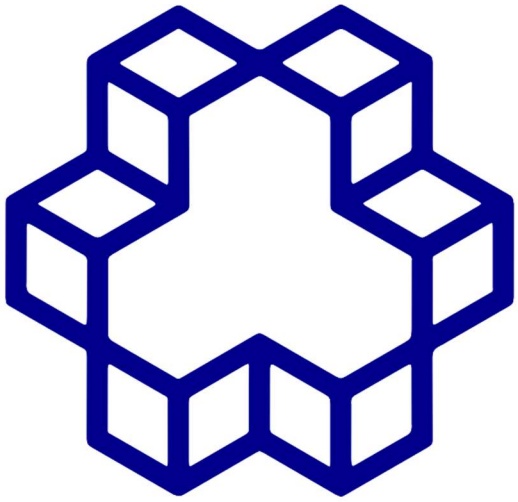


دانشگاه صنعتی خواجه نصیرالدین طوسی

گزارشکار تمرین کامپیوتری سری دوم

درس پردازش سیگنال‌های دیجیتال

آنائیس گل‌بوداغیانس ۴۰۱۲۲۱۱۳

سوال اول

بخش اول

در ابتدای هر کدی از چهار دستور زیر برای بستن پنجره و شکل‌ها، تمیز کردن Command window و پاک کردن متغیرها استفاده می‌کنیم.

```
%2nd computer asignment, DSP
%Anaies Golboudaghians 40122113
%Q1
%% Part 1
clc; clear; close all; close all hidden
```

در این سوال، تابعی نوشته شده تا از تکرار کدها پرهیز شود.

```
function FinalFilter(M,w)
    wvtool(w);
    figure
    n = 0:M-1;
    h_d = sinc(n);
    h = h_d.*w';
    H = fft(h);
    subplot(1,2,1)
    plot(1/M*(0:M-1),abs(H));
    xlabel('\omega');
    ylabel('|H(e^{j\omega})|')
    hold on
    subplot(1,2,2)
    plot(1/M*(0:M-1),angle(H));
    xlabel('\omega');
    ylabel('arg(H(e^{j\omega}))')
end
```

این تابع، فیلتر نهایی را ارائه می‌کند. ابتدا پنجره مورد نظر را می‌گیرد و آن را در حوزه زمان و فرکانس با استفاده از تابع آماده wvtool رسم می‌کند. طول فیلتر نیز از ورودی‌های این تابع است. می‌خواهیم پنجره مورد نظر را در تابع سینک ضرب کنیم پس نیاز داریم طول پنجره و بردار سینک یکی باشد.

برای از بین نرفتن شکل‌ها، از دستور figure استفاده می‌کنیم تا پنجره جدید باز شود.

نیاز داریم بردار پنجره را ترانهاده کنیم w’ زیرا بردار پنجره ساخته شده آرایه‌ای عمودی است اما مقادیر سینک تولید شده آرایه‌ای افقی می‌باشد. اگر بدون ترنسپوز یا ترانهاده کردن آن‌ها را ضرب کنیم، ماتریسی M*M تشکیل می‌شد که مورد نظرمان نبود.

از مقادیر ضرب شده سینک در پنجره تبدیل فوریه می‌گیریم و فاز و اندازه آن را رسم می‌کنیم.

این تابعی بود که در چندبار استفاده شده بود.

حال به هدف اصلی سوال برمی‌گردیم: انتخاب پنجره ایده‌آل.

با توجه به پارامترهای داده‌شده، پنجره مستطیلی و bartlett و hann، حداقل Stopband Attenuation مورد نظر را دارند. اگر پنجره‌های دیگری انتخاب کنیم، سیگنال بیشتر از مقدار خواسته‌شده در صورت سوال تضعیف خواهد شد.

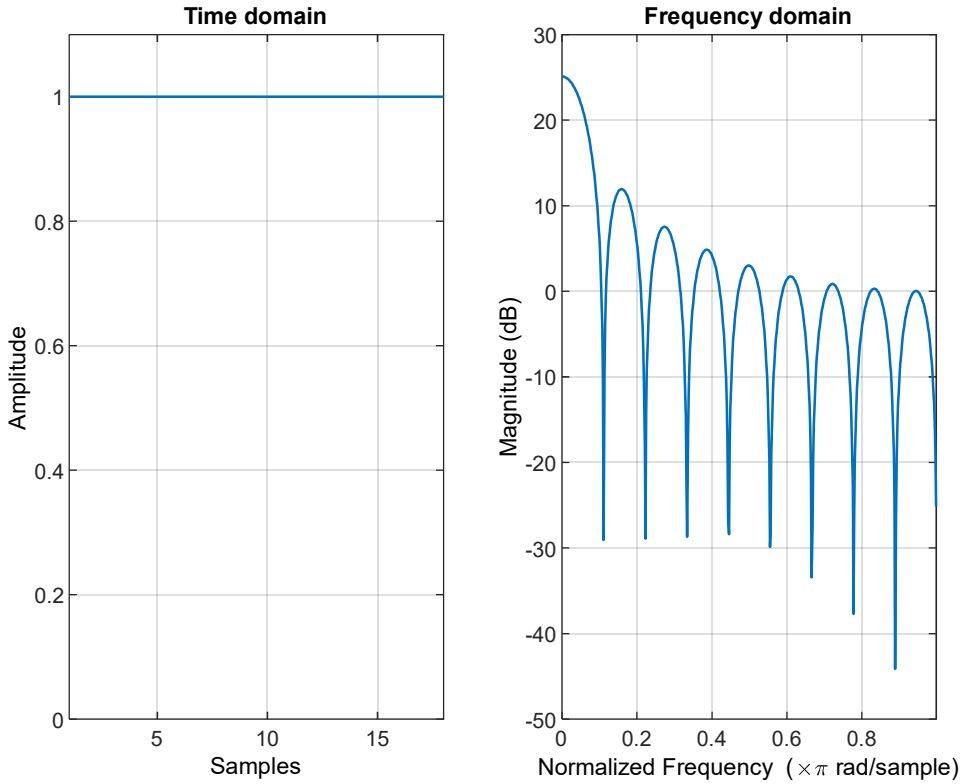
وقتی هرسه‌ی این‌ها را رسم کردیم، دامنه‌ی ریپل‌شان کم‌تر از مقدار گفته‌شده در صورت سوال بود.

```
R_p = 0.25;
A_s = 50;

w_p = 0.2*pi;
w_s = 0.3*pi;
dw = w_s - w_p;

M = round(1.8*pi/dw);
w1 = rectwin(M);
FinalFilter(M,w1);

M = round(6.1*pi/dw);
w2 = bartlett(M);
FinalFilter(M,w2);

M = round(6.2*pi/dw);
w3 = hann(M);
FinalFilter(M,w3);
```

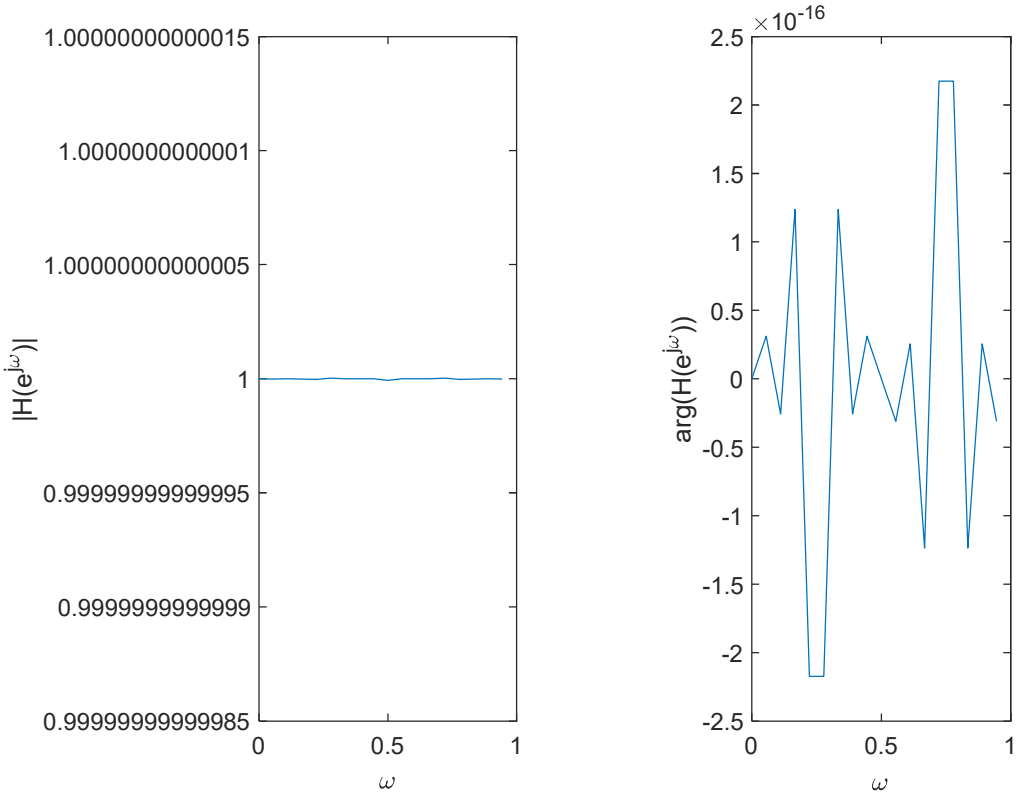
طول فیلتر را با توجه به باند گذر فیلتر dwتعیین می‌کنیم. از تابع round استفاده می‌کنیم تا مقدار به‌دست آمده صحیح باشد. هرچه طول بیشتر باشد، فیلتر نیز ایده‌آل‌تر خواهد بود.

خروجی:

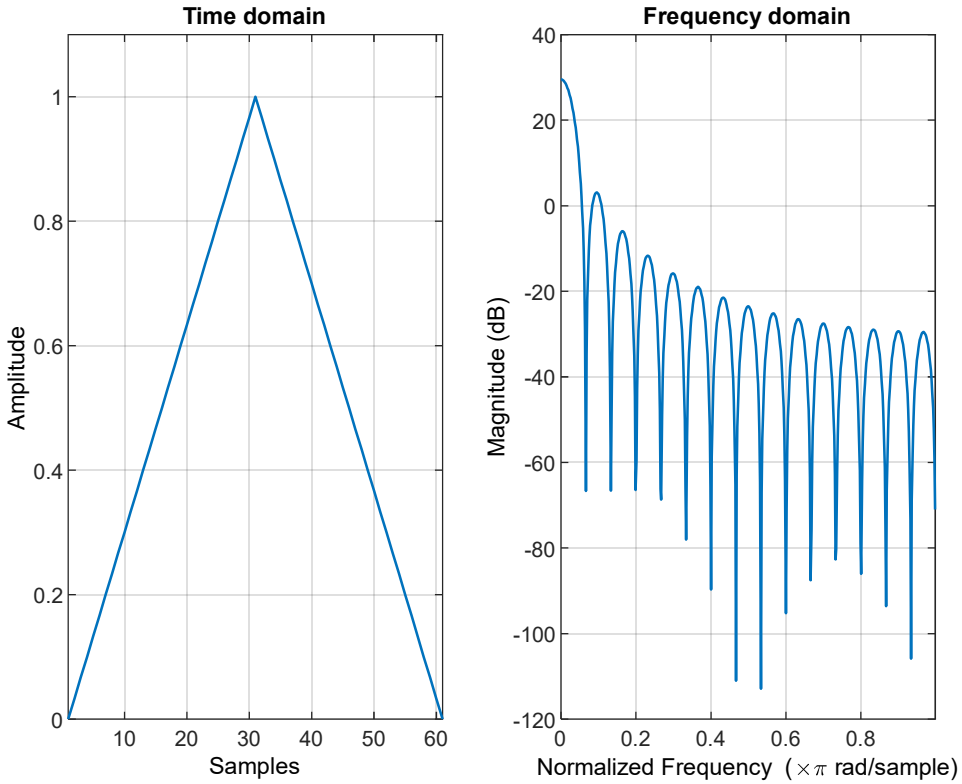
پنجره مستطیلی



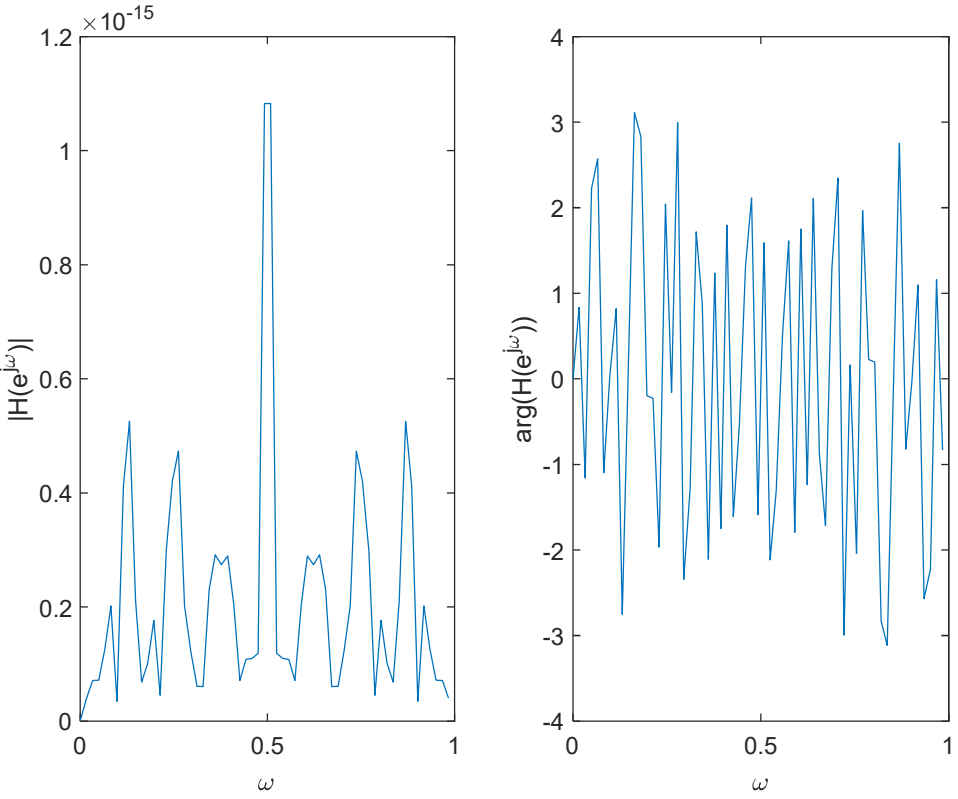
فیلتر طراحی‌شده:



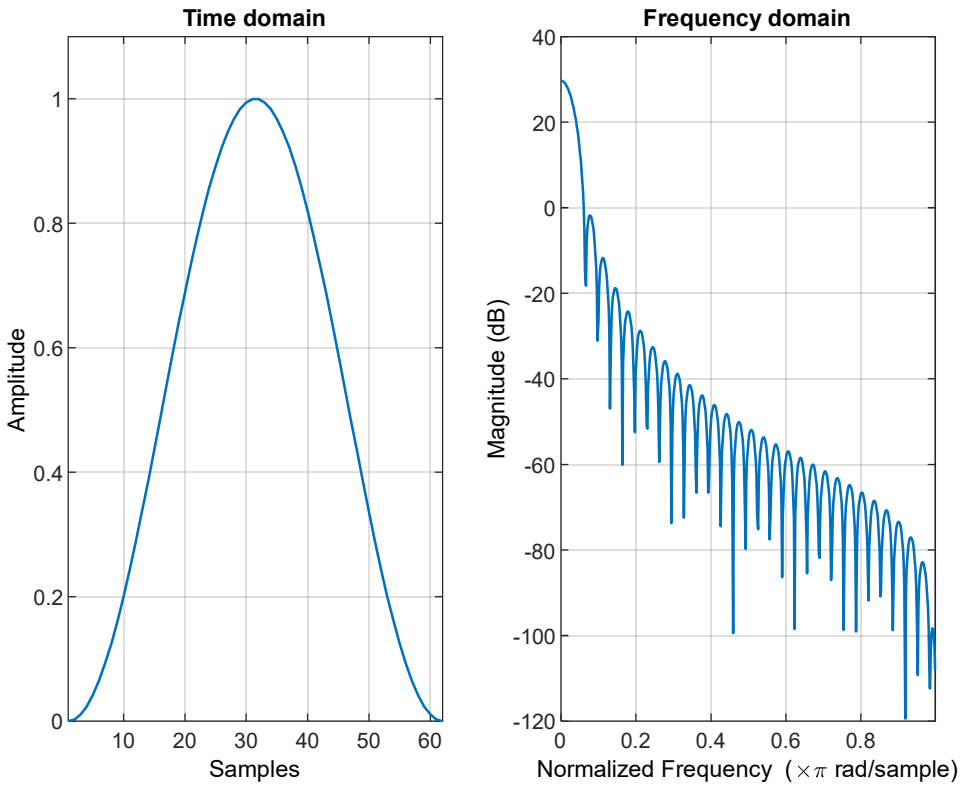
پنجره بارتلت



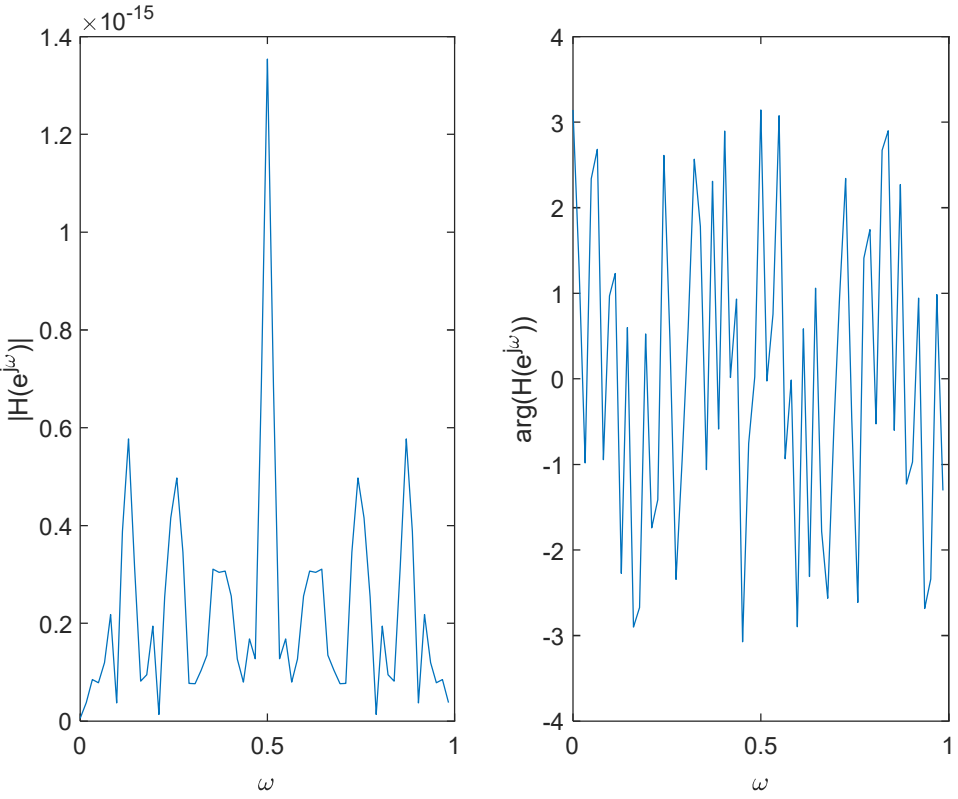
فیلتر نهایی:



پنجره هان:


Time domain
Amplitude
Samples
0
0.2
0.4
0.6
0.8
1
10
20
30
40
50
60
Frequency domain
Magnitude (dB)
Normalized Frequency (×π rad/sample)
40
20
0
-20
-40
-60
-80
-100
-120
0
0.2
0.4
0.6
0.8

فیلتر نهایی:



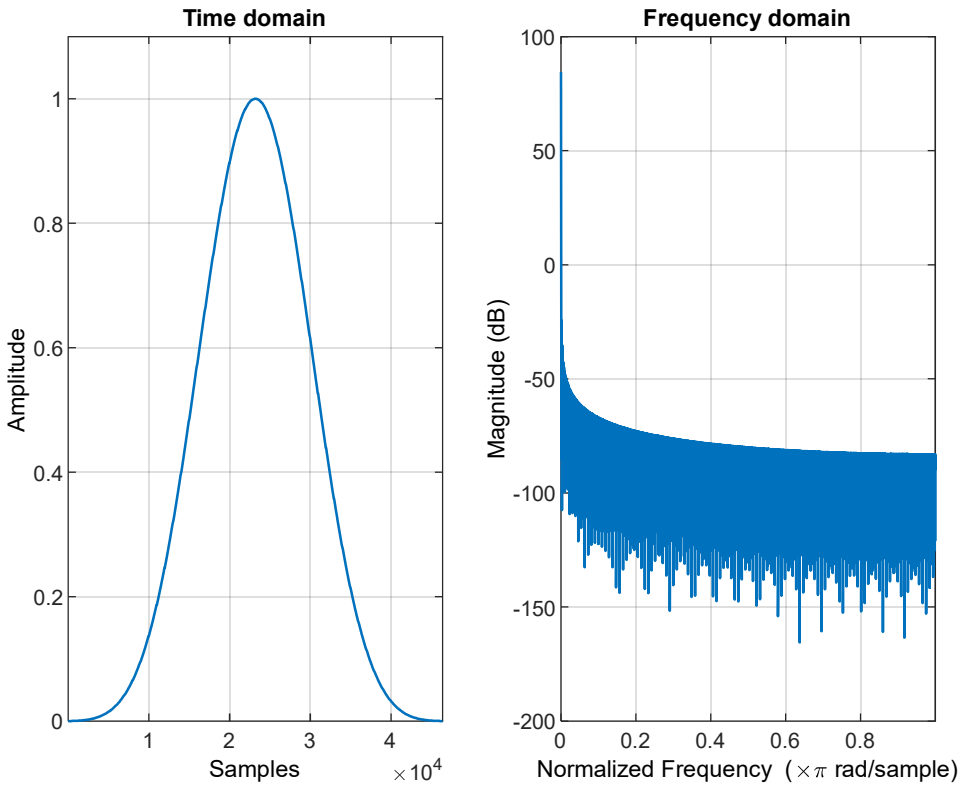
بخش دوم

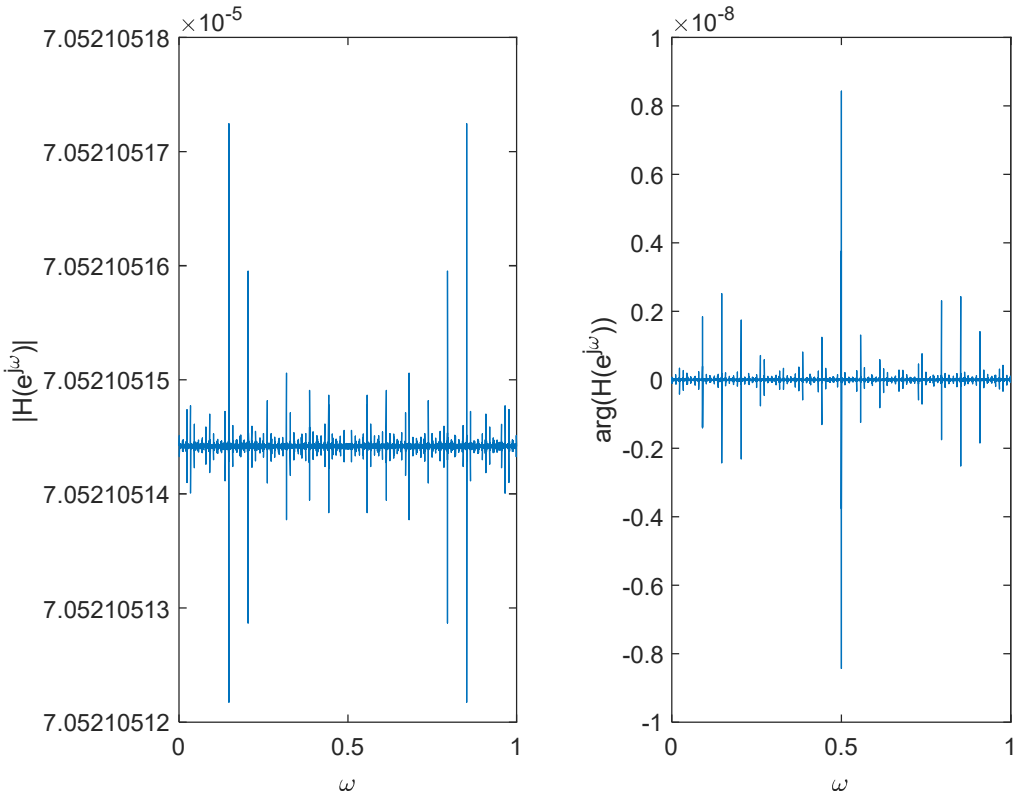
اکنون باید پنجره Kaiser را طراحی کنیم.

```
%% Part 2
d1 = (1-10^(-R_p/20))/(1+10^(-R_p/20));
d2 = (1+d1)*(10^(-A_s/20));
d = min(d1,d2);
A = -20*log(d);

M = round((A-8)/(2.285*d*dw));
if A>50
    beta = 0.1102*(A-8.7);
elseif A<21
    beta = 0;
else
    beta = (.5842*(A-21)^0.4)+(.07886*(A-21));
end
w6 = kaiser(M,beta);
FinalFilter(M,w6);
```

پارامترهای مورد نظر این فیلتر را با استفاده از فرمول‌های موجود در مرجع محاسبه می‌کنیم.

خروجی:


Time domain
Amplitude
Samples
×10^4
0
0.2
0.4
0.6
0.8
1
1
2
3
4
Frequency domain
Magnitude (dB)
Normalized Frequency (×π rad/sample)
100
50
0
-50
-100
-150
-200
0
0.2
0.4
0.6
0.8



سوال دوم

بخش اول

در این سوال نیز از تابع زیر استفاده شده. این تابع مقادیر محاسبه شده فوریه و خود سیگنال در حوزه زمان برای به‌دست آوردن طول آن، می‌گیرد و اندازه و فاز آن را رسم می‌کند.

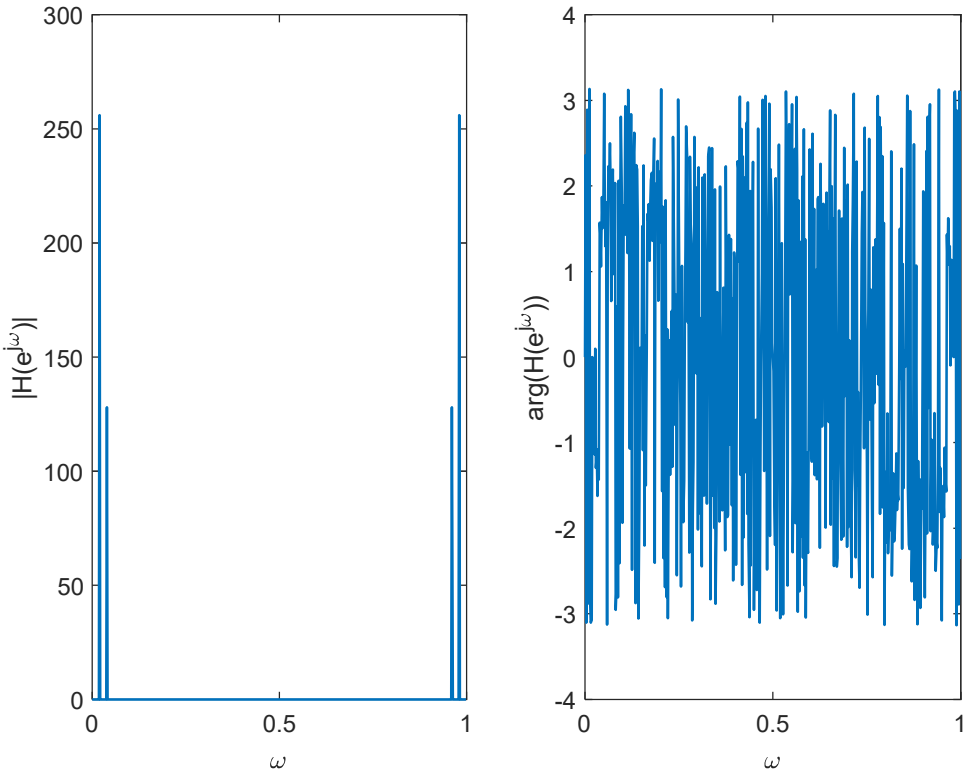
```
function freqPlot(X,x)
    figure
    w_sample=1/length(x)*(0:length(x)-1);
    subplot(1,2,1)
    plot(w_sample,abs(X),"LineWidth",1);
    xlabel('\omega');
    ylabel('|H(e^{j\omega})|')
    hold on
    subplot(1,2,2)
    plot(w_sample,angle(X),"LineWidth",1);
    xlabel('\omega');
    ylabel('arg(H(e^{j\omega}))')
end
```

حال برای سیگنال داده‌شده، تبدیل فوریه آن را رسم می‌کنیم.

```
%2nd computer asignment, DSP
%Anaies Golboudaghians 40122113
```

```
%Q2
%% part 1
clc;clear;close all;close all hidden
t = 0:(1/512):(1-(1/512));
x = cos(2*pi*10.*t) + 0.5*cos((2*pi*20.*t)+(pi/2));
X = fft(x);
freqPlot(X,x);
```

خروجی:



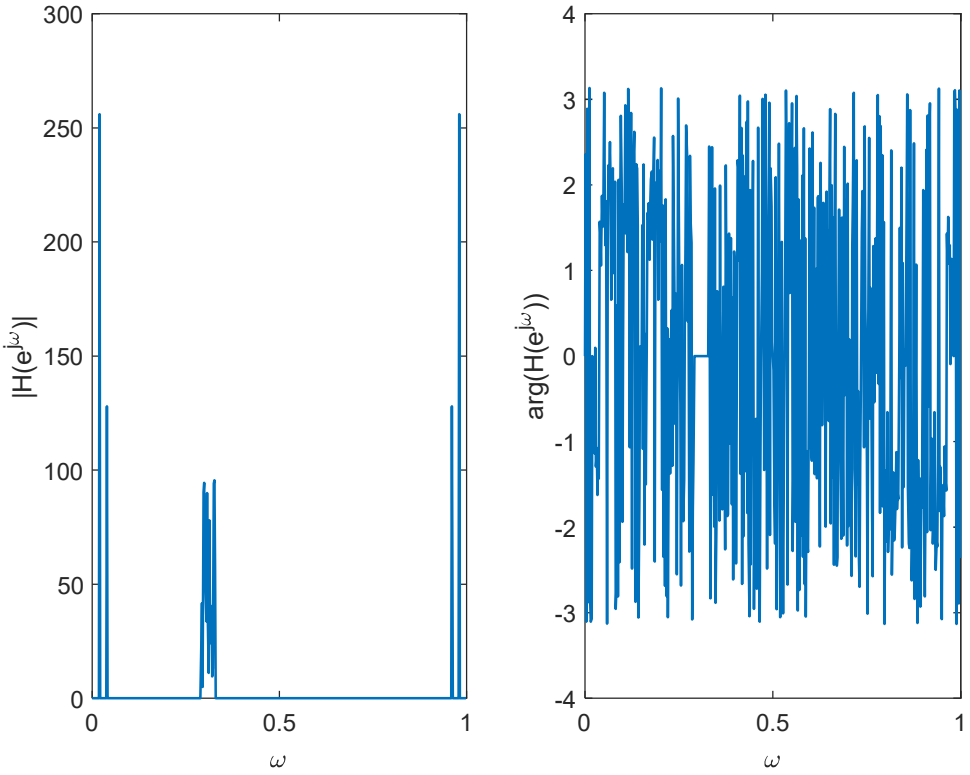
بخش دوم

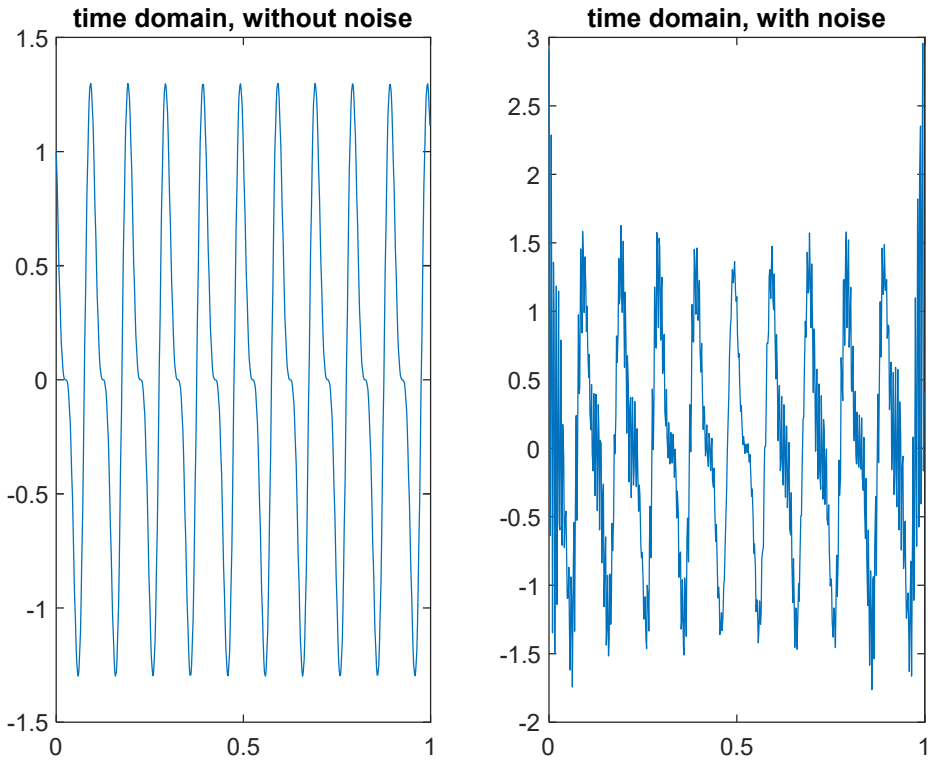
در بازه‌ی گفته‌شده نویز اضافه می‌کنیم. برای نویز، مقدار تصادفی‌ای تولید شد. این مقدار توزیع یکنواخت داشت و از ضریبی نیز استفاده شد تا نویز چشم‌گیرتر باشد.

از دستورات منطقی استفاده شده تا نویز فقط در بازه‌ی ذکر شده قرار بگیرد.

```
%% part 2
f = (0:length(x)-1)*(1/length(x));
f_range = [0.29,0.33];
f_i = f>=f_range(1) & f<=f_range(2);
noise = 100*rand(size(X));
```

```
X(f_i) = X(f_i) + noise(f_i);
freqPlot(X,x);
x_r = ifft(X);
figure
subplot(1,2,1)
plot(t,x);
title("time domain, without noise");
subplot(1,2,2)
plot(t,x_r);
title("time domain, with noise");
```





در حوزه زمان سیگنال را با و بدون نویز مقایسه کردیم.

بخش سوم

نویز اضافه شده در فرکانس‌های میانی بوده؛ پس لازم است که از فیلتر میان‌نگذر چپیشف استفاده کنیم.

```
%% part 3
fs = 1000;
w_p = [(0.2) (0.45)];
[b,a] = cheby1(3,.001,w_p,"stop");
figure
freqz(b,a,[],fs)
[H,W] = freqz(b,a,[],fs);
HH = H';
Y = X.*HH;
freqPlot(Y,x);
y = ifft(Y);
figure
plot(t,y);
title("time domain, after filtering");
```

باند را کمی وسیع‌تر از باند نویز در نظر می‌گیریم تا از فیلتر شدن نویزها اطمینان کافی داشته باشیم.

تابع cheby1 مرتبه فیلتر و دامنه ریپل و باند و نوع فیلتر را گرفته و ضرایب صورت و مخرج پاسخ فرکانسی را برمی‌گرداند.

مرتبه را ۳ در نظر گرفتم. اگر بیشتر می‌شد باند گذر فیلتر بیشتر جابه‌جا می‌شد و اگر مرتبه را ۱ در نظر می‌گرفتم، تضعیف نویز کم‌تری صورت می‌گرفت.

دامنه ریپل را مقدار حداقلی در نظر گرفتم تا ریپل ناخواسته نداشته باشیم. البته ایجاد چنین دامنه‌ای در واقعیت ممکن است چالش‌برانگیز باشد.

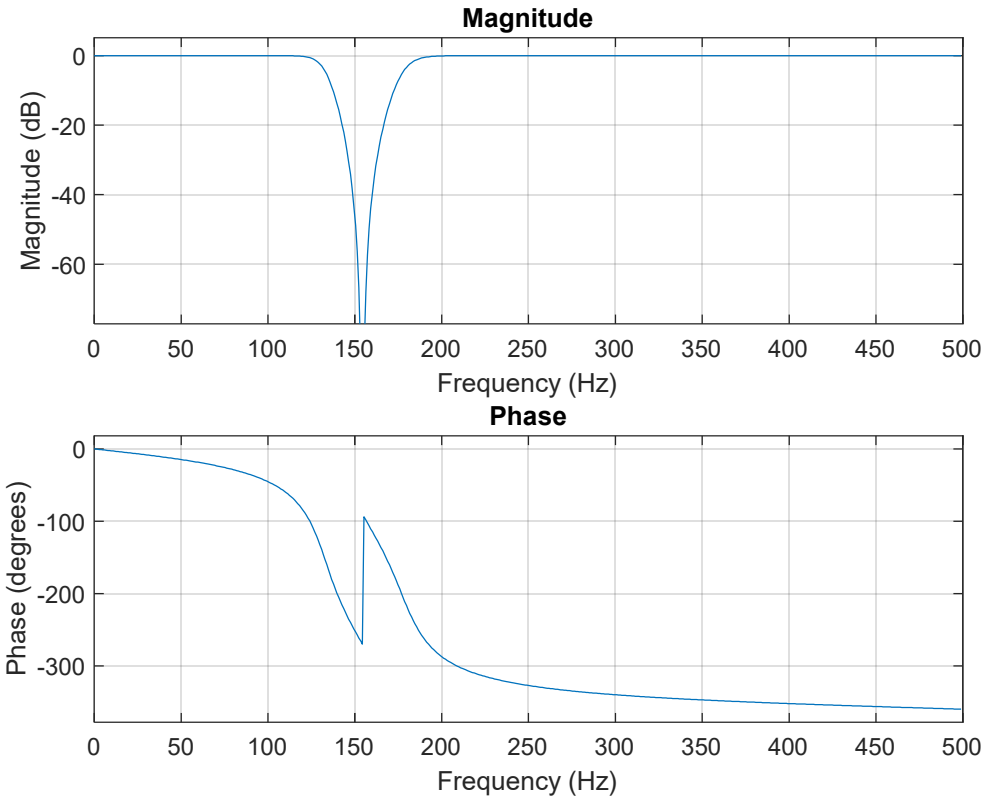
تابع freqz فیلتر را در حوزه فوریه (بهره dB و فاز) رسم می‌کند.

در خط بعدی آن را با بردار h و فرکانس‌های متناظر آن به‌دست آوردیم. باز به‌خاطر مشکلی که قبلا گفته شد آن را ترانهاده کردیم.

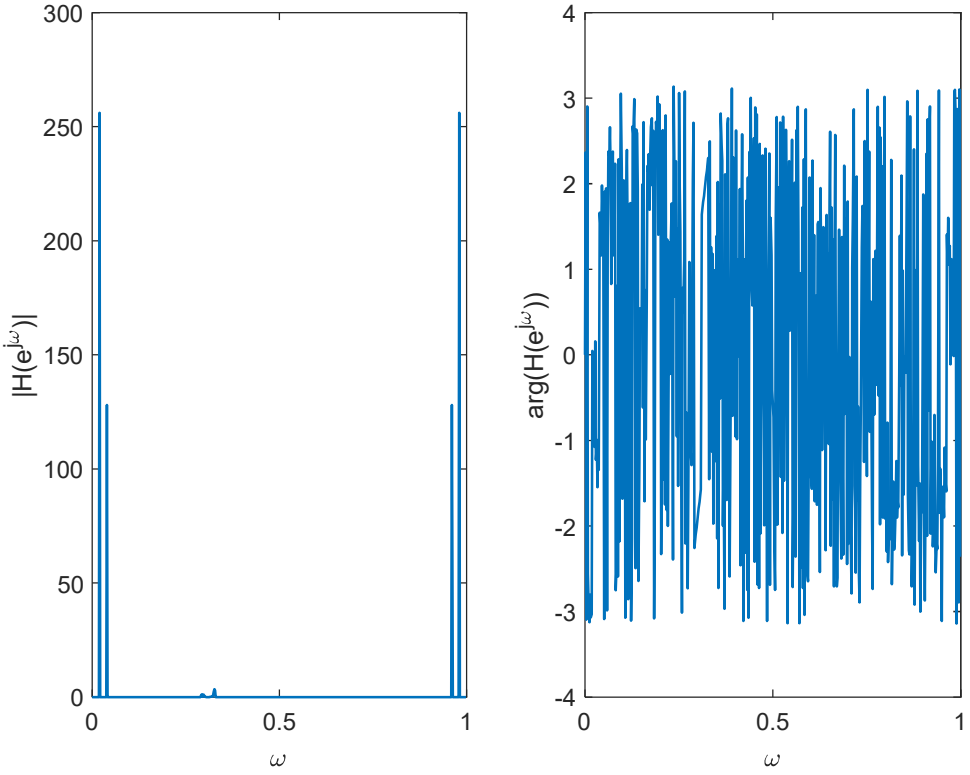
برای فیلتر کردن، در حوزه فوریه، فیلتر و سیگنال را ضرب کردیم.

خروجی‌های تولیدشده:

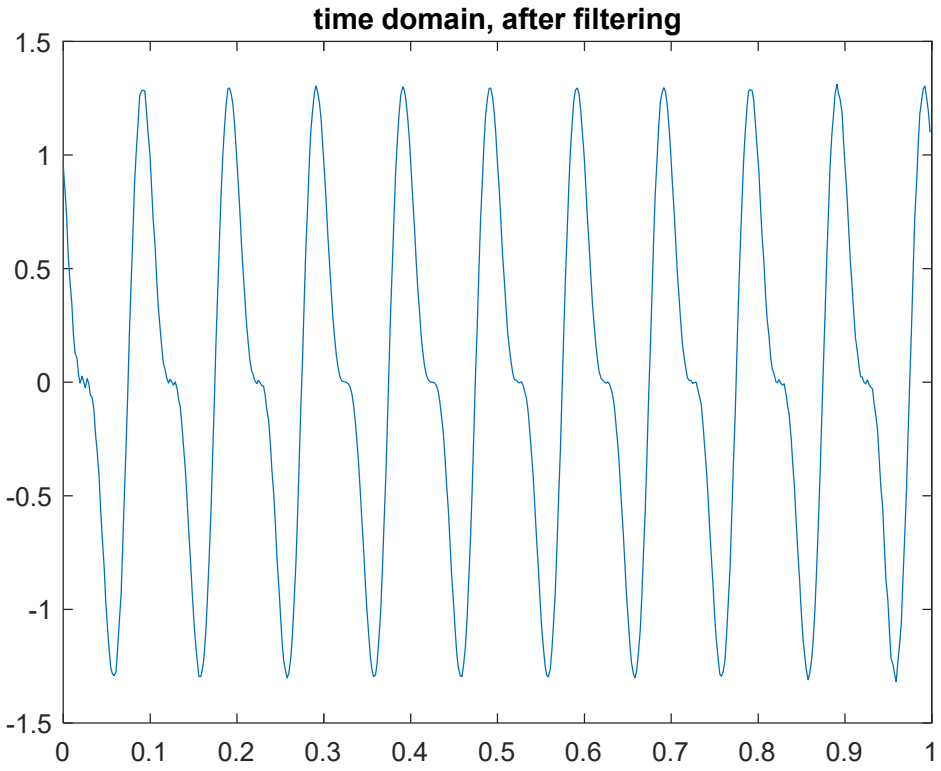
فیلتر طراحی‌شده:



سیگنال فیلتر شده در حوزه فوریه:



سیگنال فیلتر شده در حوزه زمان:

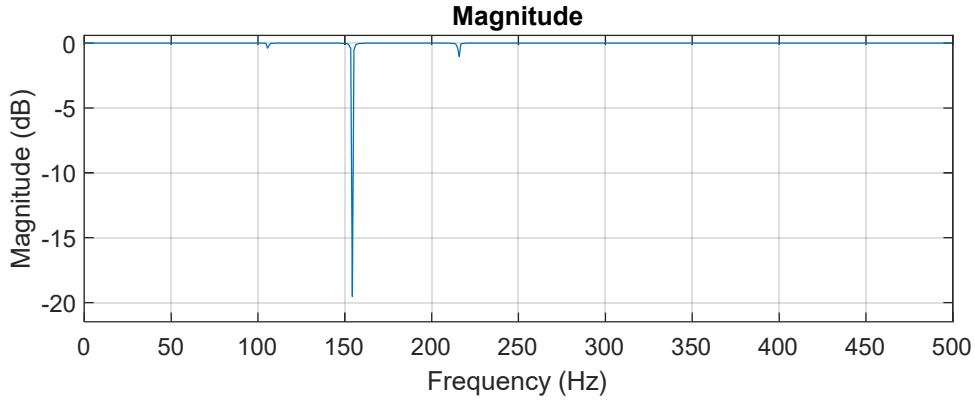


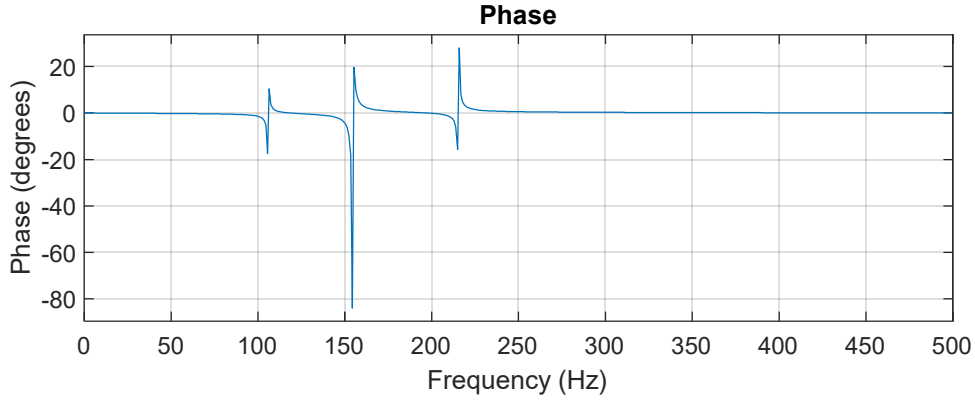
تا مقدار زیادی توانستیم نویز را تضعیف کنیم.

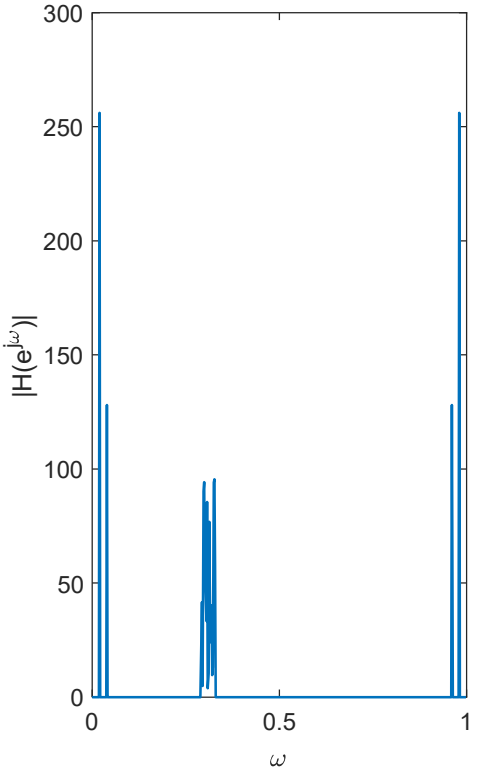
بخش چهارم

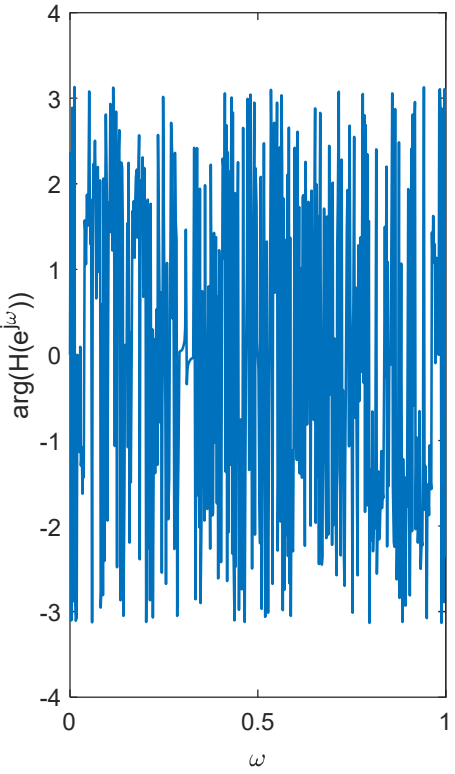
این بار IIR دلخواهی انتخاب می‌کنیم. چپیشف تایپ دو انتخاب شده. مراحل بخش قبلی را طی می‌کنیم.

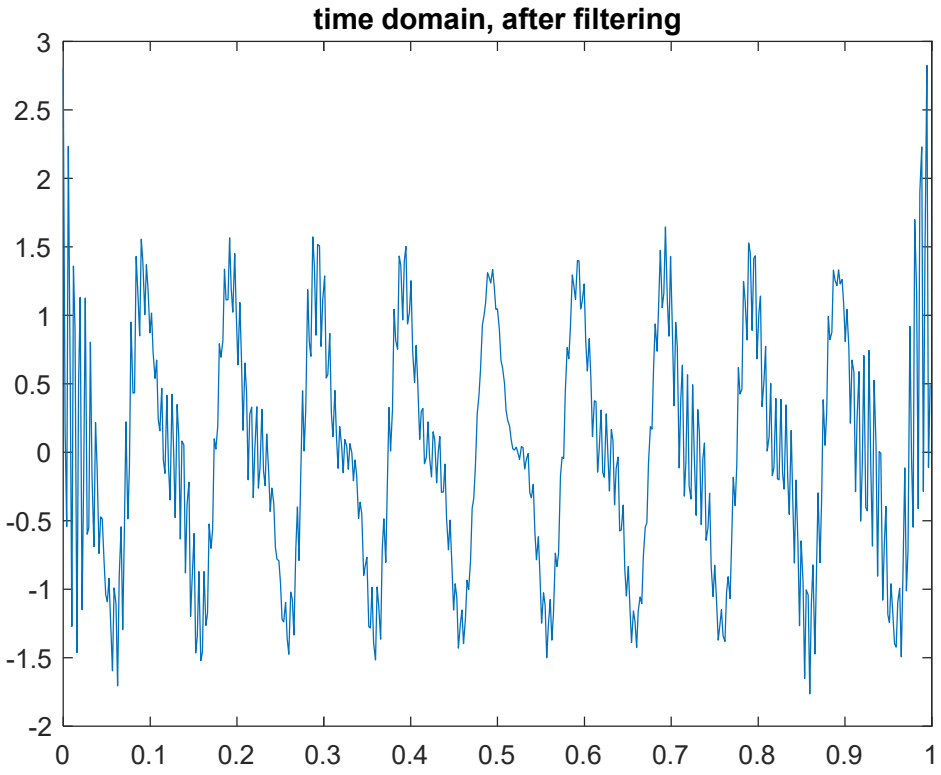
```
%% part 4
fs_des = 1000;
w_p_des = [(0.2) (0.45)];
[b_des,a_des] = cheby2(3,.001,w_p,"stop");
figure
freqz(b_des,a_des,[],fs_des)
[H_ds,W_ds] = freqz(b_des,a_des,[],fs_des);
HH_ds = H_ds';
Y_ds = X.*HH_ds;
freqPlot(Y_ds,x);
y_ds = ifft(Y_ds);
figure
plot(t,y_ds);
title("time domain, after filtering");
```


Magnitude
Magnitude (dB)
0
-5
-10
-15
-20
0
50
100
150
200
250
300
350
400
450
500
Frequency (Hz)


Phase
Phase (degrees)
20
0
-20
-40
-60
-80
0
50
100
150
200
250
300
350
400
450
500
Frequency (Hz)


|H(e^jω)|
300
250
200
150
100
50
0
0
0.5
1
ω


arg(H(e^jω))
4
3
2
1
0
-1
-2
-3
-4
0
0.5
1
ω



بخش پنجم

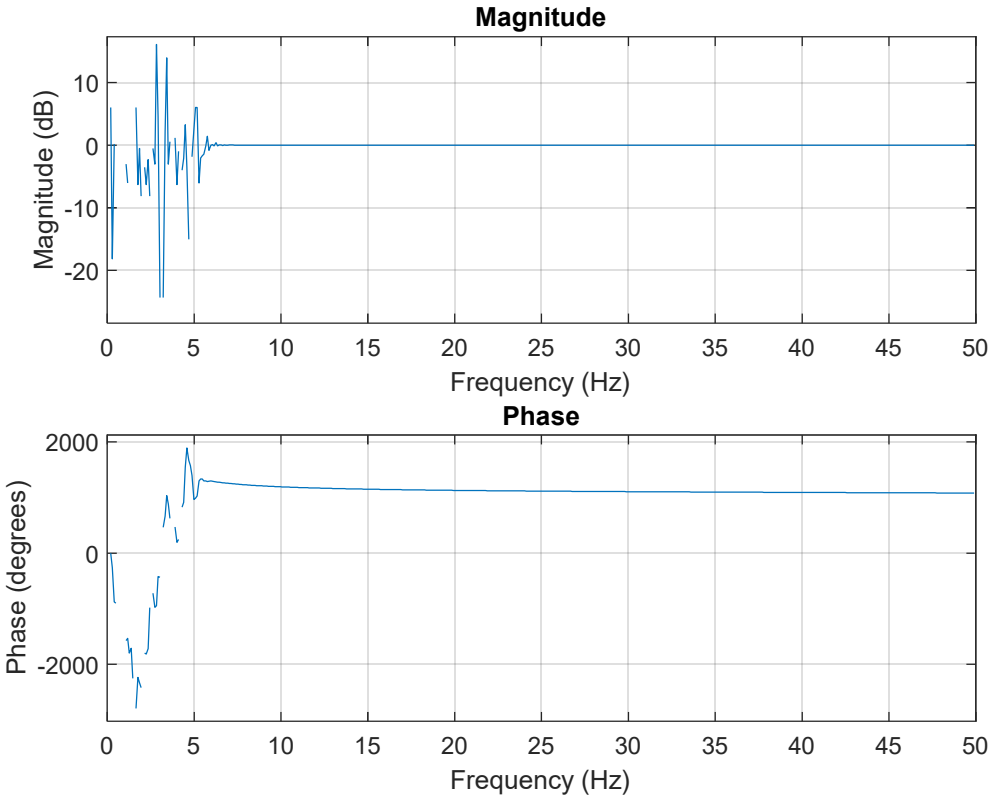
اینجا سیگنال به‌خوبی فیلتر نشد. باید بازبینی‌ای در پارامترهای این نوع فیلتر صورت می‌گرفت و از طرفی ریپل را در باند قطع داشتیم که چنین چیزی ناخوشایند است و ممکن است به‌طور تصادفی، نتیجه نامطلوب یا حتی عکس دهد و نویز را قوی‌تر کند!

بخش ششم

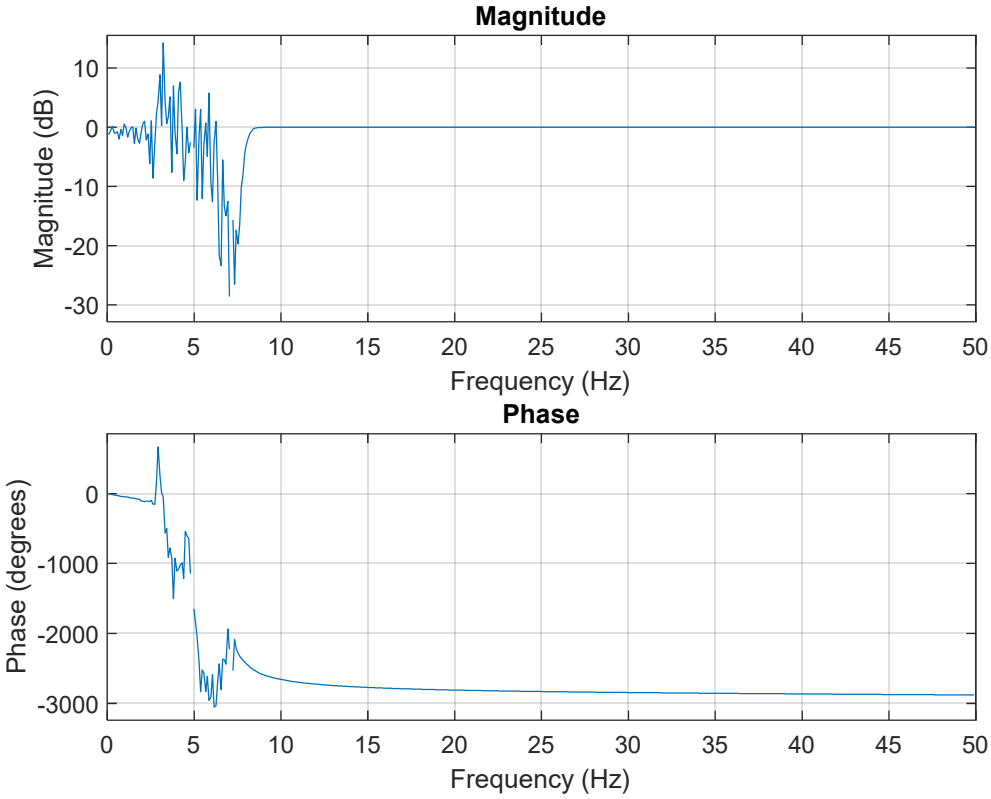
در باندهای خواسته‌شده فیلترهای باتروورث را طراحی می‌کنیم.

```
%% part 6
fs_b = 100;
[bd, ad] = butter(10,[0.02 0.08],"stop");
figure
freqz(bd,ad,[],fs_b)
[bt, at] = butter(10,[0.08 0.16],"stop");
figure
freqz(bt,at,[],fs_b)
[ba, aa] = butter(10,[0.16 0.24],"stop");
figure
freqz(ba,aa,[],fs_b)
[bb, ab] = butter(10,[0.24 0.6],"stop");
figure
```

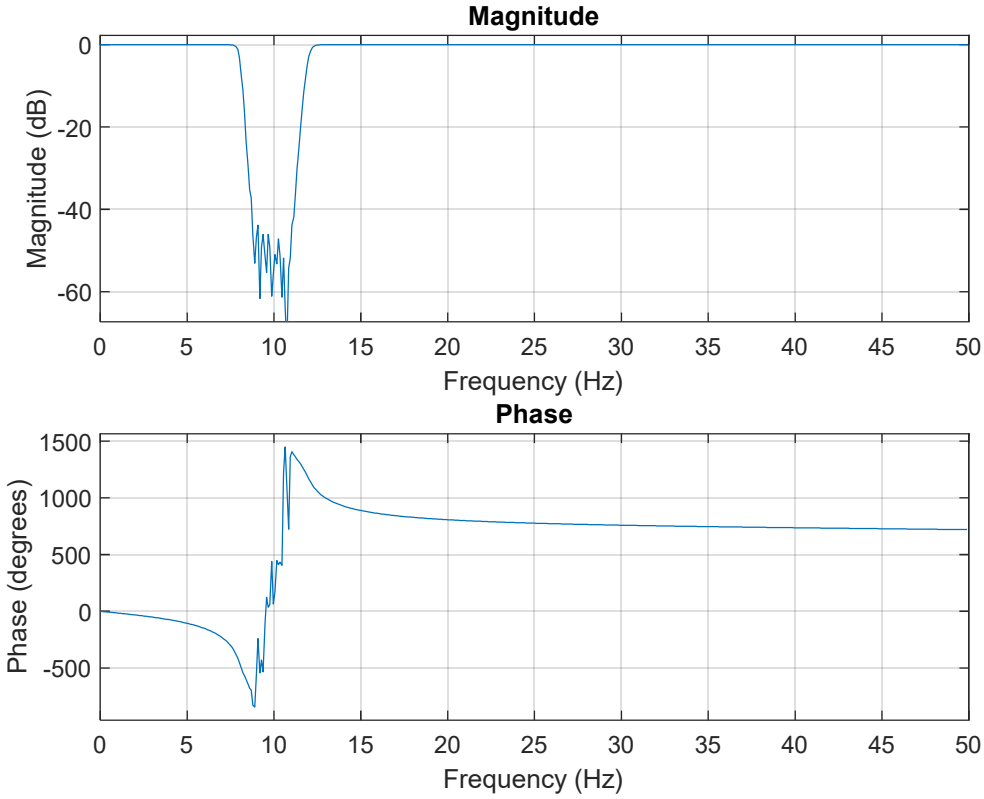
```
freqz(bb,ab,[],fs_b)
```



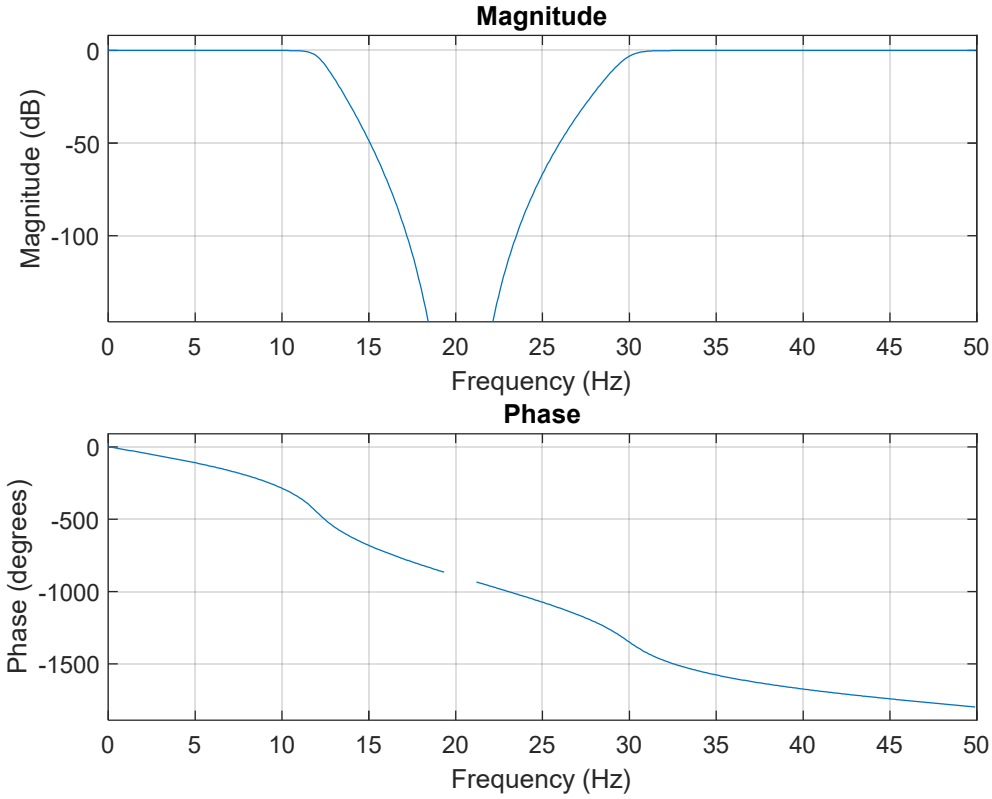
در باند اول دچار از هم‌پاشدگی بهره و فاز شده‌ایم که نتیجه نامطلوبیست.



در باند دوم فقط فاز از هم‌پاشیده؛ اما در باند گذر بهره نیز نوسانات زیادی داریم.



این نتیجه مناسب‌تر به نظر می‌رسد. فاز از هم نپاشیده و فقط در جایی که سیگنال را تضعیف می‌کنیم نوسان داریم اما این نوسانات باعث قوی شدن نویز نمی‌شود چون دامنه مطلوبی دارد و از صفر بیشتر نرفته.



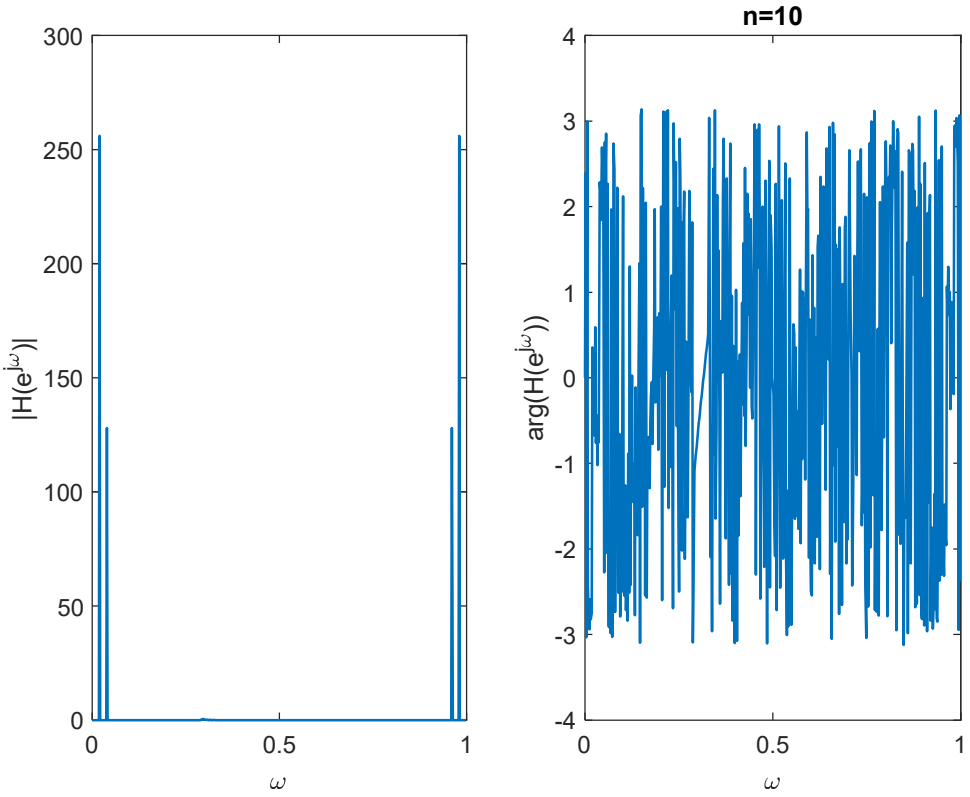
در باند آخری هم فاز دچار از هم پاشیدگی شده.

بخش هفتم

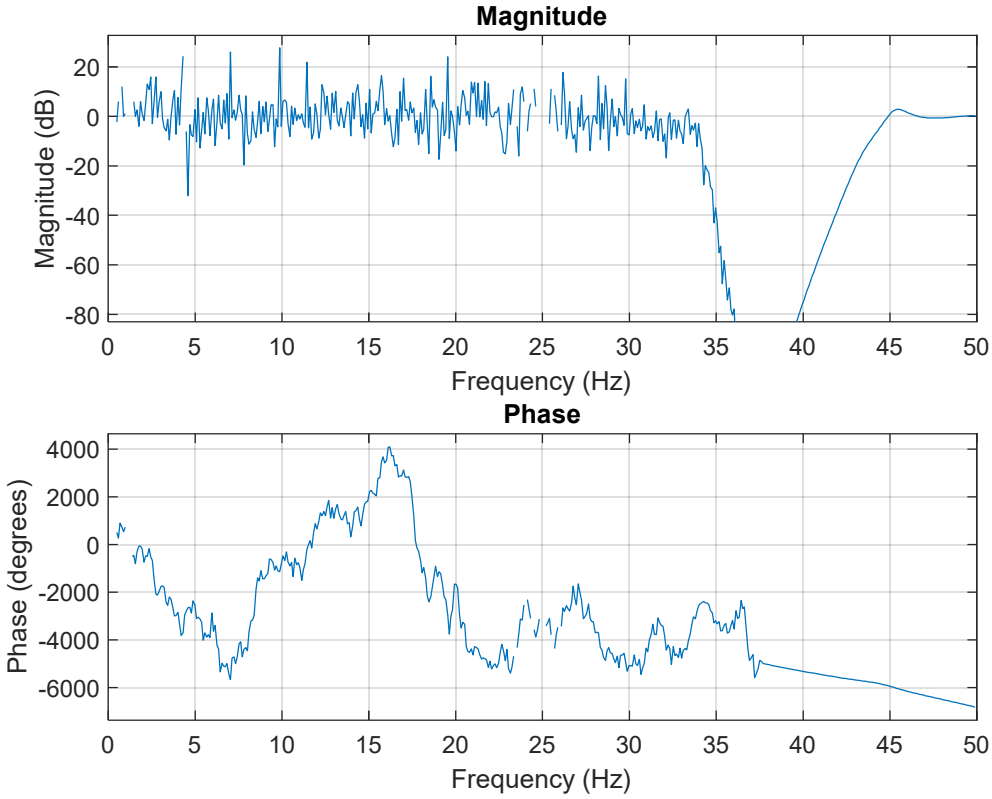
```
%% part 7
[HB, WB]=freqz(bb,ab,[],fs_b);
HHB=HB';
Y2 = X.*HHB;
freqPlot(Y2,x);
title('n=10');
[bb2, ab2] = butter(150,[0.24 0.6],"stop");
figure
freqz(bb2,ab2,[],fs_b)
[H2, W2] = freqz(bb2,ab2,[],fs_b);
HH2 = H2';
Y3 = X.*HH2;
freqPlot(Y3,x);
```

در این بخش با مرتبه 10 و 150 سیگنال نویزدار را فیلتر کردیم و نتایج را مقایسه کردیم.

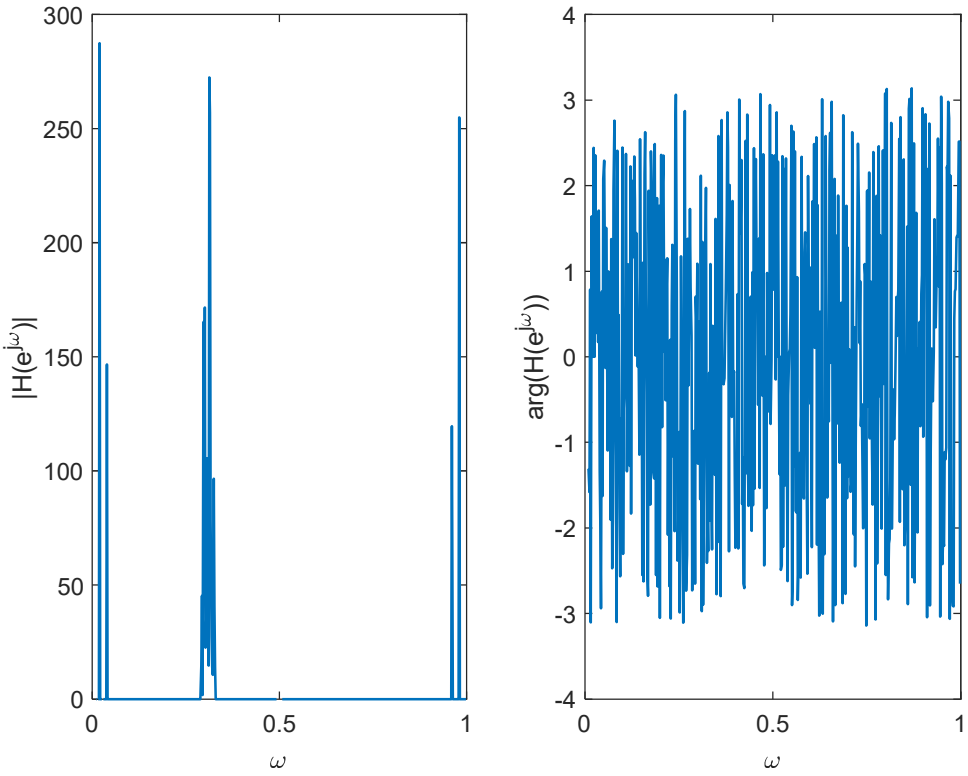
خروجی:



با مرتبه 10 نویز را به‌خوبی از بین بردیم.



شکل بالا فیلتر با مرتبه 150 است. همان‌طور که در بخش سوم گفتم، با افزایش نابه‌جای مرتبه، باندهای فیلتر تغییر می‌کند و جابه‌جا می‌شوند. نوسانات قدرت بیشتری پیدا می‌کنند و می‌توانند با این کار حتی نویز را تقویت کنند.



همان‌طور که می‌بینیم تاثیر عکس روی نویز گذاشته. پس باید مرتبه‌ای انتخاب کنیم که نه خیلی زیاد باشد که باندهای قطع و عبور فیلتر را جابه‌جا کند، نه خیلی کم باشد که تضعیف نویز را به‌خوبی انجام ندهد.

به‌نظر می‌رسید مرتبه 10 مقدار مناسبی بوده.

سوال سوم

بخش صفرم

در ابتدای سوال خواسته شده تا پاسخ فرکانسی را برحسب تابعی از A به‌دست بیاوریم.

```
clc;clear;close all hidden;close all
%% part 0
syms A;
h = [1 A 1];
syms w
for i=1:length(h)
    H_samp(i) = h(i)*exp(-sqrt(-1)*w*(i-1));
end
H = sum(H_samp);
disp(H)
```

خط اول را در تمامی کدها قرار می‌دهیم.

ابتدا به‌طور سمبولیک A را تعریف می‌کنیم. همین کار را برای w انجام می‌دهیم تا بتوانیم تابع سمبولیک فوریه آن را محاسبه کنیم.

نتیجه‌ای که در command window نمایش داده می‌شود:

```
exp(-w*2i) + A*exp(-w*1i) + 1
```

در ادامه‌ی کد و این سری از تمرین کامپیوتری، تابعی نوشته شده است. این تابع مقادیر فوریه و سیگنال اصلی را می‌گیرد و اندازه و فاز آن‌ها را رسم می‌کند.

```
function freqPlot(X,x)
    figure
    w_sample=1/length(x)*(0:length(x)-1);
    subplot(1,2,1)
    plot(w_sample,abs(X),"LineWidth",1);
    xlabel('\omega');
    ylabel('|H(e^{j\omega})|')
    hold on
    subplot(1,2,2)
    plot(w_sample,angle(X),"LineWidth",1);
    xlabel('\omega');
    ylabel('arg(H(e^{j\omega}))')
end
```

درباره‌ی به‌دست آوردن مقادیر صحیح برای A، در بخش‌های بعدی توضیح داده خواهد شد.

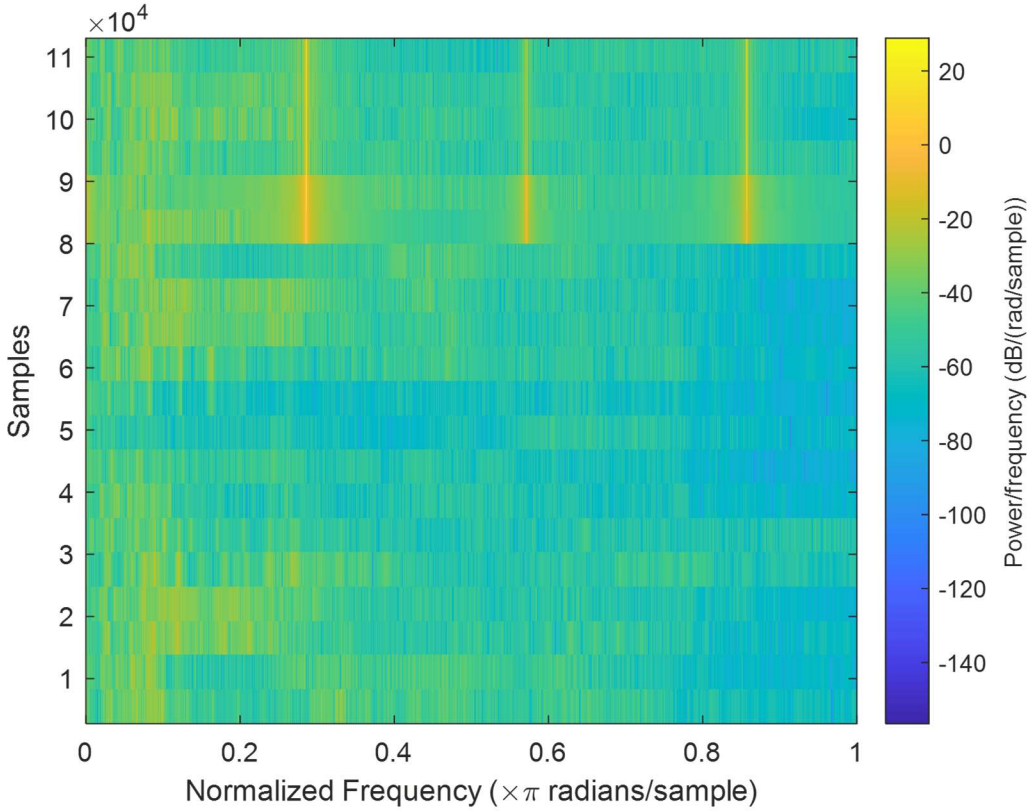
بخش اول

```
%% part 1
[x , fs] = audioread('sound.wav');
```

```
figure;
spectrogram(x ,fs);

X = fft(x);
freqPlot(X,x)
```

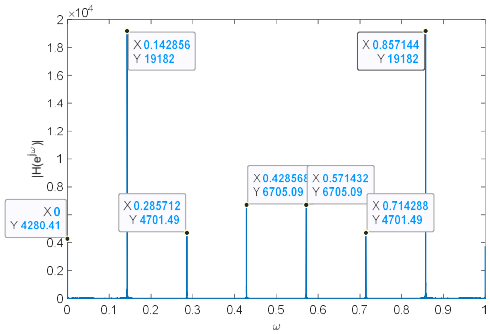
در این بخش spectrogram و تبدیل فوریه فایل صوتی را رسم می‌کنیم.

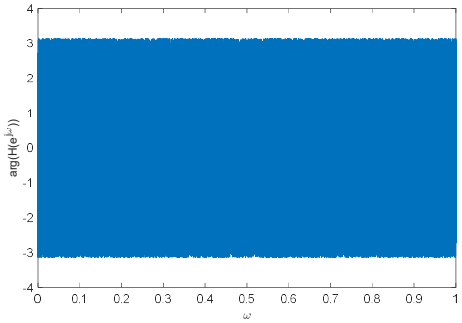
نتایج:



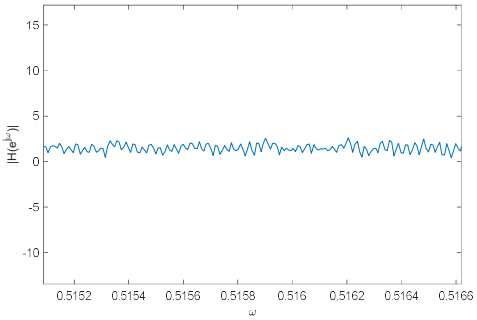
به‌نظر می‌رسد در نقاطی که زرد هستند، یعنی قدرت نویز بسیار بالاست و نویز در همان فرکانس‌ها قرار گرفته است.

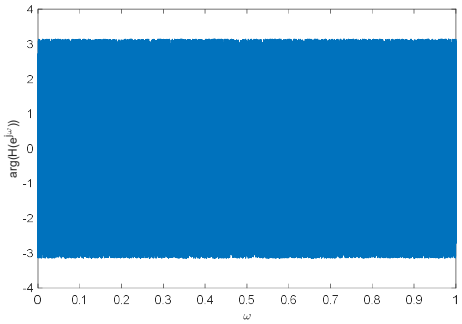
حال به نمایش فوریه توجه می‌کنیم.





سمت چپ، اندازه نمایش فوریه سیگنال صوتی را می‌توانیم ببینیم. اندازه نویزی که در فرکانس تقریبا 0.87پی رادیان و 0.142 قرار گرفته‌اند، از بقیه بیشتر است. (عدد پی در فرکانس‌های نام‌برده ضرب می‌شود) چطور می‌توانیم بگوییم که این‌ها همان نویزها هستند؟ اگر بیشتر زوم کنیم، متوجه صدای صحبت گوینده می‌شویم که با توجه به نویز، قدرت کم‌تری دارد با توجه به آن‌چه می‌شنویم.





موقع به‌دست آوردن فرکانس‌ها، باید توجه داشته باشیم که محدوده محور فرکانس تبدیل فوریه، همانند محور فرکانس spectrogram باشد تا فرکانس‌های به‌دست آمده باهم تطبیق داشته باشند.

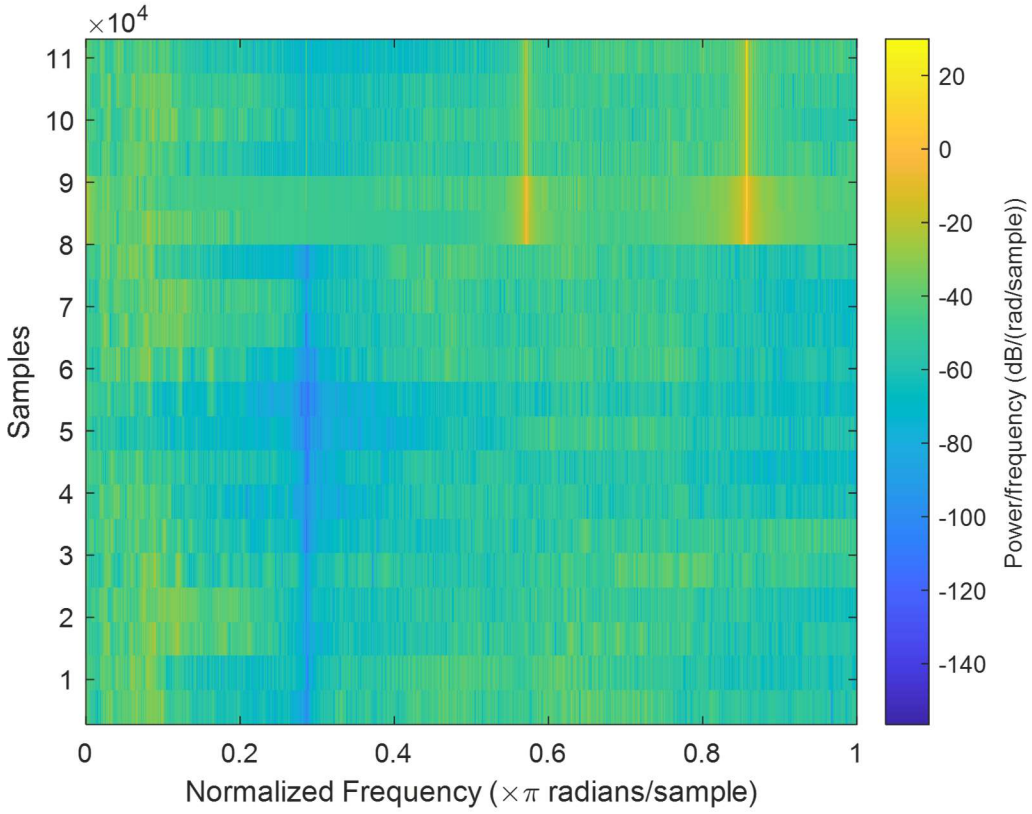
بخش دوم و سوم

با استفاده از معادله ۲ و بخش راهنمایی، فیلتر را طراحی می‌کنیم. درواقع فرکانس‌های به‌دست آمده را در معادله قرار می‌دهیم.

از فرکانس نزدیک به صفر استفاده نشد زیرا تاثیر مطلوبی ایجاد نمی‌کرد. فرکانس 0.714 باعث می‌شد بخشی که نویز می‌آمد کلاً از بین برود و هیچ‌چیزی شنیده نمی‌شد. همین امر برای فرکانس نزدیک 1 نیز صادق بود.

پس از گذاشتن هر فرکانس، spectrogram را رسم می‌کنیم تا تفاوت ایجاد شود. درواقع ۵تا فیلتر را به‌صورت سری می‌بندیم.

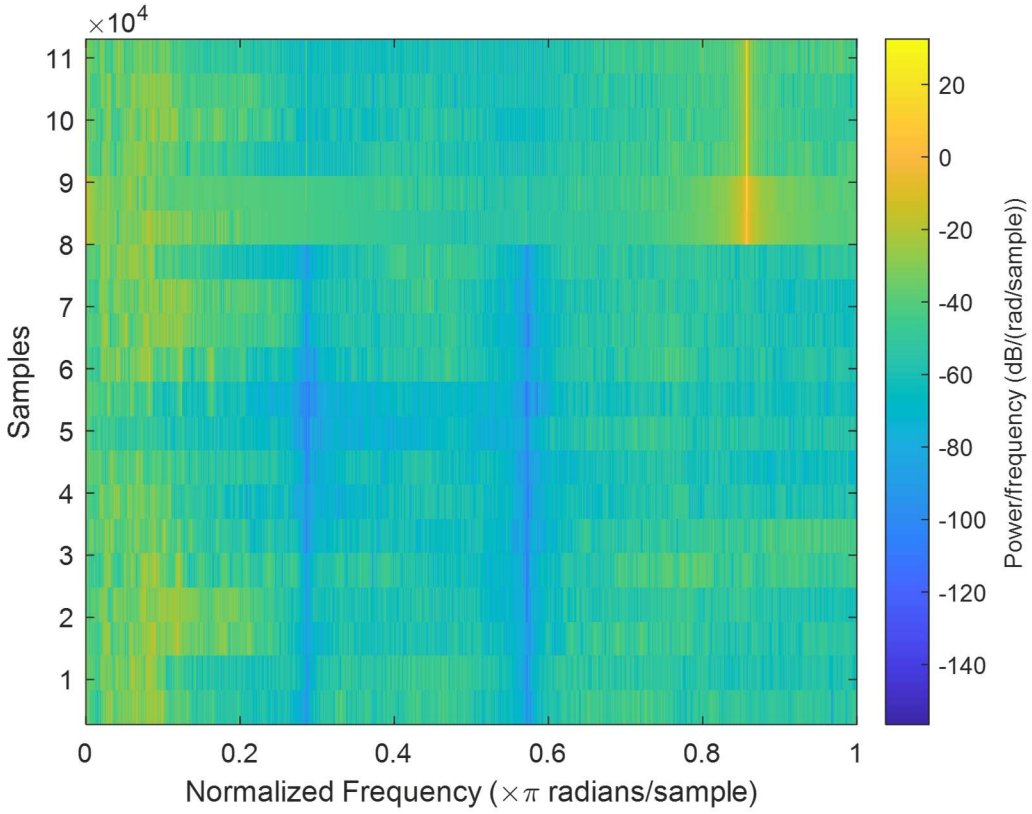
فیلتر اول



نویزی که بین 0.2 و 0.4 بود از بین رفت.

فیلتر دوم:

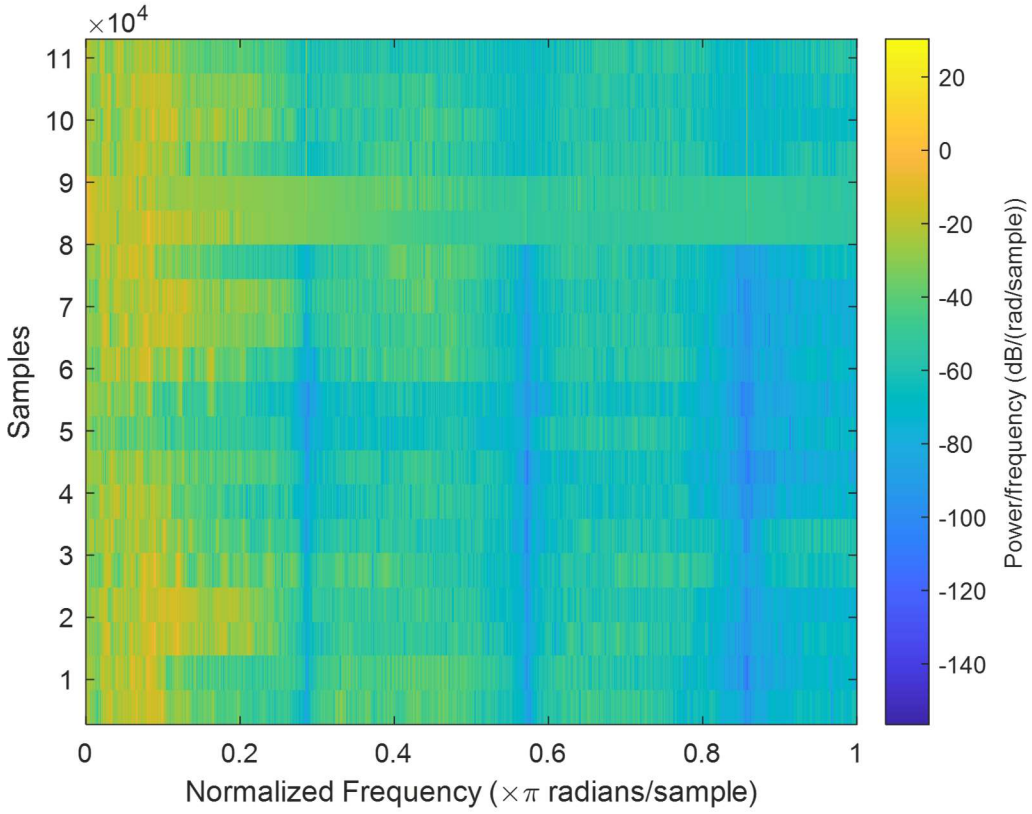
```
num = [1 -2*cos(pi*.5724) 1];
den = 1;
y2 = filter(num , den, y1);
figure
spectrogram(y2,fs);
```



تا اینجای کار آن نویز ناخوشایند از بین نرفته است.

فیلتر سوم:

```
num = [1 -2*cos(pi*.8567) 1];
den = 1;
y3 = filter(num , den, y2);
figure
spectrogram(y3,fs);
```



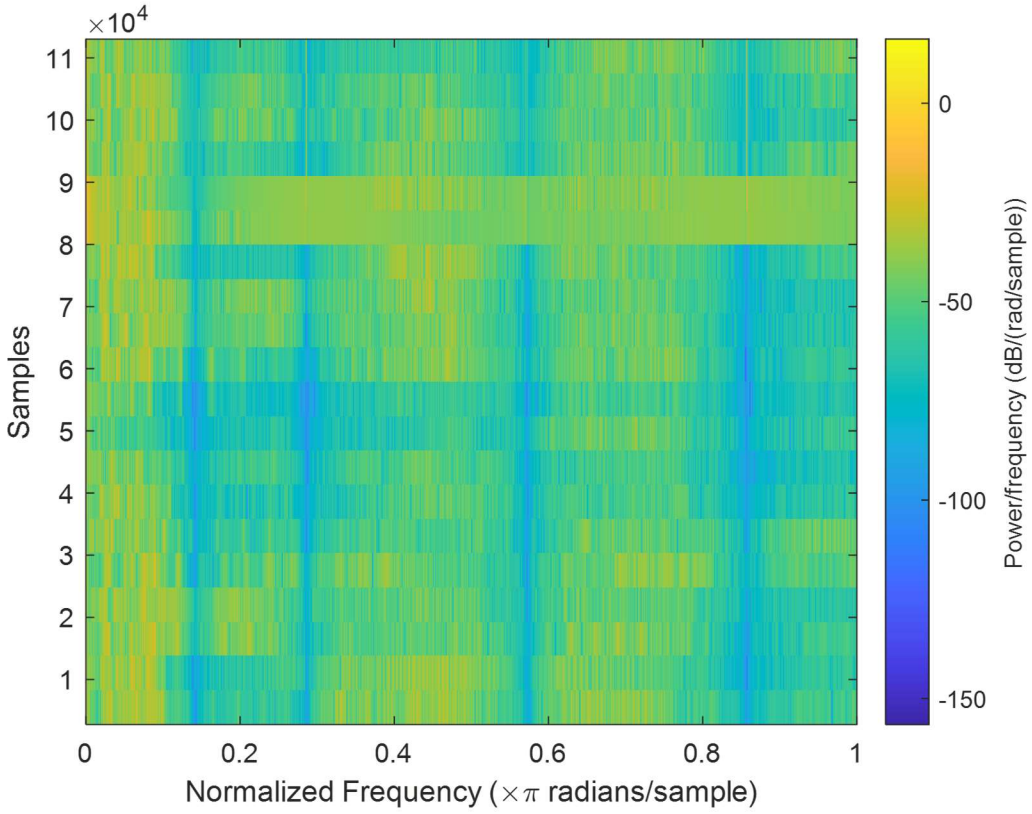
اکنون شدت آن نویز ناخوشایند بسیار کم‌تر شده؛ ولی هم‌چنان خس‌خسی وجود دارد و جای بهتر شدن دارد.

فیلتر چهارم:

اگر به نمودار بالا توجه کنیم، می‌بینیم که در فرکانس‌های پایین، صدایی با شدت بالا وجود دارد. این صدا مربوط به همان خس‌خس است که باعث می‌شود صدای گوینده در کل زمان، حتی قبل از آمدن آن نویز ناخوشایند، به‌وضوح شنیده نشود.

این نویز دیگر، در فرکانس 0.142پی وجود دارد.

```
num = [1 -2*cos(pi*.142) 1];
den = 1;
y4 = filter(num , den, y3);
figure
spectrogram(y4,fs);
```

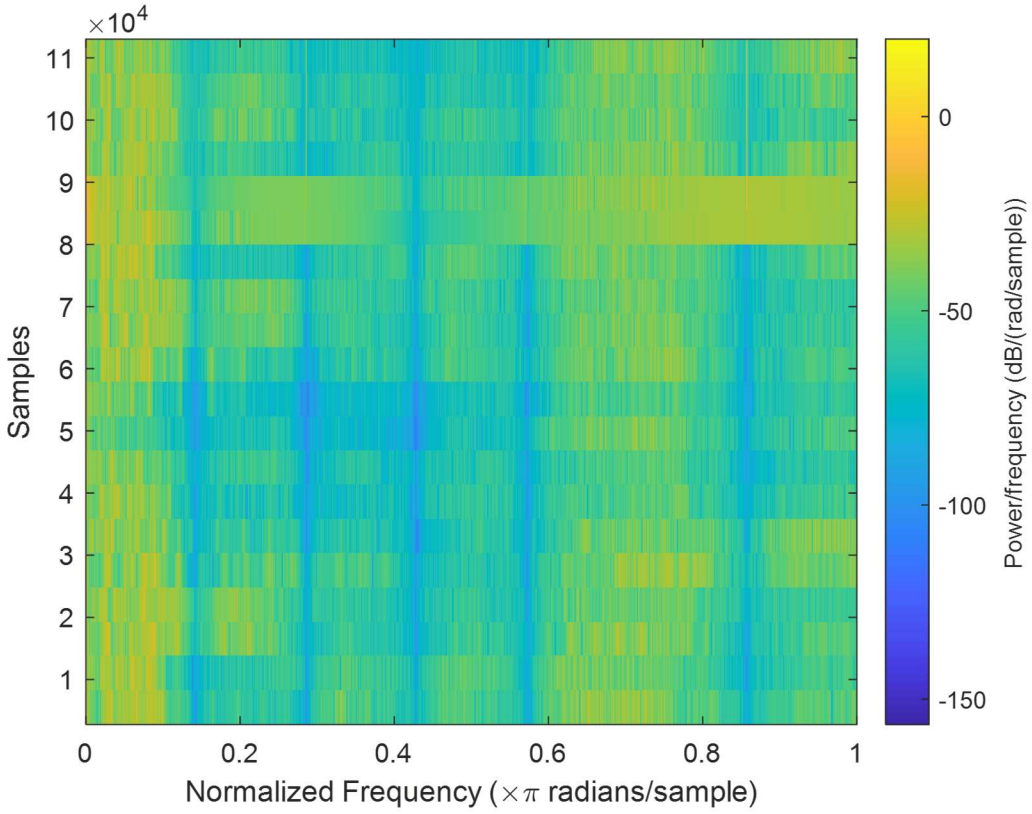


اکنون خس‌خس از بین رفته و وضوح صدای گوینده بیشتر است.

فیلتر پنجم:

حال نویز دیگری که در نمایش فوریه دیده شد از بین می‌بریم.

```
num = [1 -2*cos(pi*.428) 1];
den = 1;
y5 = filter(num , den, y4);
figure
spectrogram(y5,fs);


sound(y5,fs);
audiowrite("output.wav",y5);
```



اکنون به صدای مطلوبی رسیدیم. نتیجه در فایلی به نام output ذخیره شده است.

بهره DC برابر می‌شود با:

$2 - 2\cos(\hat{}\omega 0)$

اگر به معادله 2 توجه کنیم، می‌فهمیم که مقادیر A مورد نیاز برای هر فرکانس برابر است با:

$-2\cos(\hat{}\omega 0)$

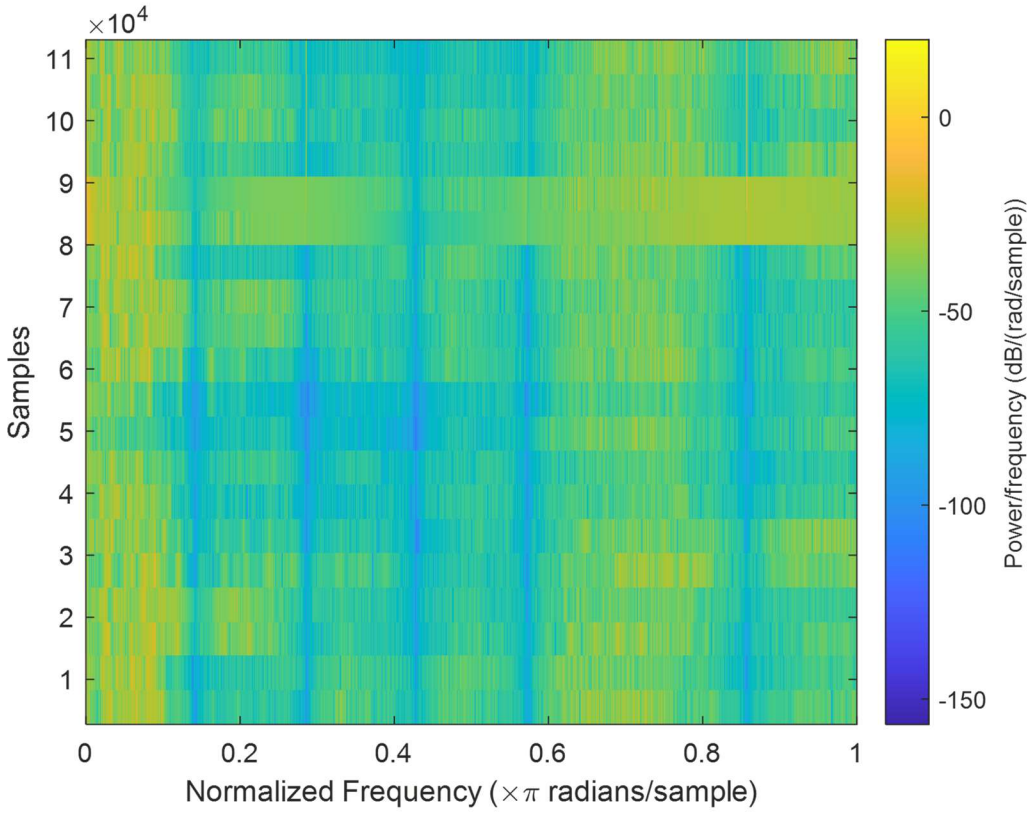
با قرار دادن ۵ فرکانس نام‌برده می‌توانیم مقادیر A به‌دست بیاوریم.

```
%% part impulse response
hh1 = [1 -2*cos(pi*.286) 1];
hh2 = [1 -2*cos(pi*.5724) 1];
hh3 = [1 -2*cos(pi*.8567) 1];
hh4 = [1 -2*cos(pi*.142) 1];
hh5 = [1 -2*cos(pi*.428) 1];
hh12 = conv(hh1,hh2);
hh34 = conv(hh3,hh4);
hh1234 = conv(hh12,hh34);
hh = conv(hh5,hh1234);
disp(hh);
yy = filter(hh,1,x);
figure
```

```
spectrogram(yy,fs);
```

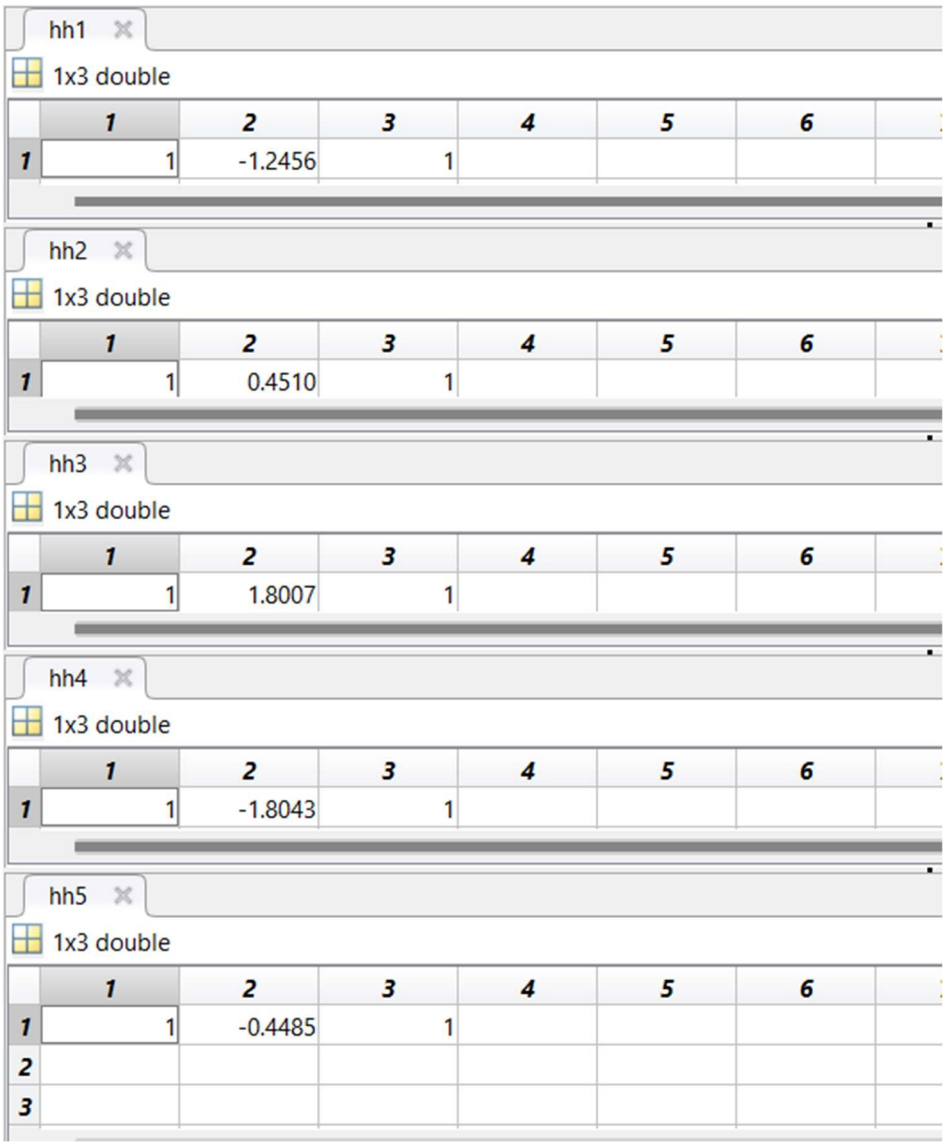
حال پاسخ ضربه کل سیستم را به‌دست می‌آوریم. اتصال سری ۵ فیلتر بالا، کانوولوشن‌شان در حوزه زمان است.

1.0000 -1.2467 1.5501 -0.6951 0.3164 0.2846 0.3164 -0.6951 1.5501 -1.2467 1.0000



شکل رسم شده برای ۵ فیلتر کانوولوشن‌شده است.

مقادیر A در هر فیلتر:



hh1

1x3 double

	1	2	3	4	5	6
1	1	-1.2456	1			

hh2

1x3 double

	1	2	3	4	5	6
1	1	0.4510	1			

hh3

1x3 double

	1	2	3	4	5	6
1	1	1.8007	1			

hh4

1x3 double

	1	2	3	4	5	6
1	1	-1.8043	1			

hh5

1x3 double

	1	2	3	4	5	6
1	1	-0.4485	1			
2						
3						

سوال چهارم

در این مقاله راجع به روش جدیدی برای پردازش گفتار مطرح شد. انواع روش‌هایی مثل CNN یا DNN، به معنای شبکه‌های عصبی کانوولوشنی یا شبکه‌های عصبی عمیق وجود داشت؛ اما فرقی که به‌نظر می‌رسید این روش داشت، علاوه بر شبکه‌ی عصبی تعلیم‌دیده، استفاده از فیلترهای طول محدود (FIR) بوده.

این فیلتر با استفاده از تابع معروف Sinc ساخته می‌شود؛ به دلیل ایده‌آل بودن آن (به معنای وجود ریپل‌های حداقلی در باند عبور) است.

در طراحی فیلتر که از تکنیک windowing استفاده شده، پنجره مورد نظر محقق از نوع Hann بوده. زیرا حساسیت این نوع پنجره به فرکانس‌های بالا، مطلوب شناسایی شده است. به‌نظر می‌رسد که محقق به پنجره‌های دیگر نپرداخته است.

در بخش نتیجه‌گیری، عملکرد SincNet، به سه معیار سنجیده شده - آنالیز فیلتر، شناسایی گفتار، تایید گوینده به تشخیص مدل - عملکرد مطلوب‌تر را مدل SincNet داشته؛ مثلاً در مقایسه با مدل CNN استاندارد، نتیجه‌ی بهتری ضبط شده است.