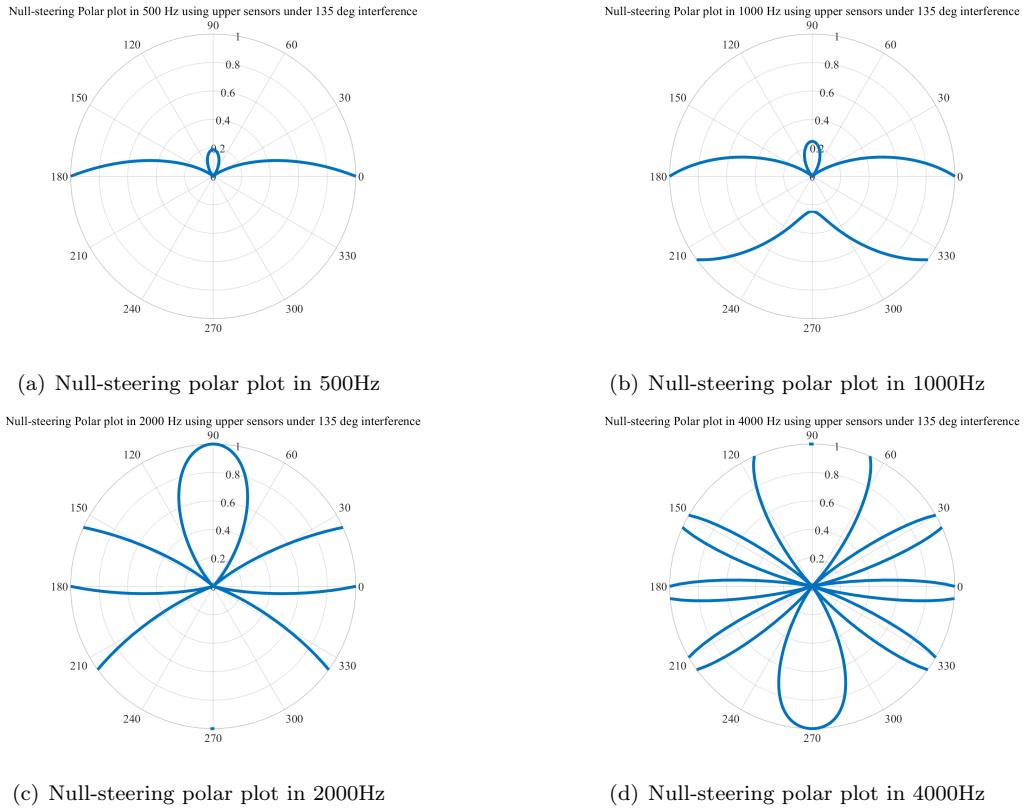


Figure 11: Null-steering polar plots in different frequencies using upper two sensors under  $135^\circ$  interference and  $0^\circ$  desired source

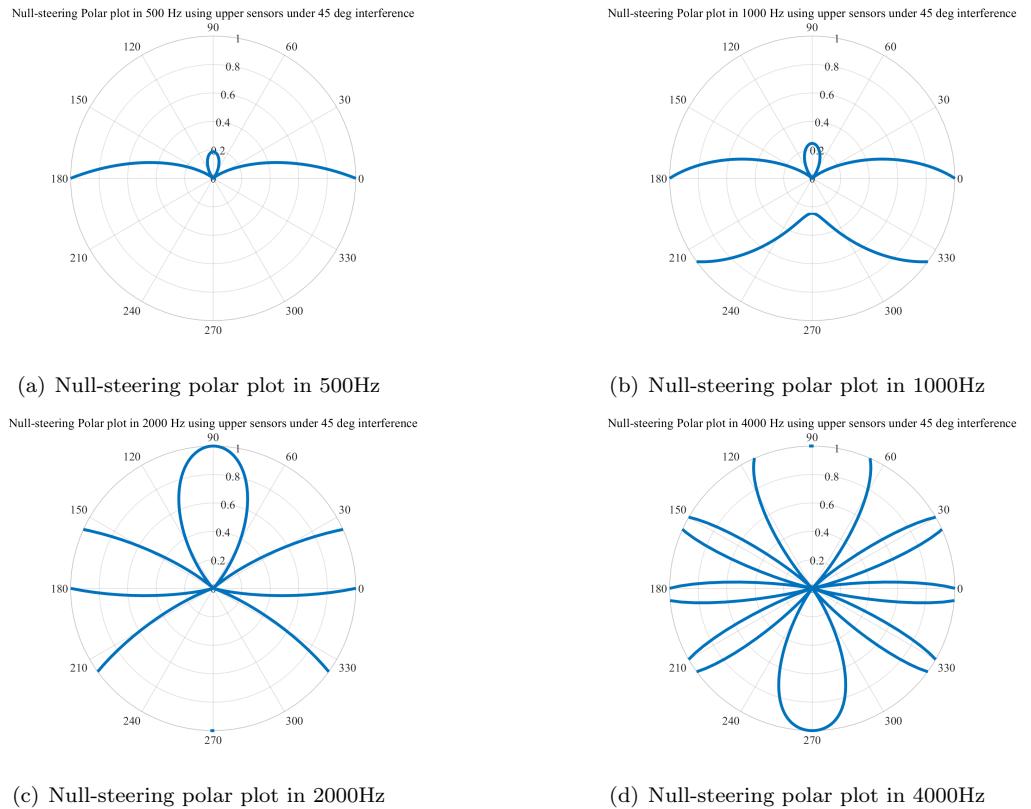


Figure 12: Null-steering polar plots in different frequencies using upper two sensors under  $45^\circ$  interference and  $0^\circ$  desired source

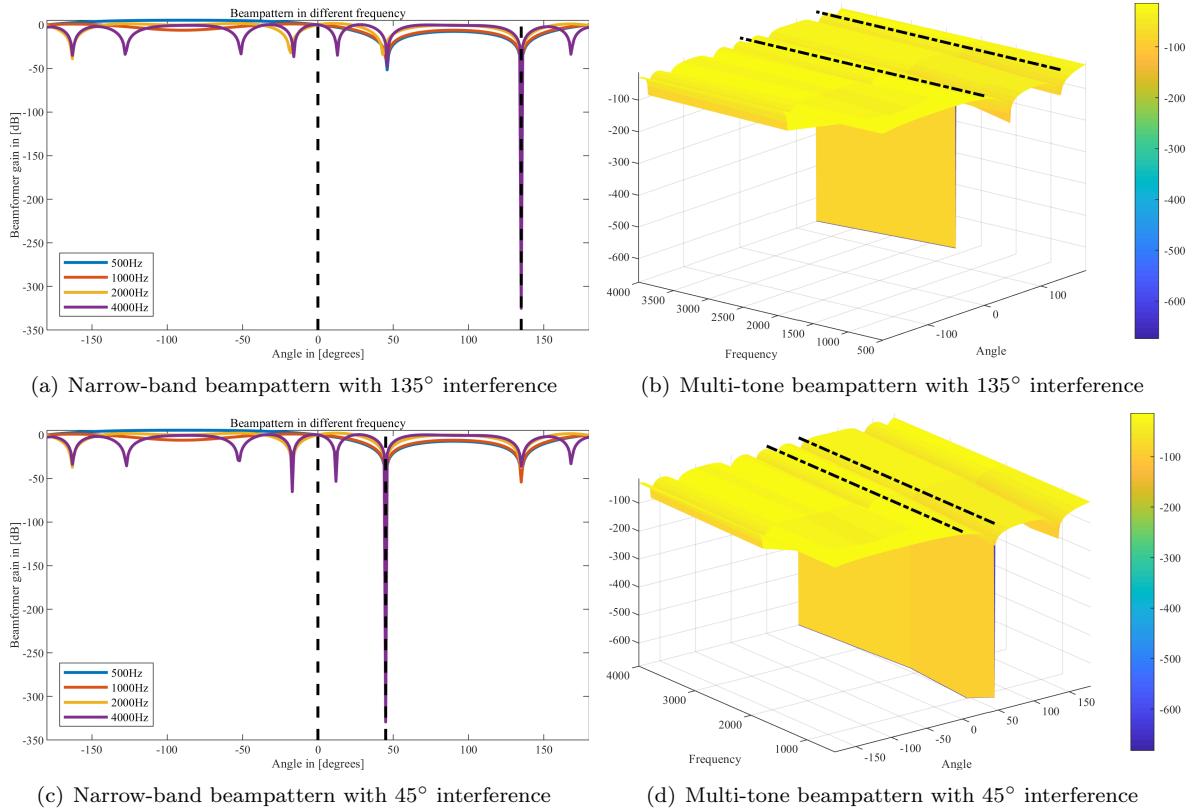


Figure 13: Null-steering beampattern using all four sensors under different interferences and the same  $0^\circ$  desired source

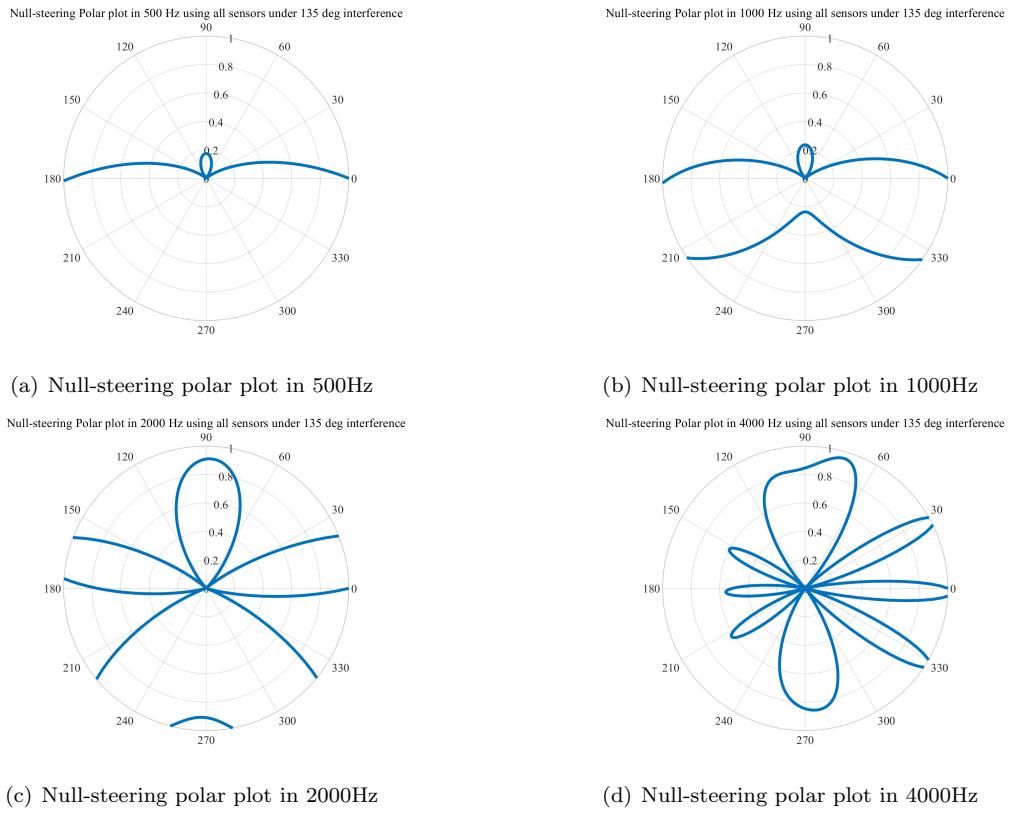


Figure 14: Null-steering polar plots in different frequencies using all four sensors under  $135^\circ$  interference and  $0^\circ$  desired source

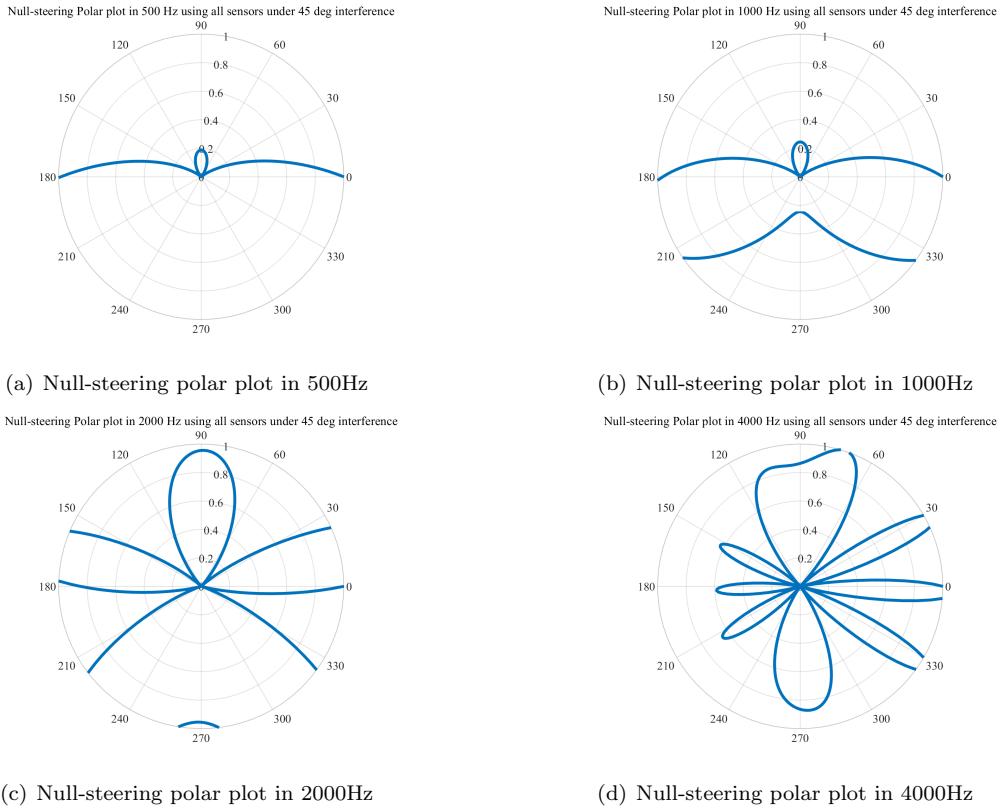


Figure 15: Null-steering polar plots in different frequencies using all four sensors under  $45^\circ$  interference and  $0^\circ$  desired source

### Assignment B: scenario 1: Fractional delay

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

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

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  $\tau$  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  $\tau$ . 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  $\tau$  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.

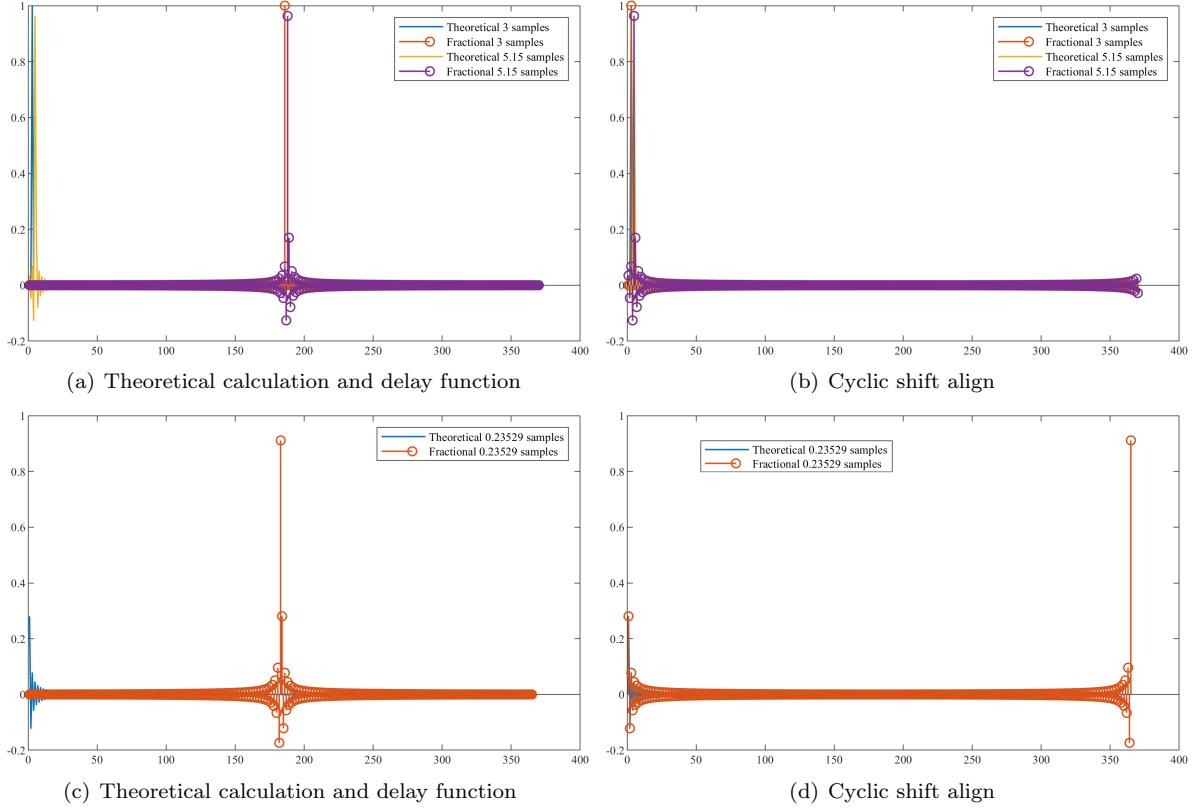


Figure 16: Fractional delay of different  $\tau$

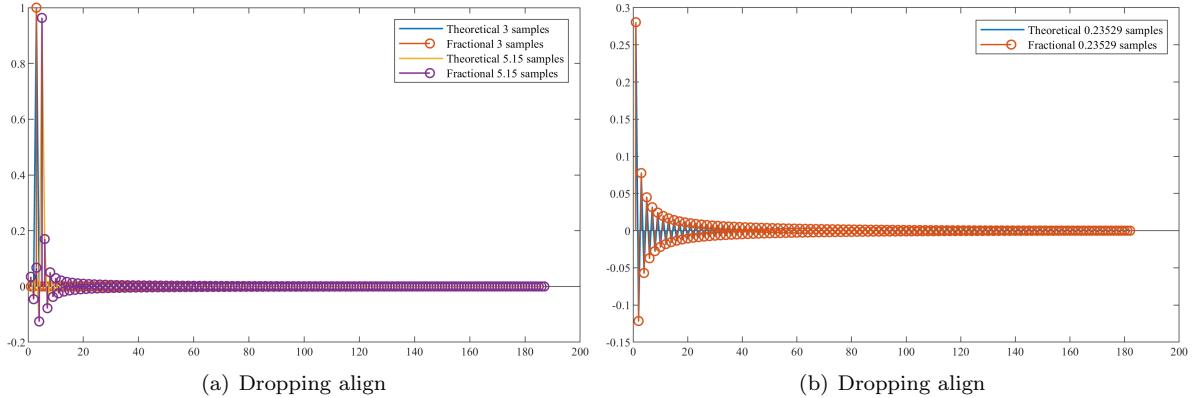


Figure 17: Fractional delay of different  $\tau$  by dropping the first 183 samples

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

- 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.
- The blocking matrix  $\mathbf{B} = [-1, 1]$  or  $\mathbf{B} = [1, -1]$  after time aligning for  $0^\circ$  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.
- 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:

$$\begin{aligned} y &= \frac{1}{2}(x_1 * h + x_2) * h_\Delta - (x_1 * h - x_2) * w \\ \mathbf{Y} &= \frac{1}{2}(\mathbf{X}_1\mathbf{H} + \mathbf{X}_2)\mathbf{H}_\Delta - (\mathbf{X}_1\mathbf{H} - \mathbf{X}_2)\mathbf{W} \\ &= \left(\frac{1}{2}\mathbf{H}\mathbf{H}_\Delta - \mathbf{H}\mathbf{W}\right)\mathbf{X}_1 + \left(\frac{1}{2}\mathbf{H}_\Delta + \mathbf{W}\right)\mathbf{X}_2 \\ &= \left(\frac{1}{2}e^{-j\tau\theta}e^{-j\theta} - e^{-j\tau\theta}\mathbf{W}\right)\mathbf{X}_1 + \left(\frac{1}{2}e^{-j\theta} + \mathbf{W}\right)\mathbf{X}_2 \end{aligned}$$

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.

### 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 :

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

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  $\mathbf{Y}$ :

$$\begin{aligned} \mathbf{Y} &= \frac{1}{4}(\mathbf{X}_1\mathbf{H} + \mathbf{X}_2 + \mathbf{X}_3 + \mathbf{X}_4\mathbf{H})\mathbf{H}_\Delta - \frac{1}{3}[(\mathbf{X}_1\mathbf{H} - \mathbf{X}_2)\mathbf{W}_1 + (\mathbf{X}_1\mathbf{H} - \mathbf{X}_3)\mathbf{W}_2 + (\mathbf{X}_1\mathbf{H} - \mathbf{X}_4\mathbf{H})\mathbf{W}_3] \\ &= \left[\frac{1}{4}\mathbf{H}\mathbf{H}_\Delta - \frac{1}{3}\mathbf{H}(\mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3)\right]\mathbf{X}_1 + \left(\frac{1}{4}\mathbf{H}_\Delta + \frac{1}{3}\mathbf{W}_1\right)\mathbf{X}_2 \\ &\quad + \left(\frac{1}{4}\mathbf{H}_\Delta + \frac{1}{3}\mathbf{W}_2\right)\mathbf{X}_3 + \left(\frac{1}{4}\mathbf{H}\mathbf{H}_\Delta + \frac{1}{3}\mathbf{H}\mathbf{W}_3\right)\mathbf{X}_4 \end{aligned}$$

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

$$\begin{aligned} \frac{\mathbf{Y}}{\mathbf{X}_1} &= \frac{1}{4}\mathbf{H}\mathbf{H}_\Delta - \frac{1}{3}\mathbf{H}(\mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3) = \frac{1}{4}e^{-j(\tau+1)\theta} - \frac{1}{3}e^{-j\tau\theta}(\mathbf{W}_1 + \mathbf{W}_2 + \mathbf{W}_3) \\ \frac{\mathbf{Y}}{\mathbf{X}_2} &= \frac{1}{4}\mathbf{H}_\Delta + \frac{1}{3}\mathbf{W}_1 = \frac{1}{4}e^{-j\theta} + \frac{1}{3}\mathbf{W}_1 \\ \frac{\mathbf{Y}}{\mathbf{X}_3} &= \frac{1}{4}\mathbf{H}_\Delta + \frac{1}{3}\mathbf{W}_2 = \frac{1}{4}e^{-j\theta} + \frac{1}{3}\mathbf{W}_2 \\ \frac{\mathbf{Y}}{\mathbf{X}_4} &= \frac{1}{4}\mathbf{H}\mathbf{H}_\Delta + \frac{1}{3}\mathbf{H}\mathbf{W}_3 = \frac{1}{4}e^{-j(\tau+1)\theta} - \frac{1}{3}e^{-j\tau\theta}\mathbf{W}_3 \end{aligned}$$

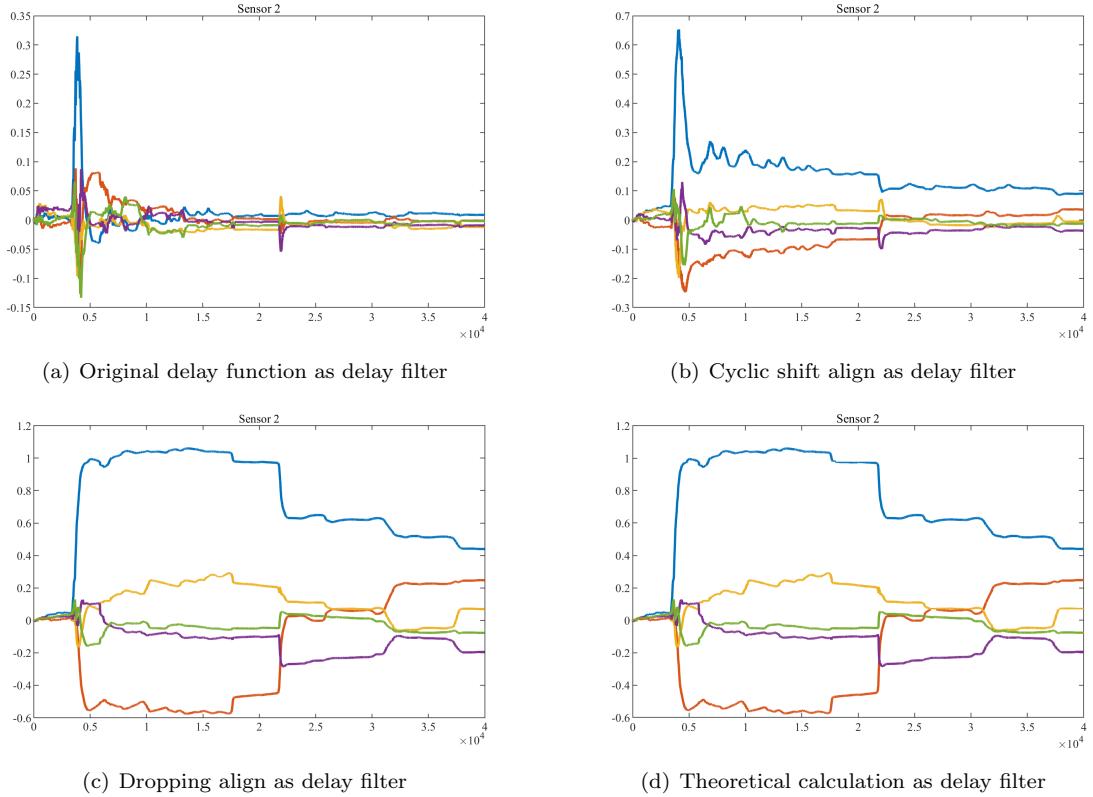


Figure 18: RLS adaptive filter coefficient using different delay filter under left two sensors

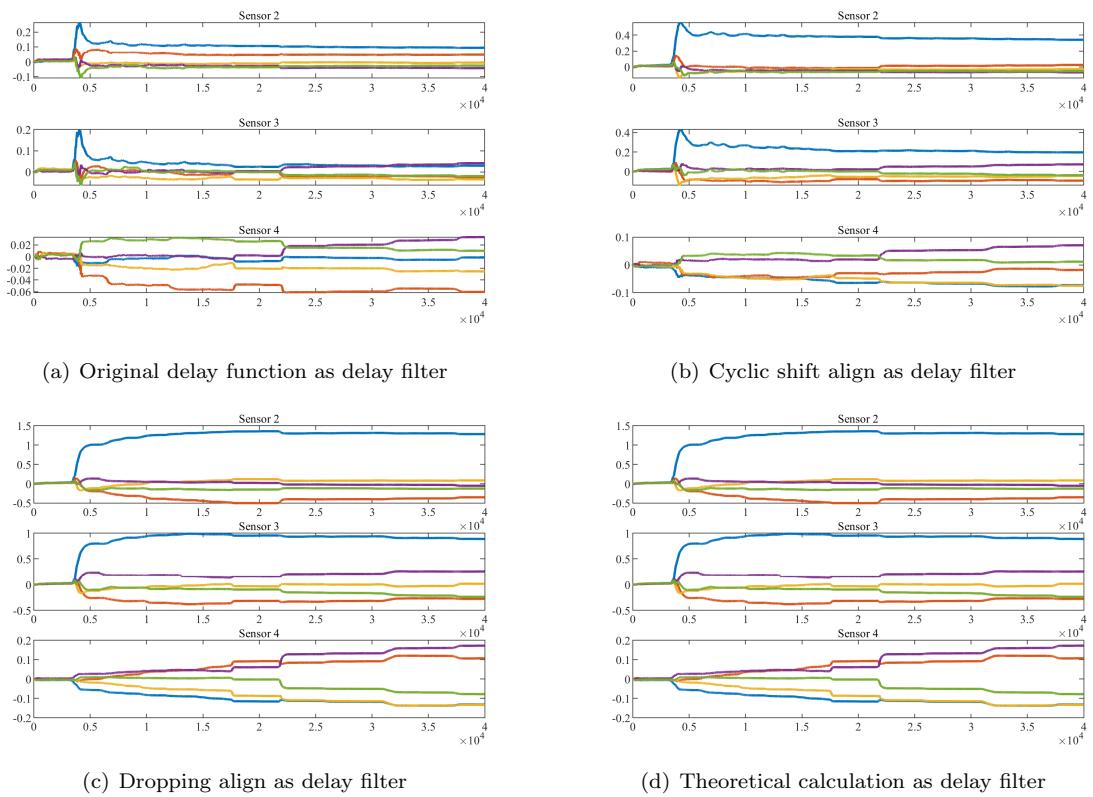


Figure 19: RLS adaptive filter coefficient using different delay filter under all four sensors

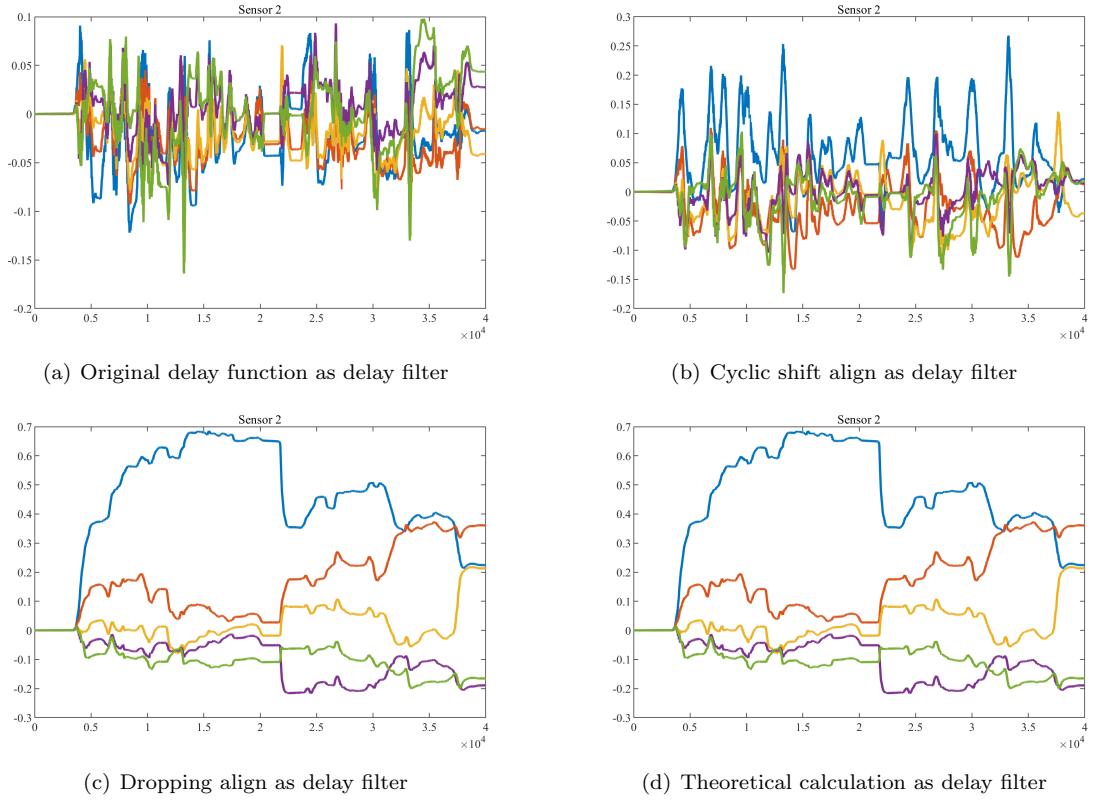


Figure 20: NLMS adaptive filter coefficient using different delay filter under left two sensors

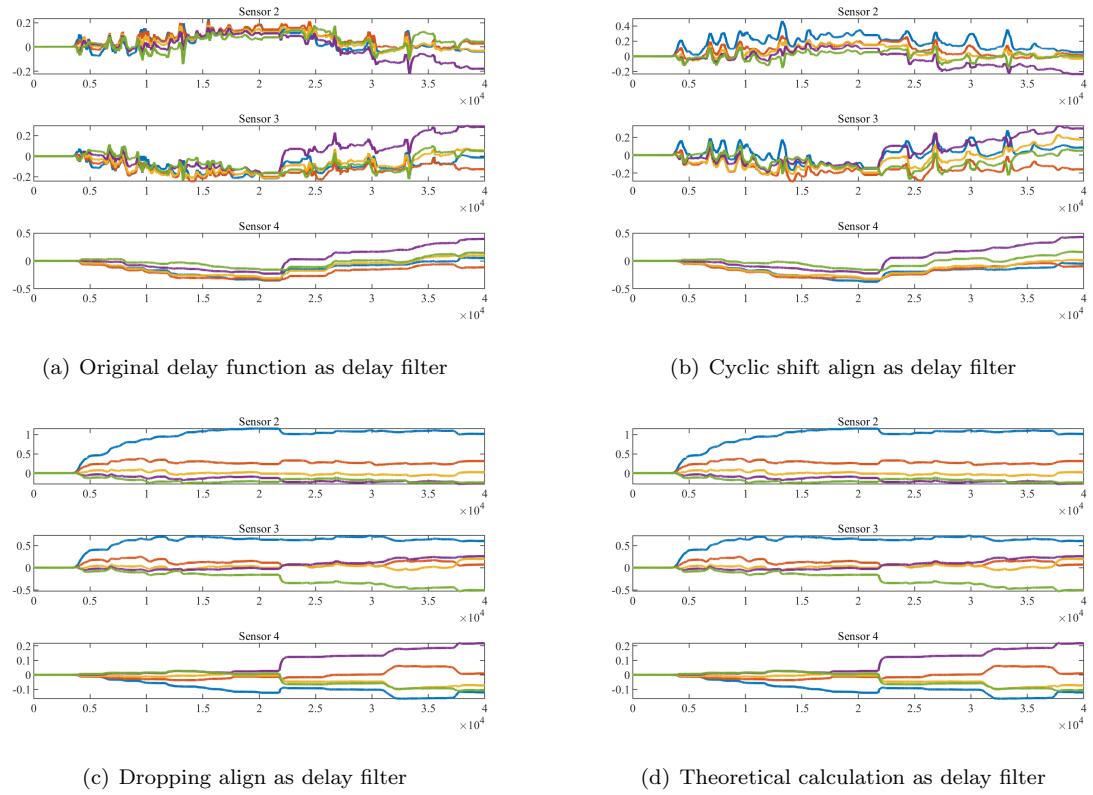


Figure 21: NLMS adaptive filter coefficient using different delay filter under all four sensors

## 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.

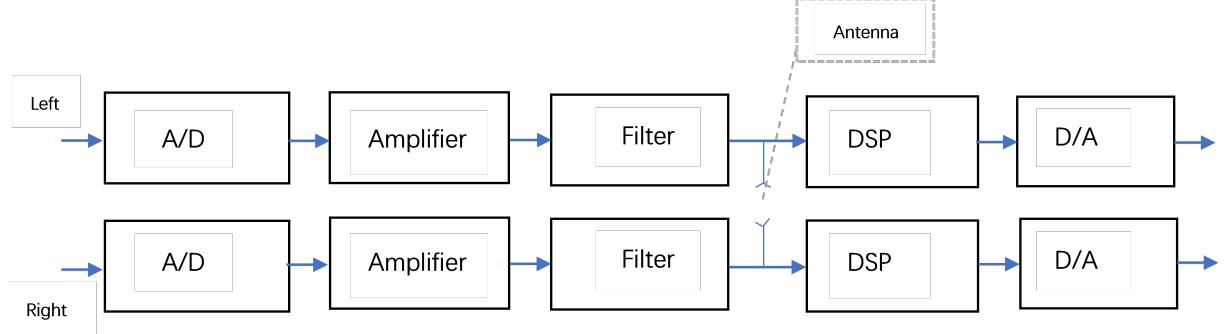


Figure 22: Data exchange flow diagram

The MATLAB codes is attached below:

```

1  clear all;
2  close all;
3  clear classes;
4  clc;
5  %% Array Configuration
6  Scenario = 'all';
7  switch Scenario
8      case 'left'
9          dx = 0;
10         dy = 0.01;
11         J = 2;
12         my_array = arrays.ULA(J,dx,dy);
13         disp('The left two microphones are ready !');
14         figure(1);
15         my_array.plot();
16         flag = 0;
17     case 'upper'
18         dx = 0.17;
19         dy = 0;
20         J = 2;
21         my_array = arrays.ULA(J,dx,dy);
22         disp('The upper two microphones are ready !');
23         figure(1);
24         my_array.plot();
25         flag = 2;
26     case 'all'
27         dx = 0.17;
28         dy = 0.01;
29         J = 4;
30         my_array = arrays.square_array(J,dx,dy);
31         disp('All four microphones are ready !');
32         figure(1);
33         my_array.plot();
34         flag = 1;
35 end
36
37
38 %% Beamformer
39 mt_f = [500 1000 2000 4000];           % multiple tones in Hz
40 c = 340;                                % the sound speed
41 theta = [0,45];                          % row vector containing angles for which ...
42 constraints hold
43 target = [1,0];                           % row vector containing target values for the ...
44 beampattern
45
46 b = beamformer;
47 set(b, 'array', my_array);

```

```

47 set(b, 'angles', -180:1:180);
48 set(b, 'n_sensor', J);
49 set(b, 'sound_speed', c);
50 set(b, 'mt_frequency', mt_f);
51
52 %% Pattern with Weights 1/J
53 figure(2)
54 for i = 1:length(mt_f)
55 nb_f = mt_f(i);
56 set(b, 'nb_frequency', nb_f);
57 b.nb_weights = 1/J*ones(J,1);
58 b.calc_nb.beampattern;
59 Beam = b.nb.beampattern;
60 plot(b.angles, 10*log10(Beam), 'LineWidth', 2);
61 %b.plot_nb;
62 hold on;
63 end
64 legend('500Hz', '1000Hz', '2000Hz', '4000Hz', 'Location', 'southwest')
65 axis tight
66 title('Beampattern in different frequency')
67 xlabel('Angle in [degrees]')
68 ylabel('Beamformer gain in [dB]');
69
70 b.mt_weights = 1/J*ones(J,length(mt_f));
71 b.calc_mt.beampattern;
72 figure(3)
73 b.plot_mt;
74
75 for i = 1:length(mt_f)
76 nb_f = mt_f(i);
77 set(b, 'nb_frequency', nb_f);
78 b.nb_weights = 1/J*ones(J,1);
79 b.calc_nb.beampattern;
80 Beam = b.nb.beampattern;
81 figure(100+i)
82 polarplot(b.angles.*pi/180, Beam, 'LineWidth', 2);
83 rlim([0 1])
84 title(['Polar plot in ', num2str(mt_f(i)), ' Hz using ', Scenario, ' sensors']);
85 end
86
87
88 %% Null-steering pattern
89 figure(4)
90 for i = 1:length(mt_f)
91 nb_f = mt_f(i);
92 set(b, 'nb_frequency', nb_f);
93 b.beam_steering.nb(theta, target);
94 b.calc_nb.beampattern;
95 Beam = b.nb.beampattern;
96 plot(b.angles, 10*log10(Beam), 'LineWidth', 2);
97 hold on;
98 end
99 plot([theta; theta], [-350 max(10*log(Beam))]* ones(1, length(theta)), 'k--', 'LineWidth', 2);
100 legend('500Hz', '1000Hz', '2000Hz', '4000Hz', 'Location', 'southwest')
101 axis tight
102 title('Beampattern in different frequency')
103 xlabel('Angle in [degrees]')
104 ylabel('Beamformer gain in [dB]');
105
106
107 b.beam_steering.mt(theta, target);
108 b.calc_mt.beampattern;
109 figure(5);
110 b.plot_mt(theta, {'k-.', 'LineWidth', 2});
111
112 for i = 1:length(mt_f)
113 nb_f = mt_f(i);
114 set(b, 'nb_frequency', nb_f);
115 b.beam_steering.nb(theta, target);
116 b.calc_nb.beampattern;
117 Beam = b.nb.beampattern;
118 figure(200+i)
119 polarplot(b.angles.*pi/180, Beam, 'LineWidth', 2);
120 rlim([0 1])
121 title(['Null-steering Polar plot in ', num2str(mt_f(i)), ' Hz using ', Scenario, ' ...
    sensors under ', num2str(theta(2)), ' deg interference']);
122 end

```

```

123
124
125 %% Fractional Delay
126 if flag ≠ 2
127 Dataset = load('DatasetAssignBs2.mat');
128 fs = Dataset.fs;
129 time_align = dy/c * fs;
130 tau = time_align;
131 %tau = [3,5.15];
132 %tau = [15.551,42.443];
133 disp(['Time align = ', num2str(tau), ' samples !']);
134
135 legend_the = cell(1,length(tau));
136 legend_fra = cell(1,length(tau));
137 for i = 1:length(tau)
138 [h,my_h] = Fractional_delay(tau(i));
139 my_h(isnan(my_h)) = 1;
140 legend_the{i} = ['Theoretical ' num2str(tau(i)) ' samples'];
141 legend_fra{i} = ['Fractional ' num2str(tau(i)) ' samples'];
142 figure(50)
143 plot(my_h,'LineWidth',1)
144 hold on;
145 stem(h,'LineWidth',1)
146 hold on;
147 figure(51)
148 plot(my_h(1:end-183),'LineWidth',1)
149 hold on;
150 %stem(circshift(h,length(h)-183),'LineWidth',1)
151 stem(h(184:end),'LineWidth',1)
152 hold on;
153 end
154 %legend({legend_the});
155 if length(tau) == 1
156 figure(50)
157 legend([legend_the, legend_fra]);
158 figure(51)
159 legend([legend_the, legend_fra]);
160 else
161 figure(50)
162 legend(legend_the{1}, legend_fra{1}, legend_the{length(tau)}, legend_fra{length(tau)});
163 figure(51)
164 legend(legend_the{1}, legend_fra{1}, legend_the{length(tau)}, legend_fra{length(tau)});
165 end
166 h = circshift(h,length(h)-183);
167 %h = h(184:end);
168
169 %% Time Align
170 x = Dataset.x(1:J,:);
171 u = x;
172 u(1,:) = filter(h,1,x(1,:).' );
173 %u(1,:) = filter(my_h,1,x(1,:).' );
174 if flag == 1
175 u(J,:) = filter(h,1,x(J,:).' );
176 %u(J,:) = filter(my_h,1,x(J,:).' );
177 end
178 v0 = 1/J * sum(u);
179 d = v0;
180 d(:,1:end-1) = v0(:,2:end);
181 d(:,end) = v0(:,1); % d(:,end) = 0;
182
183 %d(:,1:end-183) = v0(:,184:end);
184 %d(:,end-182:end) = v0(:,1:183);
185
186
187 %v0 = (v0 - sum(v0)./length(v0))./std(v0);
188
189 soundsc(v0,fs);
190 disp('Listen, the upper branch audio is playing !');
191
192 %audiowrite('GSCupper_branch.wav',v0,fs);
193 %v = u(1,:) - u(J,:);
194 %soundsc(v,fs);
195 %B = [-1 1];
196 B(:,1) = -1*ones(J-1,1);
197 B(:,2:J) = eye(J-1);
198 v = B*u;
199

```

```

200 %% Adaptive
201 %w_plot = [];
202 state = Init_Als2(J);
203 w_plot = zeros(state.nw,J-1,length(d));
204 for i = 1:length(d)
205     state = Update_Als2(state,d(i),v(:,i).');
206     %w_plot = [w_plot,state.w];
207     w_plot(:,:,i) = state.w;
208 end
209 w_a = state.w;
210 g_temp = zeros(J-1,length(d));
211 for i = 1:J-1
212     g_temp(i,:) = filter(w_a(:,i),1,v(i,:));
213 end
214 %g = filter(w_a,1,v);
215 g = 1/(J-1) * sum(g_temp,1);
216 y = d - g;
217
218 %soundsc(y,fs);
219 disp('Listen,the output audio is playing !');
220
221 figure
222 %for i = 1:state.nw
223 %    plot(w_plot(i,:));
224 %    hold on;
225 %end
226 for i = 1:J-1
227 subplot(J-1,1,i);
228     for j = 1:state.nw
229         plot(squeeze(w_plot(j,i,:)), 'LineWidth', 1.5);
230         hold on;
231     end
232 title(['Sensor ' num2str(i+1)]);
233 end
234
235 else
236     disp('Warning!! There is no delay!');
237 end
238
239 function [h,my_h] = Fractional_delay(tau)
240 dmax = ceil(tau);
241 d_int = fix(tau);
242 index_dot = find(num2str(tau)== '.');
243 if isempty(index_dot)
244     L = 10;
245     fd = 0;
246 else
247     L = 10^(length(num2str(tau)) - index_dot);
248     fd = fix((tau - d_int)*L);
249 end
250 h = delay(d_int,fd,L,dmax);
251 %H = 10*log10(abs(fft(h)));
252 %% Analytical expression
253 len_filter = length(h);
254 %half = fix((len_filter-1)/2);
255 %n = linspace(-half,half,len_filter);
256 n = 1:len_filter;
257 %n = 1:len_filter-183;
258 my_h = sin(pi*(n - tau))./(pi*(n - tau));
259 end
260
261 function [ state ] = Init_Als2(J)
262 % Initialize all variables that you want to use:
263 deta = 0.1;           %0.1
264 state.J = J;
265 state.alpha = 0.00008; %0.0008
266 state.nw = 5;
267 state.var = 0.5*ones(1,state.J-1);
268 state.w = zeros(state.nw,state.J-1);
269 state.x = zeros(state.nw,state.J-1);
270 state.R_inv = deta * eye(state.nw);
271 end
272
273 function state = Update_Als2( state, x1, x2 )
274 alpha = state.alpha;
275 e = x1;
276 state.x(2:end,:) = state.x(1:end-1,:);      % this is a vector

```

```

277 state.x(1,:) = x2;
278 %r = e - state.w.' * state.x;
279 r = e - (1/(state.J-1)) * sum(diag(state.w.' * state.x));
280
281 %% RLS
282 function rls
283 lamda = 0.9999999;           %deta = 0.1
284 %lamda = 1;
285 gain = (state.R_inv * state.x)./(lamda^2 + diag(state.x.' * state.R_inv * state.x)).';
286 state.R_inv = (state.R_inv - gain * state.x.' * state.R_inv)./(lamda^2);
287 state.w = state.w + r * gain;
288 end
289
290 %% NLMS
291 function nlms
292 beta = 0.9999999;  %0.5
293 %beta = 1;
294 state.var = beta*state.var + (1-beta)*((diag(state.x.*state.x)).'/state.nw);
295 state.w = state.w + 2*alpha*r*state.x./state.var;
296 end
297 end

```