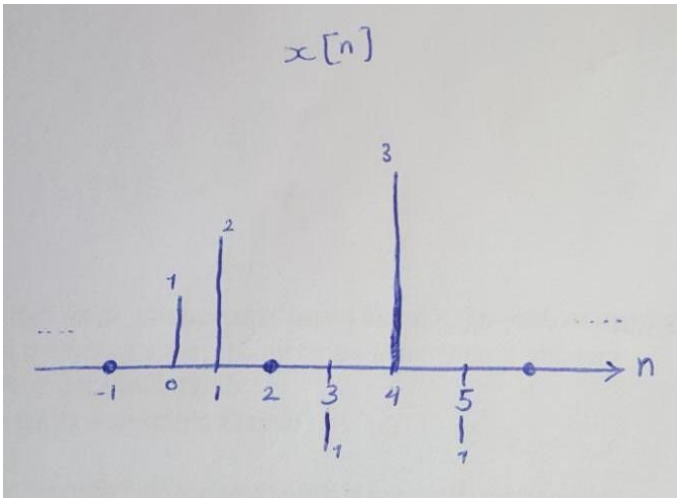
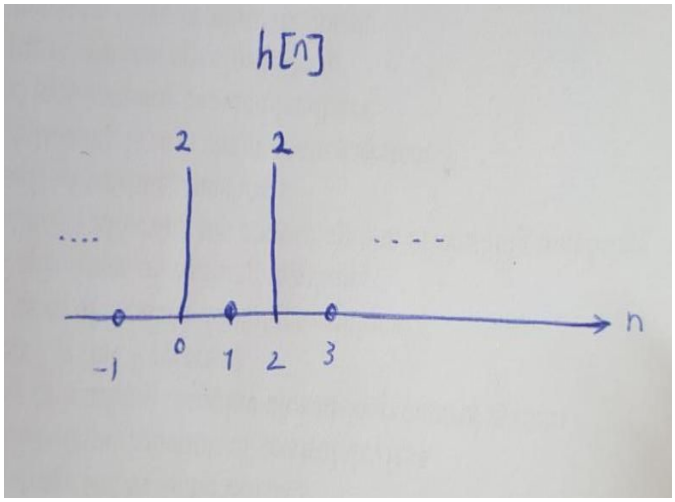
سوال ۱

کدهای مربوط به این سوال در پوشه‌ی کد در فایلی به نام E1_810194346 موجود است:

الف) شکل توابع x و h به صورت زیر اند:





به همین دلیل این سیگنال‌ها را به شکل زیر در متلب تعریف می‌کنم. (x1 همان x است و x2 همان x[n-2] است که ۲ واحد شیفت خورده‌ی x1 به راست است.)

```
x1 = [1, 2, 0, -1, 3, -1, 0, 0, 0, 0];
x2 = [0, 0, 1, 2, 0, -1, 3, -1, 0, 0];
n = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
h = [2, 0, 2, 0, 0, 0, 0, 0, 0, 0];
```

سپس مطابق خواسته‌ی سوال کانولوشن های خواسته شده را به صورت زیر انجام می‌دهیم:

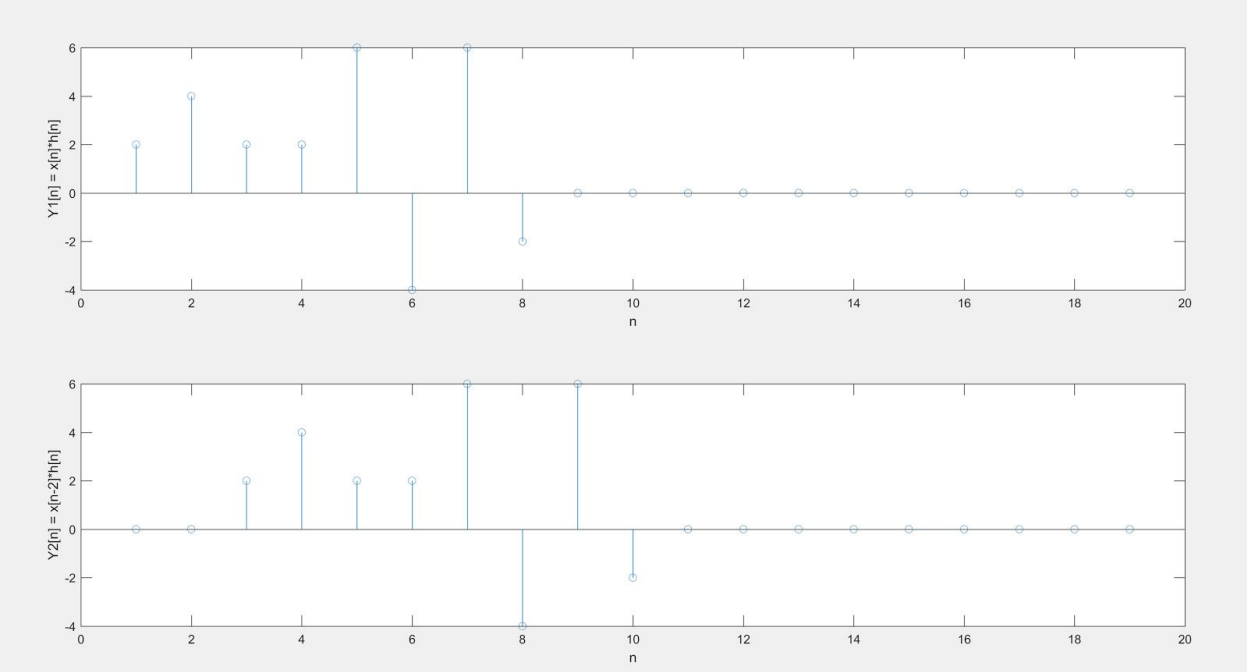
```
y1 = conv(x1, h);
y2 = conv(x2, h);
```

و در نهایت برای اینکه بتوانیم هر دو خروجی را در یک شکل رسم کنم دستورات رو برو را نوشتم کردم:

```
subplot(2, 1, 1);
stem(y1);
xlabel('n');
ylabel('Y1[n] = x[n]*h[n]');

subplot(2, 1, 2);
stem(y2);
xlabel('n');
ylabel('Y2[n] = x[n-2]*h[n]');
```

خروجی برنامه به شکل زیر بود:



ب) این سیستم تغییر پذیر با زمان نیست و می‌تواند تغییرناپذیر با زمان باشد. چون طبق تعریف سیگنالی تغییر ناپذیر با زمان است که رفتار آن با زمان تغییر نکند به عبارت دیگر اگر ورودی p واحد شیفت خورد خروجی هم p واحد شیفت بخورد. که با توجه به شکل بالا می‌بینیم زمانی که ورودی ۲ واحد به راست شیفت پیدا کرده خروجی نیز ۲ واحد به راست شیفت خورده است. ولی نمی‌توانیم بگوییم که قطعا تغییر ناپذیر با زمان است چون باید به ازای تمام ورودی ها چنین باشد ولی می‌توانیم با اطمینان بگوییم تغییر پذیر با زمان نیست چون شکل‌های بالا مثال نقضی برای تغییر پذیر بودن با زمان اند.

سوال ۲

کدهای مربوط به این سوال در پوشه‌ی کد در فایلی به نام E2_810194346 موجود است:

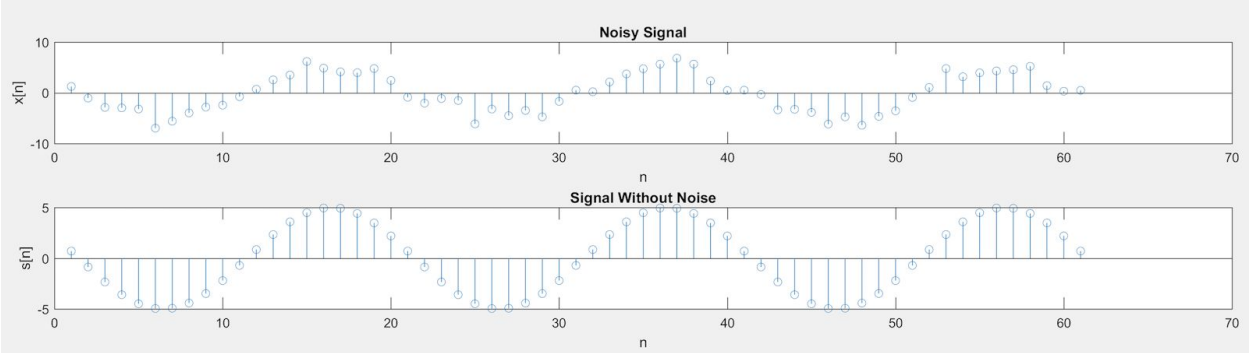
الف) برای رسم تابع x ابتدا s را با توجه به فرمول داده شده در صورت سوال تعریف می‌کنیم و تابع Zn که نویز روی سیگنال است را نیز به صورت روبرو تعریف می‌کنیم (که در آن m تعداد اعداد تصادفی است که می‌خواهیم تولید کند)
برای اینکه بتوانم خروجی x را که همان نویز دار s را با سیگنال بدون نویز یعنی همان s مقایسه کنم به صورت زیر دستور رسم شدن هر دو سیگنال را وارد کردم:

```
Zn = randn(1, m);
n = [0:1:60];
Sn = [];
Xn = [];
Yn = zeros(61);
Yn2 = zeros(61);
Sn = 5*sin((2*pi*n)/20 + 3);
Xn = Sn + Zn;
```

```
subplot(4, 1, 1);
stem(Xn);
title('Noisy Signal');
xlabel('n');
ylabel('x[n]');
```

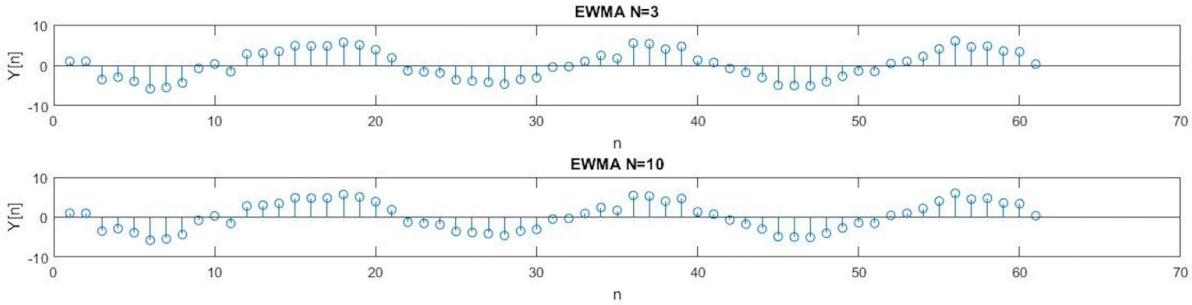
```
subplot(4, 1, 2);
stem(Sn);
title('Signal Without Noise');
xlabel('n');
ylabel('s[n]');
```

خروجی این بخش به صورت زیر شد:



البته خروجی رسم شده برای x می‌تواند پس از هر بار اجرا کردن برنامه متفاوت باشد چون هر بار یک سری عدد تصادفی جدید تولید می‌شود با این حال شکل کلی آن به صورت بالا خواهد بود.

ب و ج) خروجی این دو بخش به شکل زیر است:



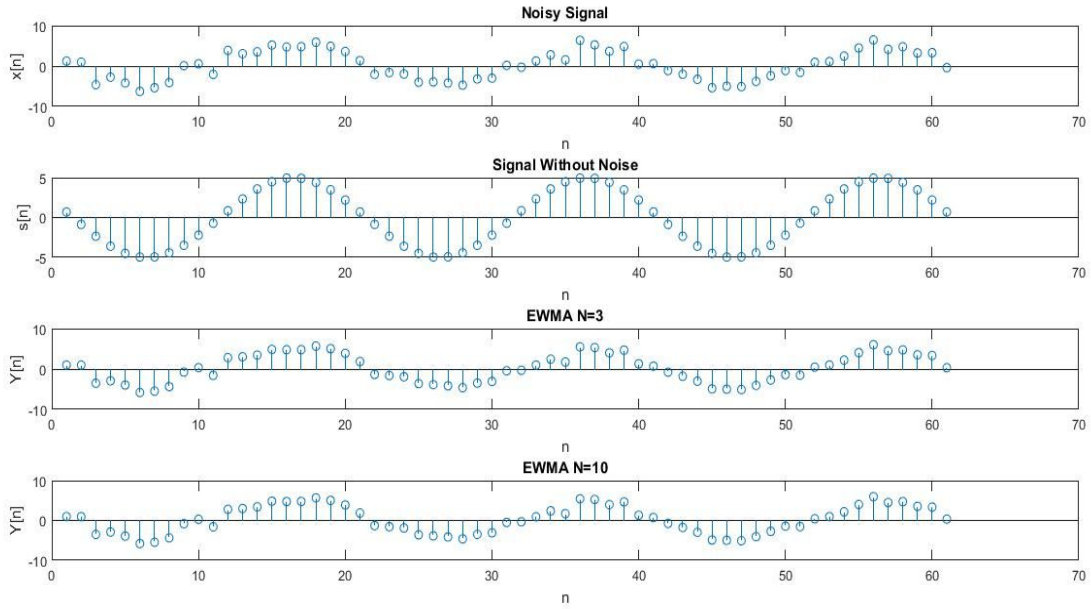
برای کشیدن این نمودار ها به صورت زیر عمل کردیم:
متغیرهای b و N را تعریف کردم و سپس با استفاده از ۲ حلقه‌ی تو در توی زیر مقدار جمع انباره‌ای صورت سوال را پیاده سازی کردم و در نهایت با استفاده از دستور stem تابع حاصل را رسم کردم.

```
b = 0.2;
N2 = 10;
N = 3;
```

```
for j = 0:60
    for i = 0:N-1
        if((j-i)>=0)
            Yn(j+1) =(b.^i)*(Xn(j-i+1)) + Yn(j+1);
        end
    end
    Yn(j+1) = (1-b)/(1-(b.^N))*Yn(j+1);
end
```

```
subplot(4, 1, 3);
stem(Yn);
title('EWMA N=3');
xlabel('n');
ylabel('Y[n]');
```

د) برای مقایسه‌ی حاصل مرحله‌ی ب و ج با سیگنال بدون نویز از خروجی زیر استفاده کردم:(در پایین آمده است)
با توجه به این شکل به تحلیل Yn بر اساس سیگنال بدون نویز می‌پردازم:



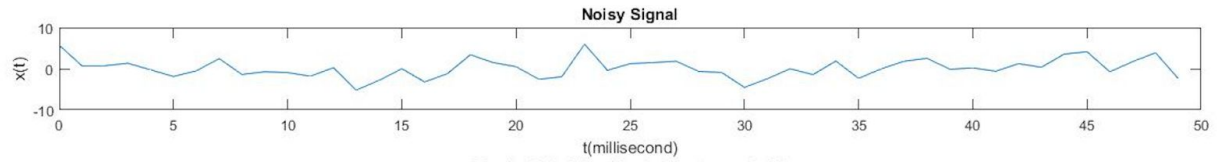
طبق فرمول انتظار داریم: نمودار ها با زیاد شدن n به شکل کلی سیگنال بدون نویز نزدیک شده و از سیگنال نویز دار کمی فاصله می‌گیرند چون چندین نقطه‌ی کنار هم را با هم جمع می‌کنیم پس پایین و بالا شدن‌های ناگهانی سیگنال نویز دار اندکی جبران شده و شکل به شکل S و شکل کلی سینوسی نزدیک می‌شود
اما با توجه به شکل می‌بینیم که میان دو نمودار تفاوت چندانی وجود ندارد چون مقدار b به قدری کم است و اختلاف مقادیر n نیز به اندازه‌ای نیست که باعث ایجاد تغییر بزرگی شود. ولی به طور کلی از شکل کلی سینوسی پیروی می‌کنند

ه) همان طور که می‌بینیم سیگنال Y شباهت زیادی به سیگنال سینوسی یعنی همان سیگنال دوم دارد با این تفاوت که هر چه N بزرگ تر می‌شود این شکل منظم تر می‌شود در واقع چون روش ما به این شکل است که برای محاسبه‌ی هر Y[n] تعدادی نمونه از سیگنال با نویز را برمی‌داریم و انگار میان آن‌ها میانگین می‌گیریم پس هر چه N بزرگتر می‌شود انگار نمونه‌های بیشتری را برمی‌داریم پس اگر یکی از نمونه ها خیلی پرت هم باشد چون با بقیه جمع شده تعدیل شده و شکل به سینوسی منظم نزدیک تر می‌شود و به دلیل نوع نوشتن فرمول بازه‌ای که روی آن روی محور عمودی تعریف شده نیز کوچکتر می‌شود.

افزایش b: اگر در بازه‌ی ۰ تا ۱ زیاد شود مخرج کوچکتر شده پس کلا کسری که ضرب می‌شود بزرگتر می‌شود پس سیگنال در راستای محور عمودی بازتر خواهد شد و اگر در بازه‌ی ۱ تا بینهایت جابجا شود چون هم صورت و هم مخرج می‌شود پس کلا + می‌ماند و نسبت به محور افقی قرینه نمی‌شود ولی چون مخرج بزرگتر می‌شود پس شکل کلی تولید شده در راستای محور عمودی جمع تر می‌شود.

سوال ۳

الف) سیگنال نویزی را طبق فرمول‌های داده شده رسم می‌کنیم خروجی به شکل زیر است:



البته این سیگنال در هر بار خروجی گرفتن می‌تواند شکل متفاوتی داشته باشد چون هر بار اعداد تصادفی جدیدی تولید شده و با آن جمع می‌شود.

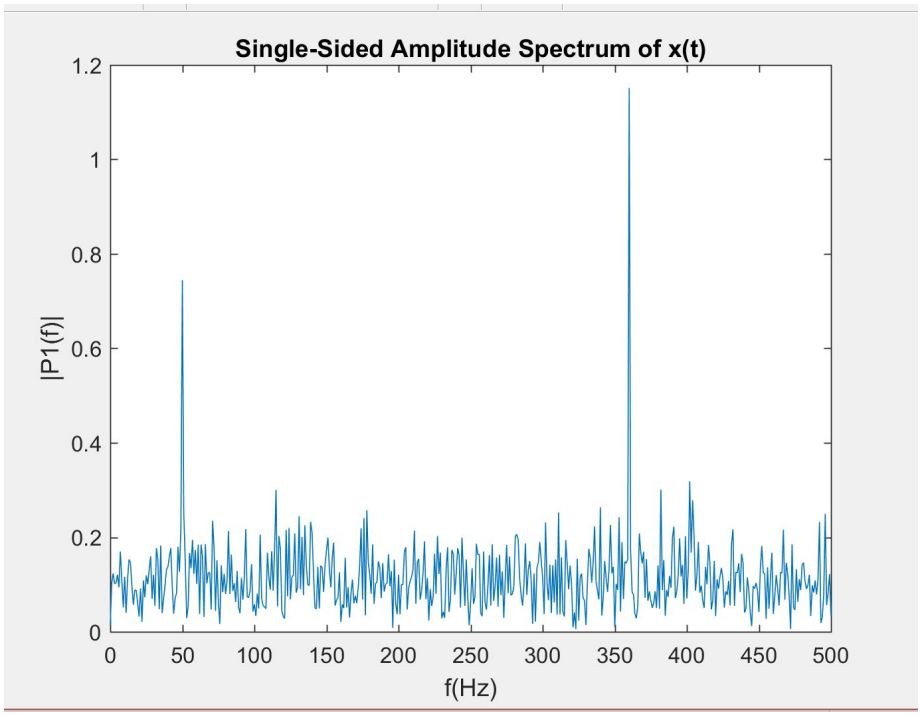
برای رسم این سیگنال نیز از دستورهای زیر در برنامه استفاده کردم:

```
Fs = 1000;
T = 1/Fs;
L = 1000;
t = (0: L-1)*T;


St = 0.7*sin(2*pi*50*t) + sin(2*pi*360*t);
Xt = St + 2*randn(size(t));

subplot(4, 1, 1);
plot(1000*t(1:50), Xt(1:50));
title('Noisy Signal');
xlabel('t(millisecond)');
ylabel('x(t)');
```

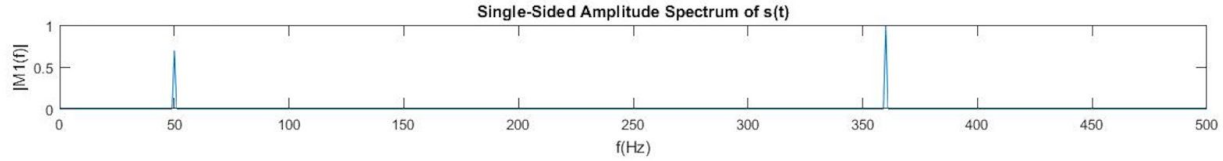
ب) طیف یک طرفه X را نیز به کمک دستورات صورت پروژه رسم کردم:



علت اینکه در ۵۰ هرتز به دقیقا ۰.۷ و در ۳۶۰ هرتز به دقیقا ۱ نمی‌رسیم این است که:

نویز موجود روی سیگنال است یعنی نویزی که به سیگنال اضافه کرده ایم روی مقادیر آن تاثیر می‌گذارد و باعث می‌شود از محاسبات دقیق ما دور شوند.

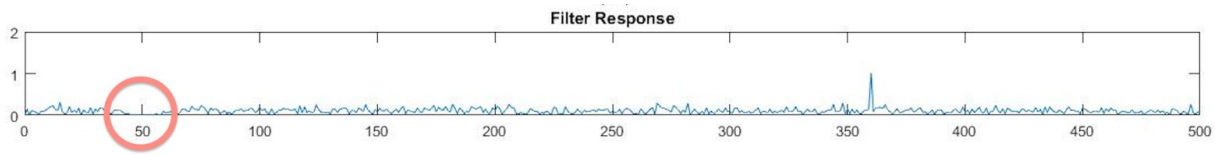
ج) مشابه بخش ب طیف یک طرفه‌ی سیگنال S را رسم می‌کنیم:



اگر بخواهیم این سیگنال را با خروجی بخش ب مقایسه کنیم می‌توانیم بگوییم:

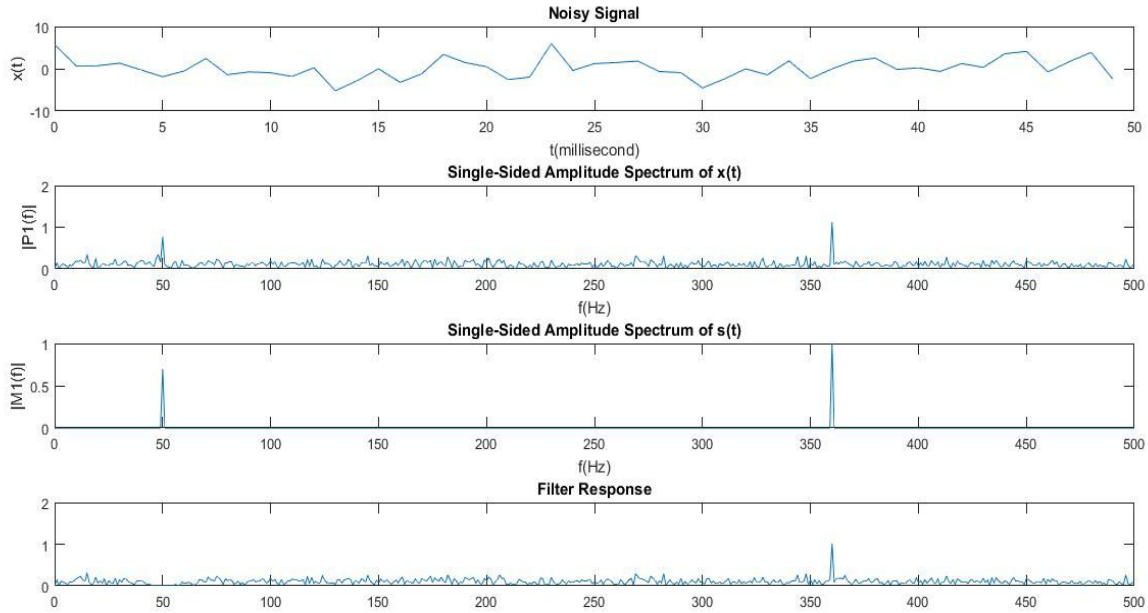
همان طور که می‌بینیم در اینجا فقط در دو فرکانس ۵۰ و ۳۶۰ هرتز مقدار غیر صفر داریم و در سایر فرکانس‌ها صفر را مشاهده می‌کنیم در صورتی که در طیف یک طرفه‌ی مربوط به سیگنال x چنین نبود و در سایر نقاط هم مقدار دارد علاوه بر این موضوع در فرکانس ۵۰ هرتز مقدار ۰.۷ و در فرکانس ۳۶۰ هرتز مقدار دقیقا یک است که برای x این مقادیر دقیق نبود و مقداری کمتر یا بیشتر بود. که این‌ها همه به دلیل نویز اضافه شده به سیگنال x است که روی این سیگنال وجود ندارد.

د) فیلتر را به صورت کدی که در پوشه ی کد ها موجود است تعریف کردم و سپس طیف یک طرفه‌ی سیگنال رد شده از فیلتر را رسم کردم:



و می‌توان مشاهده کرد که فرکانس ۵۰ هرتز حذف شده است.

تمامی خروجی های این بخش در کنار هم به شکل زیر اند:



کد فیلتر نیز به صورت زیر بود:

```
function Hd = Filter_Response
%FILTER_RESPONSE Returns a discrete-time filter object.

% MATLAB Code
% Generated by MATLAB(R) 9.0 and the Signal Processing Toolbox 7.2.
% Generated on: 10-Dec-2017 11:43:24

% Equiripple Bandstop filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 1000;  % Sampling Frequency

Fpass1 = 35;              % First Passband Frequency
Fstop1 = 49.5;              % First Stopband Frequency
Fstop2 = 50.5;            % Second Stopband Frequency
Fpass2 = 64;              % Second Passband Frequency
Dpass1 = 0.028774368332;  % First Passband Ripple
Dstop  = 0.001;           % Stopband Attenuation
Dpass2 = 0.057501127785;  % Second Passband Ripple
dens   = 20;              % Density Factor

% Calculate the order from the parameters using FIRPMORD.
[N, Fo, Ao, W] = firpmord([Fpass1 Fstop1 Fstop2 Fpass2]/(Fs/2), [1 0 ...
                          1], [Dpass1 Dstop Dpass2]);

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, Fo, Ao, W, {dens});
Hd = dfilt.dffir(b);

% [EOF]
```