Engineering Mathematic

Computer Assignment #1 (Fourier Analysis)

گزارش کار تمرین کامپیوتری اول

صادق حایری

۸۱۰۱۹۴۲۹۸

سوال ۱:

سیگنال $x[n]$ به شکل زیر را در نظر بگیرید:

$$x[n] = u[n - 200] - u[n - 400] + 0.05 \sin (2Pi/N \times 100 \times n) + 0.1 \cos (2Pi/N \times 50 \times n)$$

a) سیگنال را رسم کنید.

برای رسم ابتدا بازه خود را مشخص می‌کنیم که من آرایه‌ای از ۰ تا ۱۰۰۰ در نظر گرفته‌ام و دوره تناوب (N) را نیز ۱۰۰۰ در نظر گرفته ام.

```
n = (0:1000-1);
```

تابع خود را به صورت روبرو تعریف می‌کنیم:

```
X = heaviside(n-200) - heaviside(n-400) + 0.05 * sin( (2*pi)/N *
100 * n ) + 0.1 * cos( (2*pi)/N * 50 * n );
```

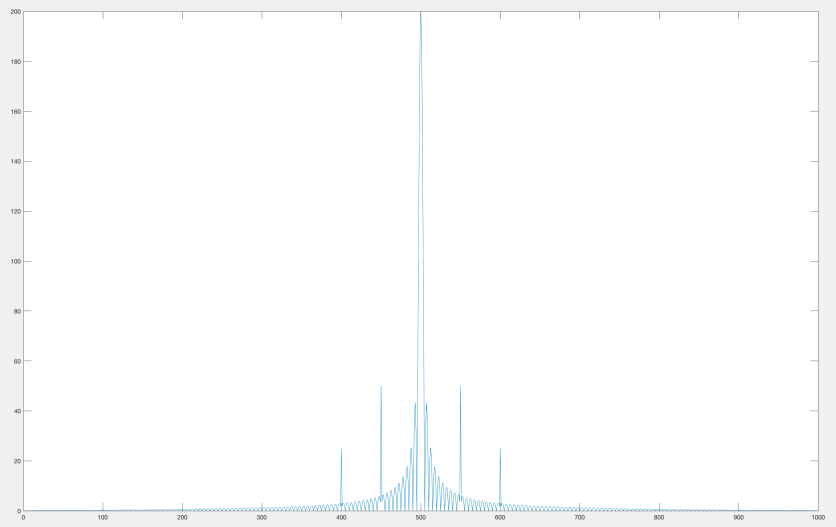
و با استفاده از دستور plot آنرا رسم می‌کنیم:

```
plot(n, X);
```

b, c) تبدیل فوریه سیگنال را با استفاده از "fft" بدست آورید. ، "fftshift" تبدیل فوریه بدست آمده را به فرم مناسب تغییر دهید.

```
fX = fftshift( fft(X) )
```

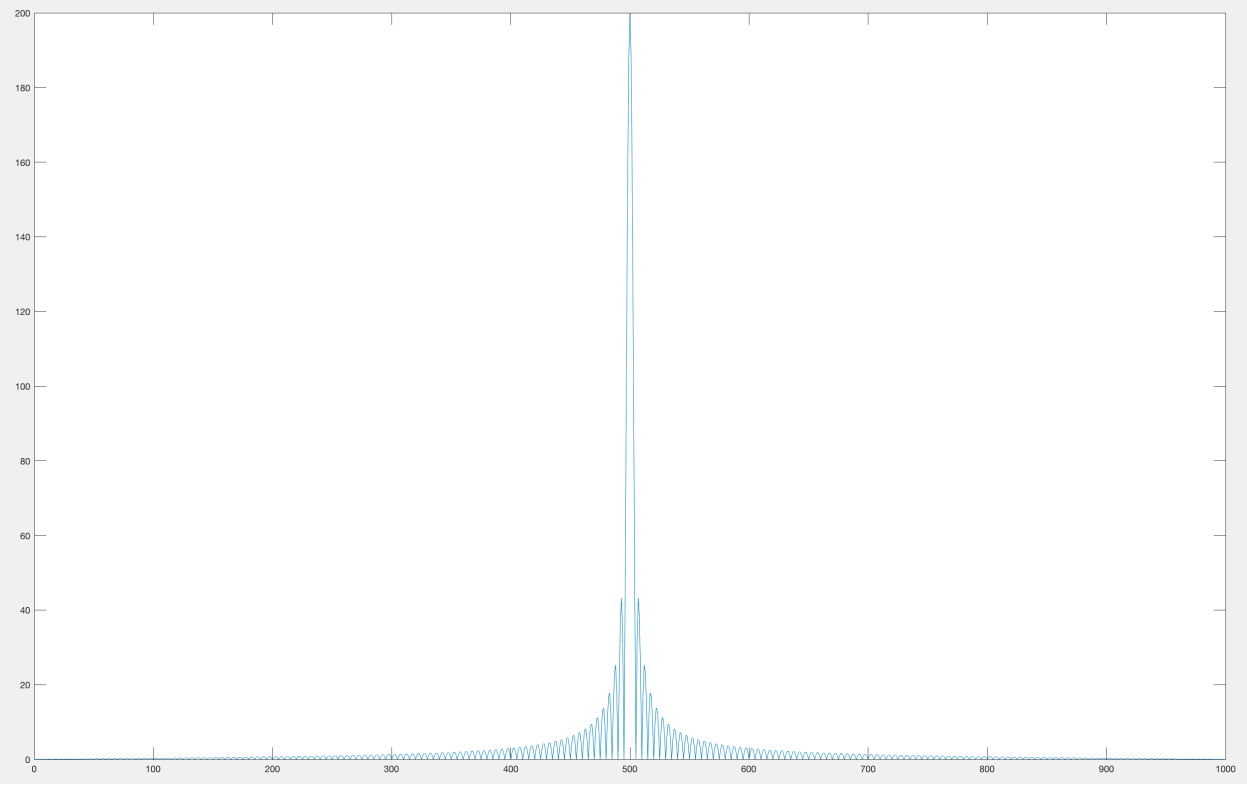
d) به روشی دلخواه و با استفاده از تبدیل فوریه بدست آمده، سیگنال را به گونه ای فیلتر کنید که نویز سیگنال حذف شود.



ابتدا با دستور plot قدرمطلق تابع تبدیل فوریه‌ی آنرا رسم می‌کنیم:

مشاهده میشود که در ۴ نقطه دارای نویز است (با توجه به اینکه تابع از سینوس و کسینوس با فرکانس متفاوت تشکلیل شده است قابل توجیه است)

برای از بین بردن نویزها از روش دستی استفاده می‌کنیم (!) و مقدار تابع در نقاط ۴۰۱، ۴۵۱، ۵۵۱ و ۶۰۱ را صفر می‌کنیم (این نقاط با تست و خطا به دست اند!) که نتیجه تصویر زیر می‌شود:

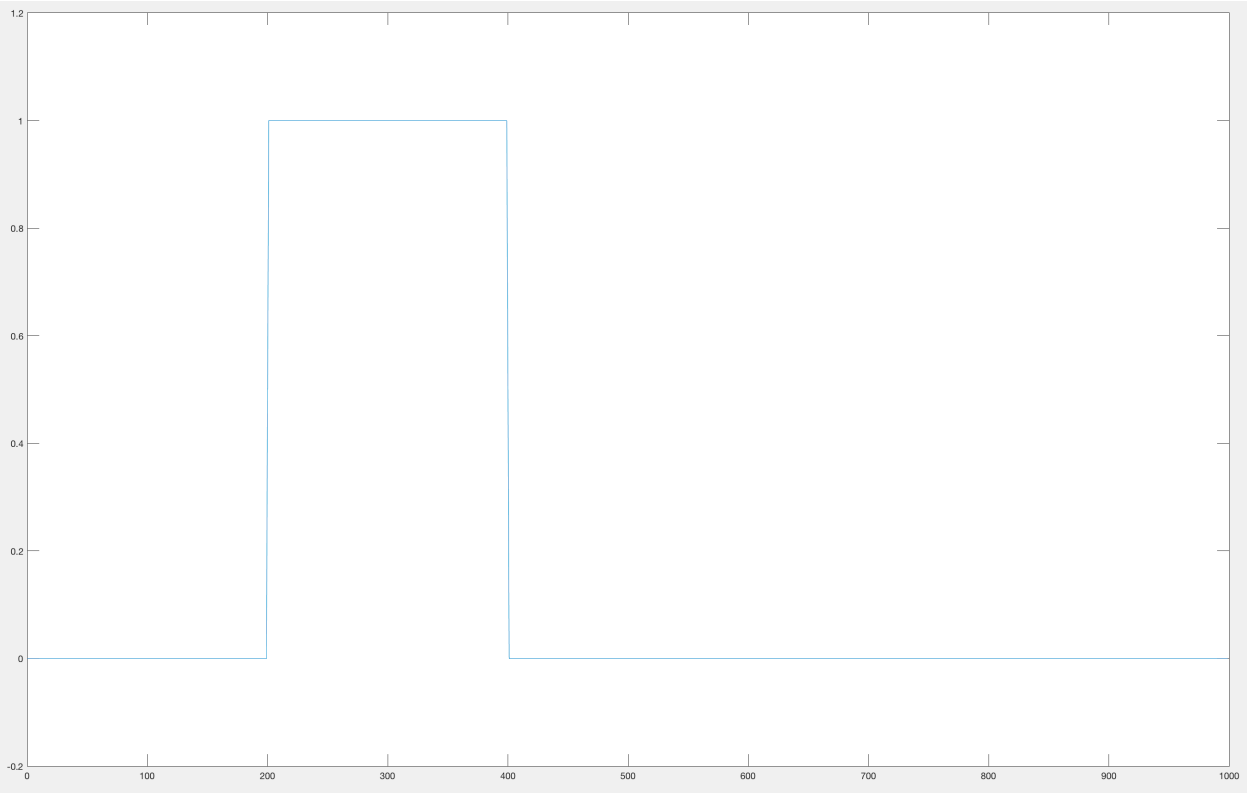


e) با استفاده از تابع "ifft" از تبدیل فوریه فیلتر شده، سیگنال فیلتر شده در حوزه زمان را بدست آورید و آن را ترسیم کنید.

تابع جدید را دوباره شیفت می‌دهیم و از این معکوس می‌گیریم و آنرا plot می‌کنیم:

```
iX = ifft( fftshift(fX) );

plot(n, iX);
```



f) نتایج را به تفصیل بررسی و تحلیل کنید.

...

سوال ۲:

فایل "voice_noisy.wav" موجود در پوشه پروژه را با استفاده از تابع "audioread" بخوانید. (در یک بردار ذخیره کنید.)

از دستور audioread برای خواندن فایل voice_noisy.wav استفاده میکنیم، مقدارهای برگشتی آن یک آرایه ۱۰۰۰۰۰۰تایی از اعداد double و همینطور فرکانس صدا یعنی ۴۴۱۰۰ هرتز است. تابع sound این دو مقدار را برای پخش صدا به عنوان آرگومان دریافت میکند.

```
[y,Fs] = audioread('voice_noisy.wav');
```

a) با استفاده از تابع "sound" سیگنال مربوطه را به صورت یک فایل صوتی پخش کنید.

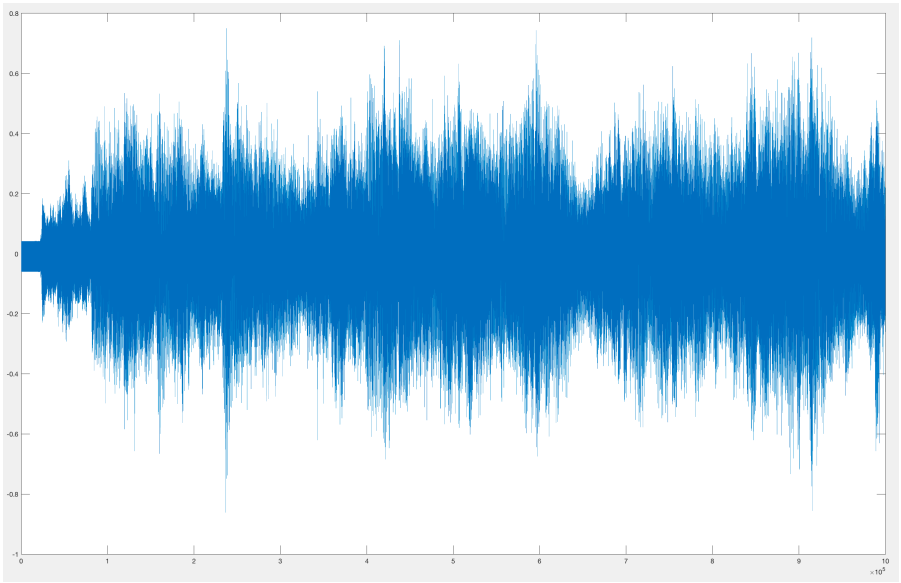
با استفاده از دستور sound و دادن آرایه و فرکانس، صوت پخش می‌شود!

```
sound(y,Fs);
```

b) سیگنال را ترسیم کنید.

برای اینکار آرایه‌ای از ۱ تا طول فایلمان (۱۰۰۰۰۰۰) میسازیم و با استفاده از آن صوت را plot می‌کنیم که نتیجه شکل زیر است:

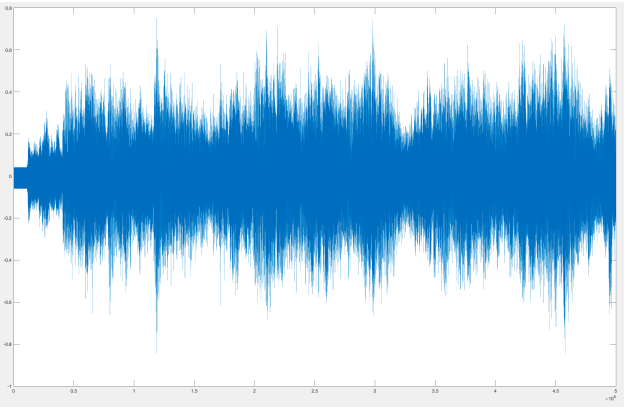
```
l = length(y);
n = (0:l-1);
plot(n, y);
```



c) می خواهیم سرعت پخش فایل صوتی را دو برابر کنیم. چه تغییری در سیگنال باید ایجاد کنیم؟ سیگنال مربوطه را ترسیم کنید و آن را پخش کنید.

برای اینکه سرعت فایل ما ۲برابر شود می‌توانیم داده‌های صوت‌مان را یک در میان حذف کنیم که با اینکار در زمان پخش صوت به نظر می‌آید که سرعت اجرای صوت ۲برابر شده است. برای اینکار آرایه‌ی جدید fast را تعریف می‌کنیم و داده‌های صوت‌مان را یکی در میان در آن میریزیم و آنرا پخش می‌کنیم (در شکل صوت تغییر زیادی مشاهده نمی‌شود ولی طول بازه‌ی آن نصف شده است)

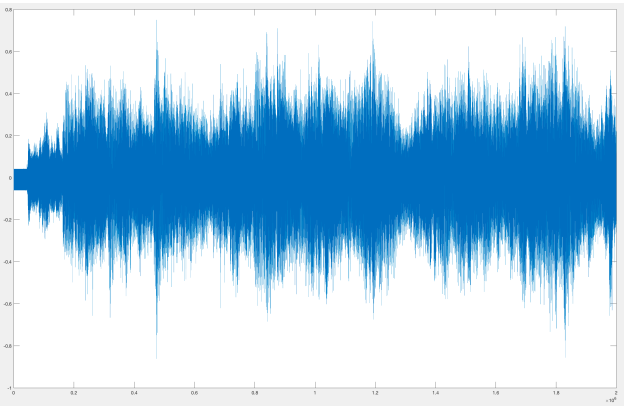
```
for R = 1:l
    if mod(R,2) == 0
        fast(end+1) = y(R);
    end
end
plot( (0:length(fast)-1) , fast);
sound(fast, Fs);
```



d) می خواهیم سرعت پخش فایل صوتی را نصف کنیم. چه تغییری در سیگنال باید ایجاد کنیم؟ سیگنال مربوطه را ترسیم کنید و آن را پخش کنید.

برای اینکه سرعت فایل ما نصف شود می‌توانیم داده‌های صوت‌مان را دو برابر کنیم که با اینکار در زمان پخش صوت به نظر می‌آید که سرعت اجرای صوت نصف شده است. برای اینکار آرایه‌ی جدید slow را تعریف می‌کنیم و داده‌های صوت‌مان ۲بار در آن میریزیم و آنرا پخش می‌کنیم (در شکل صوت تغییر زیادی مشاهده نمی‌شود ولی طول بازه‌ی آن دو برابر شده است)

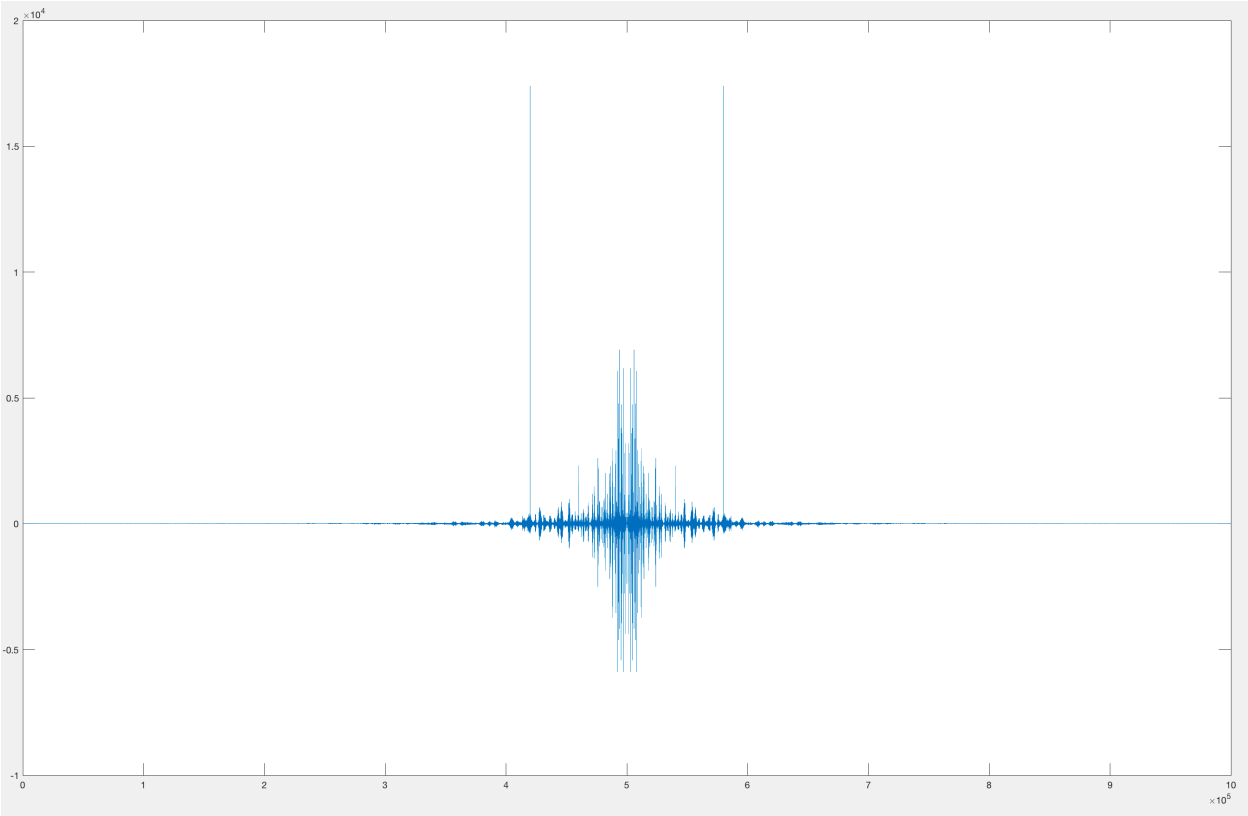
```
for R = 1:l
    slow(end+1) = y(R);
    slow(end+1) = 0;
end
plot( (0:length(slow)-1) , slow)
sound(slow, Fs);
```



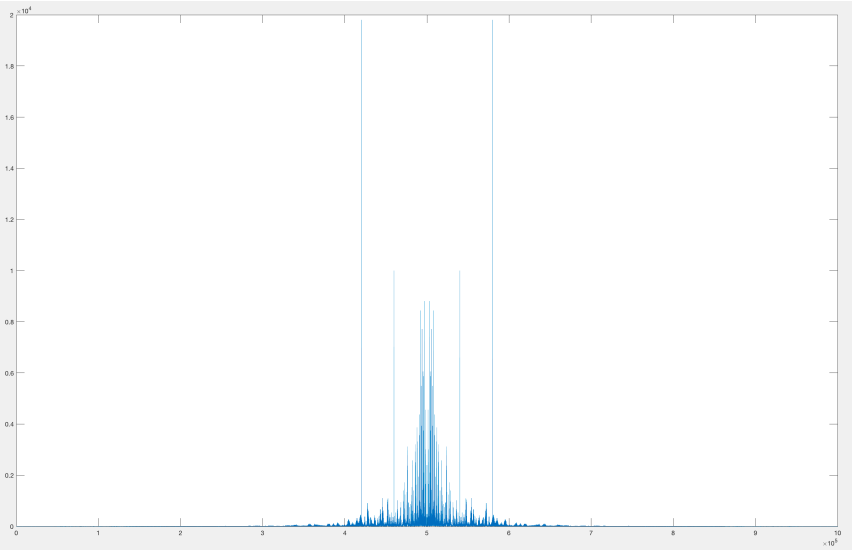
e & f) تبدیل فوریه سیگنال را با استفاده از "fft" بدست آورید و تبدیل فوریه بدست آمده را به فرم مناسب تغییر دهید. (یکی از روش های ممکن استفاده از تابع "fftshift" خواهد بود.)

```
f = fftshift( fft(y) );

plot( n, f );
```



مشاهده می‌شود که در فایل و در فرکانس پایین نویزهایی وجود دارند! برای مشاهده بهتر قدرمطلق آنرا رسم می‌کنیم:



g) حال تبدیل فوریه بدست آمده را به گونه ای فیلتر کنید تا نویز سیگنال حذف شود.

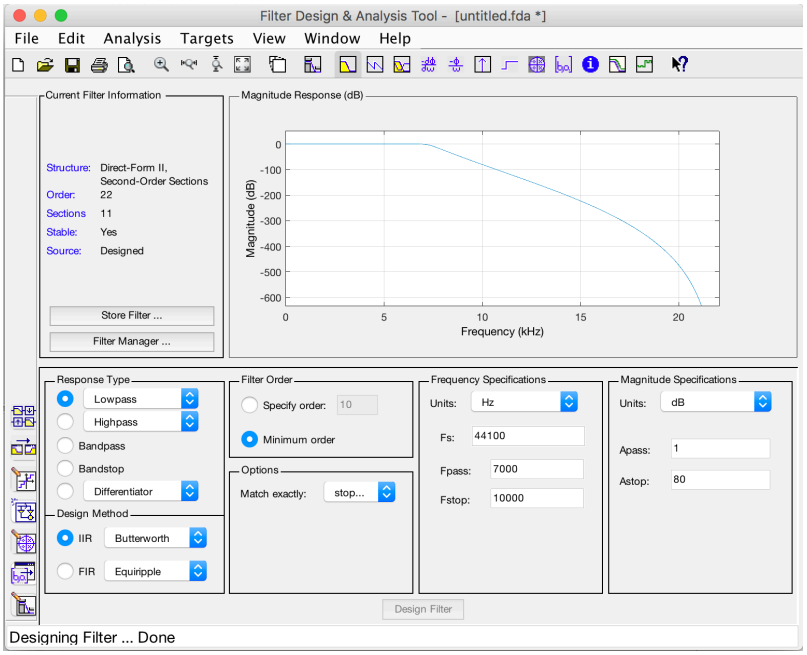
برای حذف نویز از ۲روش می‌توان استفاده کرد.

روش اول)روی آرایه حرکت می‌کنیم و اگر به داده‌ای رسیدیم که از مقدار خاصی (خودمان تعیین می‌کنیم که اینجا ۱۰۰۰۰ را درنظر گرفتیم) بزرگتر بود آنرا حذف می‌کنیم:

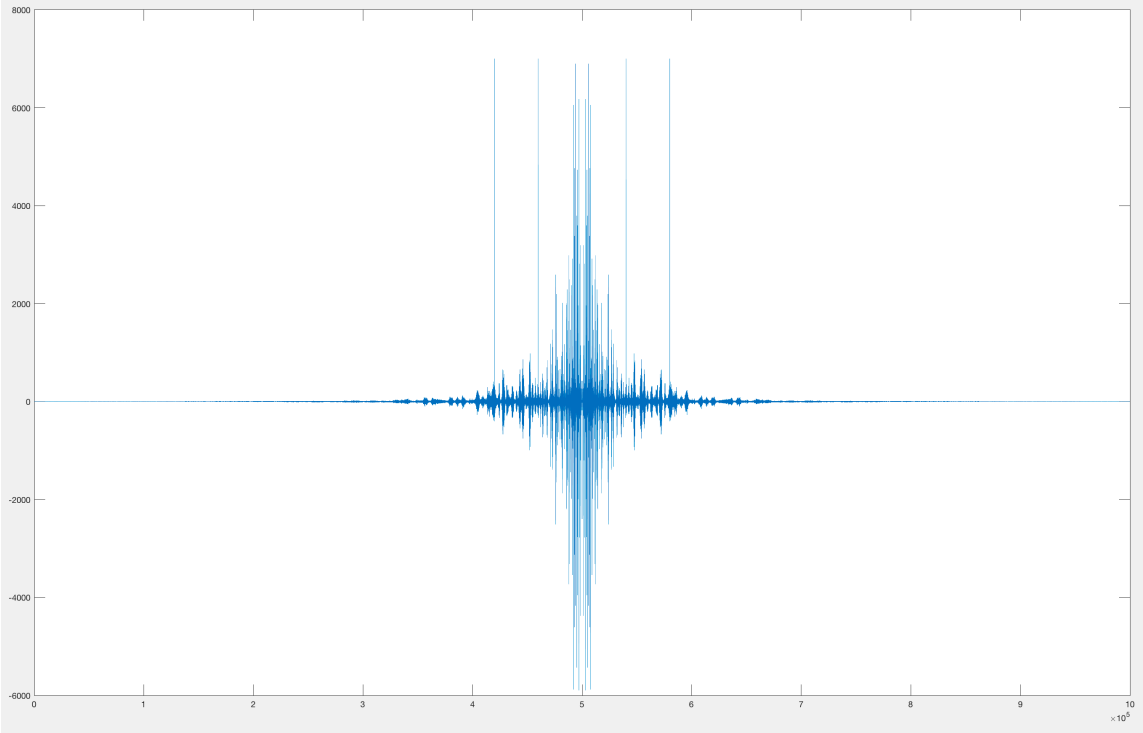
```
noisyLess = f;
for R = 1:l
    if absF(R) < 10000
        noisyLess(R) = f(R);
    else
        noisyLess(R) = 0.000;
    end
end
```

روش دوم)

با استفاده از دستور fdatool فیلتر پایین گذری برای حذف نویز تنظیم می‌کنیم و فرکانس قطع را همان ۱۰۰۰۰ انتخاب می‌کنیم و آنرا با نام myFilter ذخیره می‌کنیم.



با استفاده از دستور filter و دادن صوت و myFilter به آن صوت را از فیلتر عبور می‌دهیم، و با رسم تبدیل فوریه صوت مشاهده می‌شود که دامنه‌های بالای نویز حذف می‌شود:



h) با استفاده از عکس تبدیل فوریه، سیگنال فیلتر شده را بدست آورید و رسم کنید.

```
noisyLess = fftshift(noisyLess);

noisyLess = ifft( noisyLess );

plot( (0:length(noisyLess)-1), noisyLess );
```

i) سیگنال فیلتر شده را پخش کنید. آیا نویز حذف شده است؟

بله، نویز در هر دو روش از بین رفته است و می‌توان صدای موسیقی را شنید.

j) سیگنال بدون نویز را با استفاده از تابع "audiowrite" و در فرمت ".wav" ذخیره کنید.

```
audiowrite('noisyLess.wav', noisyLess, Fs);
```