

Engineering Mathematic

Computer Assignment #1 (Fourier Analysis)

گزارش کار تمرین کامپیوتروی اول

صادق حائری

۸۱۰۱۹۴۲۹۸

سوال ۱:

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

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

(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 تناب (N) را نیز ۱۰۰۰ در نظر گرفته‌ام.

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

تابع خود را به صورت رو برو تعریف می‌کنیم:

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

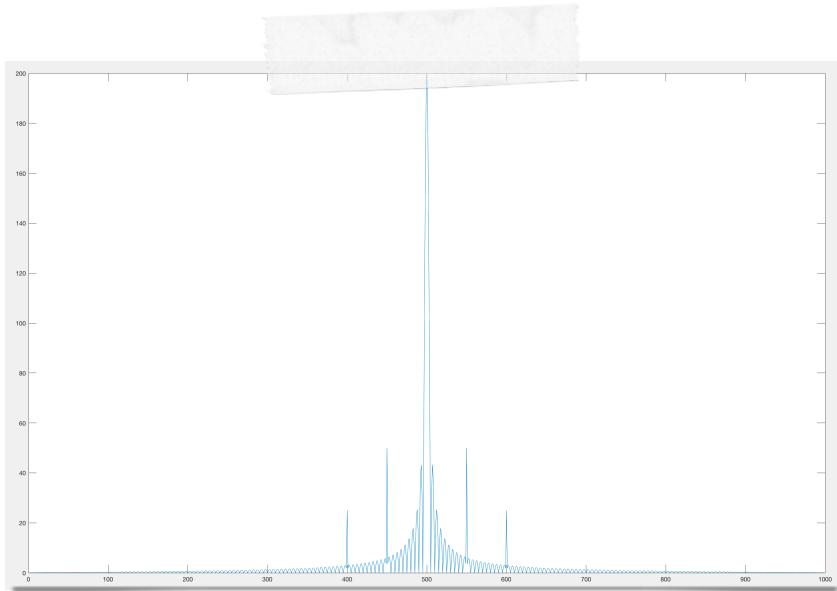
و با استفاده از دستور plot آنرا رسم می‌کنیم:

$$\text{plot}(n, X);$$

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

$$fX = \text{fftshift}(\text{fft}(X))$$

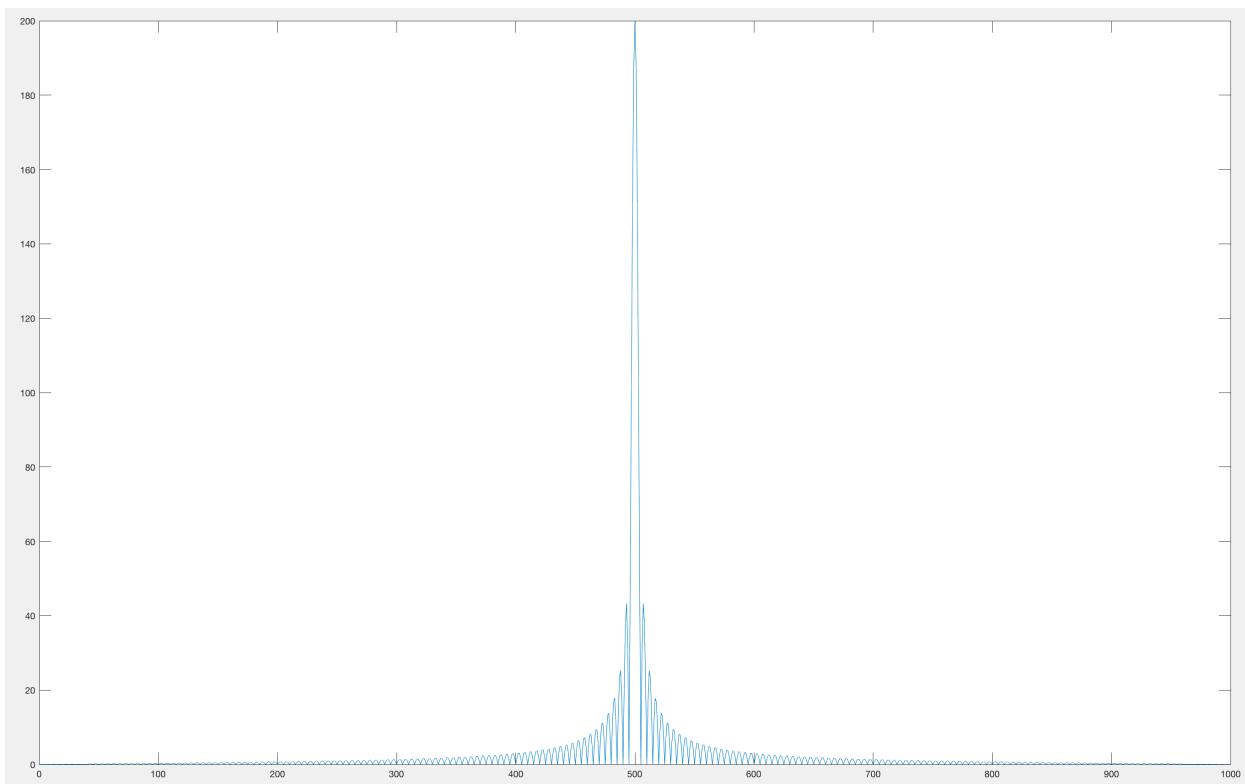
(d) به روشی دلخواه و با استفاده از تبدیل فوریه بدست آمده، سیگنال را به گونه‌ای فیلتر کنید که نویز سیگنال حذف شود.



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 تشكیل شده است قابل توجیه است)

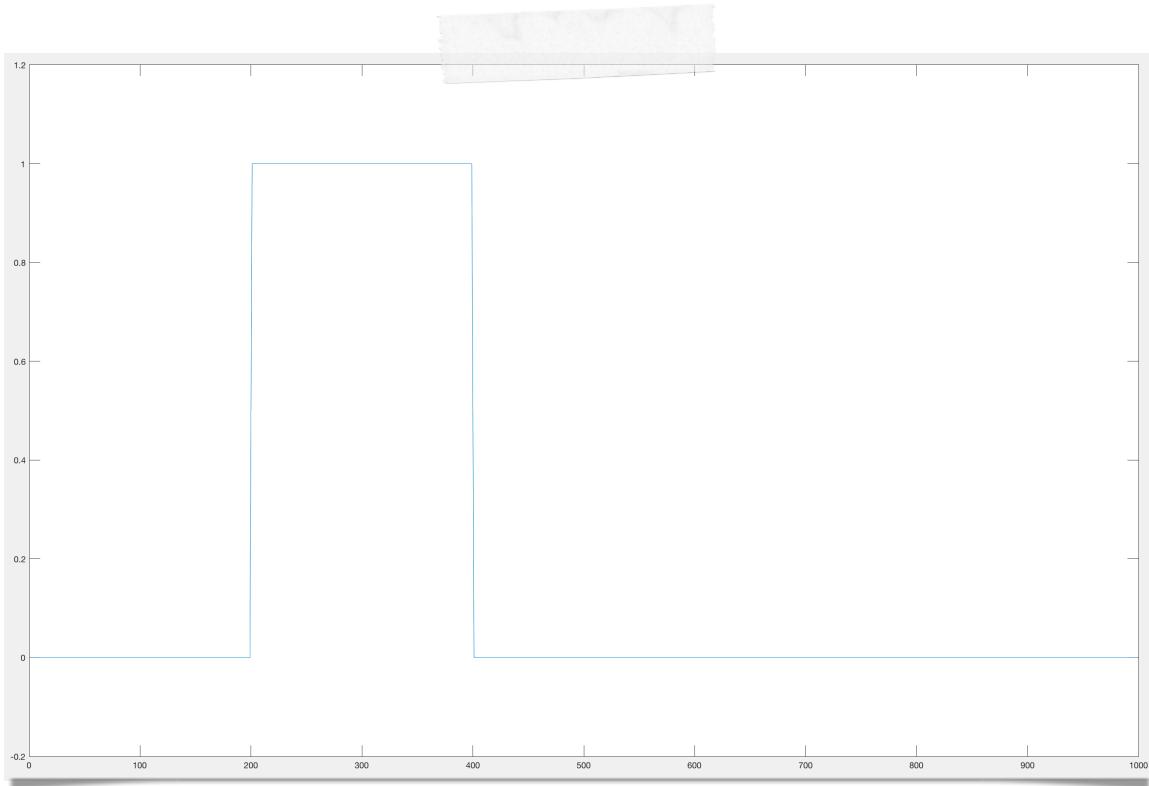
برای از بین بردن نویزها از روش دستی استفاده می‌کنیم (!) و مقدار تابع در نقاط ۴۰۱، ۴۵۱، ۵۵۱ و ۶۰۱ را صفر می‌کنیم (این نقاط با تست و خطابه دست اند!) که نتیجه تصویر زیر می‌شود:



(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.

ما با بردن موج به حوزه‌ی فرکانس توانستیم انواع فرکانس‌های موجود در موج را ببینیم و در ۴ نقطه دارای نوسان زیاد بود که با حذف آنها، آن سینوس و کسینوس‌هایی که آنجا بودند را حذف کردیم و زمانی که به حوزه‌ی زمان بازگشتم مشاهده می‌شود که نویزهای موج ما از بین رفته اند و ما سیگنال را بدون نویز می‌بینیم.

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 برگشته آن یک آرایه ۱۰۰۰۰۰۰۰۰ اتایی 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" سیگнал مربوطه را به صورت یک فایل صوتی پخش کنید.

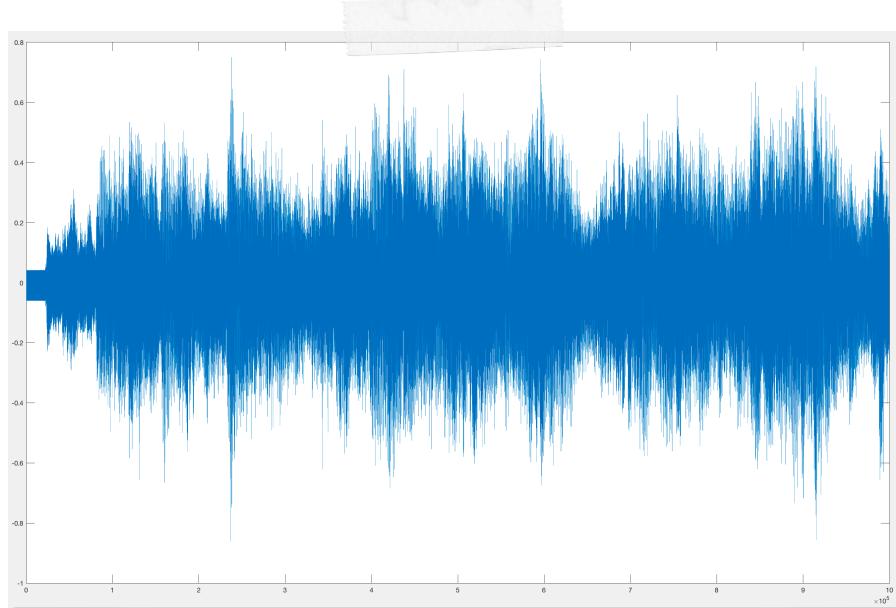
با استفاده از دستور sound و دادن آرایه و فرکانس، صوت پخش میشود!

```
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:

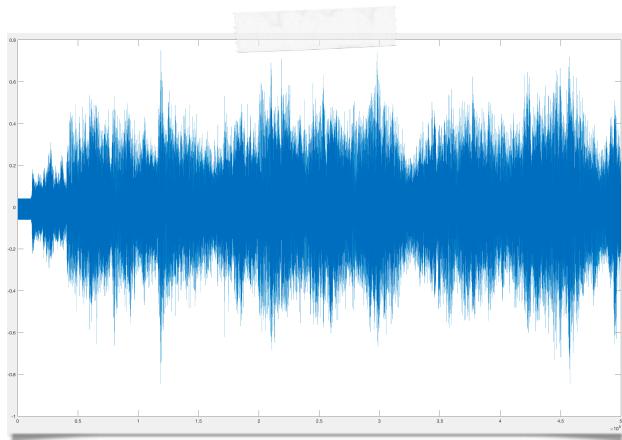
```
l = length(y);
n = (0:l-1);
plot(n, y);
```



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 ۲ برابر شود می‌توانیم داده‌های صوت‌مان را یک در میان حذف کنیم که با اینکار در زمان پخش صوت به نظر می‌آید که سرعت اجرای صوت ۲ برابر 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)

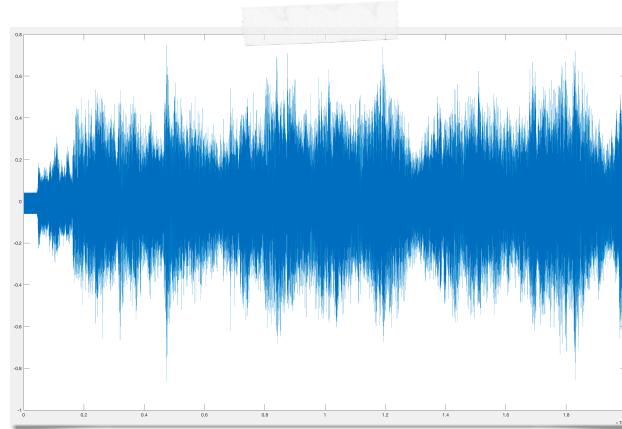
```
for R = 1:1
    if mod(R,2) == 0
        fast(end+1) = y(R);
    end
end
plot( 0:length(fast)-1 , fast );
sound(fast, Fs);
```



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 ۲ بار در آن میریزیم و آنرا پخش می‌کنیم (در شکل صوت تغییر زیادی مشاهده نمی‌شود ولی طول بازه‌ی آن دو برابر شده است)

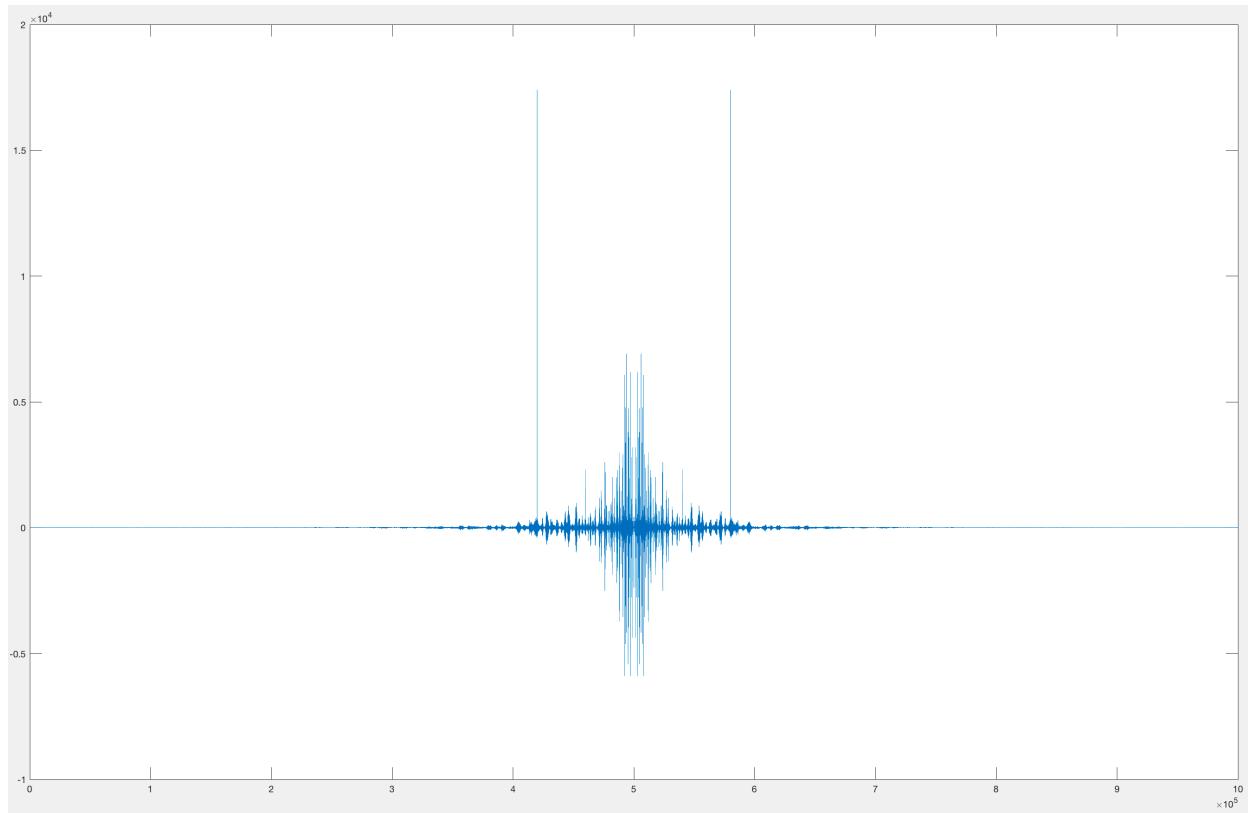
```
for R = 1:1
    slow(end+1) = y(R);
    slow(end+1) = 0;
end
plot( 0:length(slow)-1 , slow )
sound(slow, Fs);
```



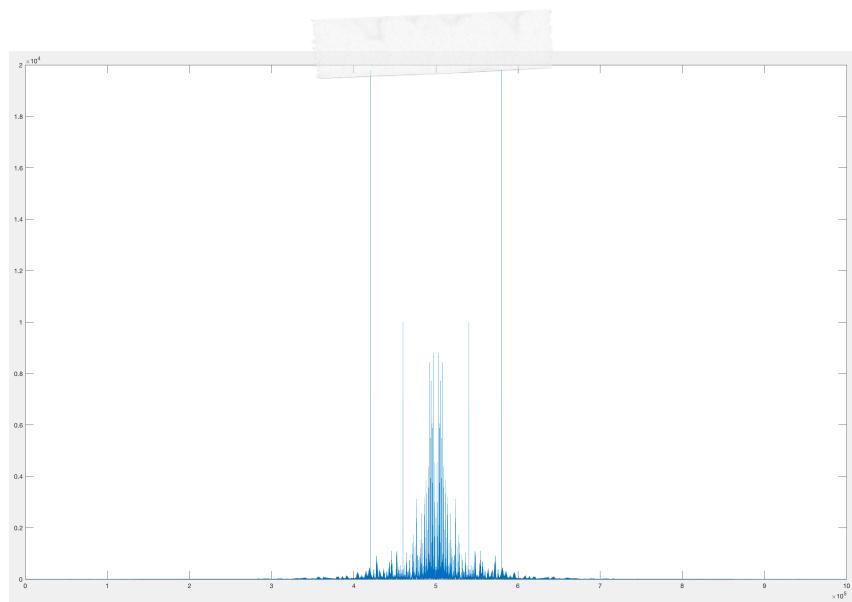
(e & f) تبدیل فوریه سیگنال را با استفاده از “fft” بدست آورید و تبدیل فوریه بدست آمده را به فرم مناسب تغییر دهید. (یکی از روش های ممکن استفاده از تابع “fftshift” خواهد بود.)

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

```
plot( n, f );
```



مشاهده می شود که در فایل و در فرکانس پایین نویزهای وجود دارند! برای مشاهده بهتر قدر مطلق آنرا رسم می کنیم:

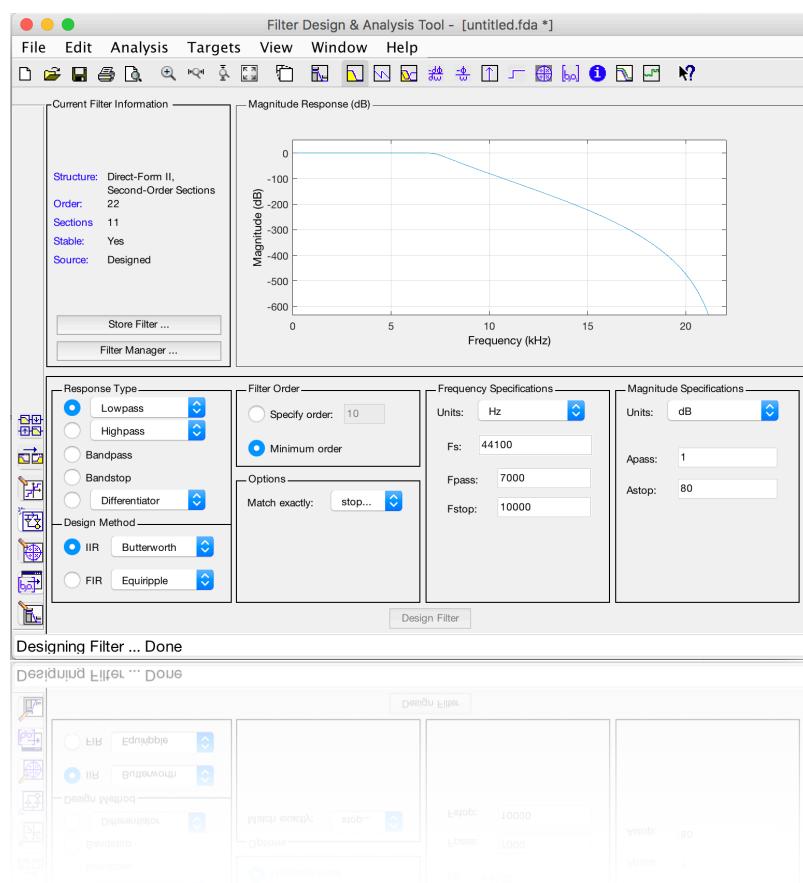


(g) حال تبدیل فوریه بدست آمده را به گونه ای فیلتر کنید تا نویز سیگنال حذف شود.

برای حذف نویز از ۲ روش می‌توان استفاده کرد.

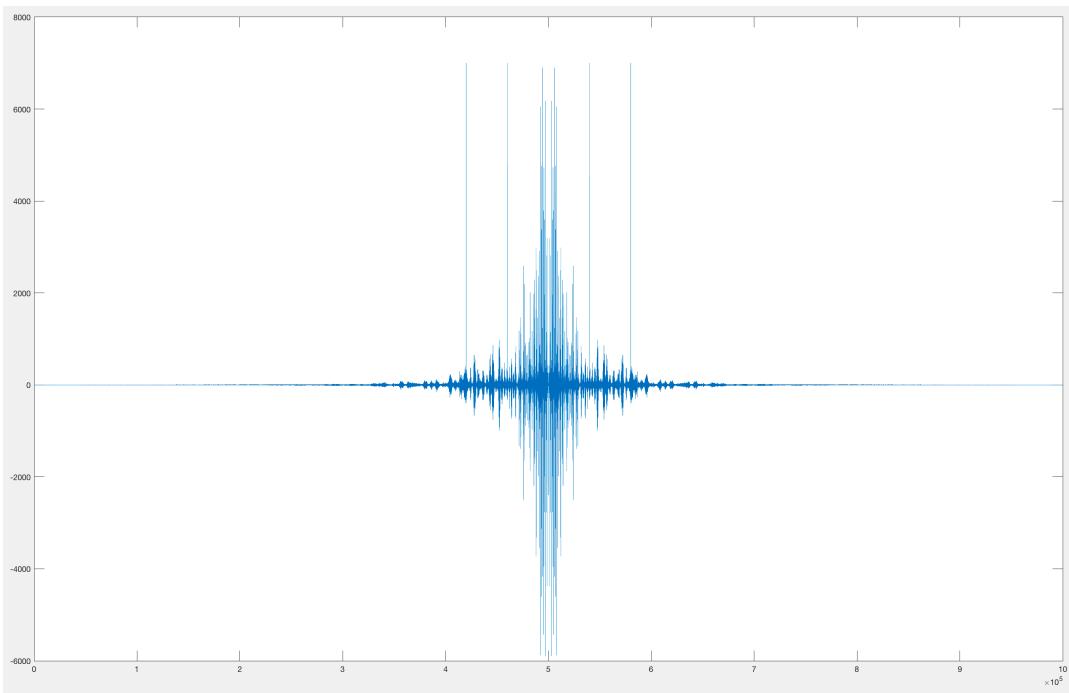
روش اول) روی آرایه حرکت می‌کنیم و اگر به داده‌ای رسیدیم که از مقدار خاصی (خودمان تعیین می‌کنیم که اینجا ۱۰۰۰۰) را درنظر گرفتیم) بزرگتر بود آنرا حذف می‌کنیم:

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



با استفاده از ابزار 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 تبدیل یافته‌ی صوت و myFilter به آن صوت را از فیلتر عبور می‌دهیم، با رسم نتیجه‌ی فیلتر مشاهده می‌شود که دامنه‌های بالای نویز حذف شده‌اند:



(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);
```

سوال ۳:

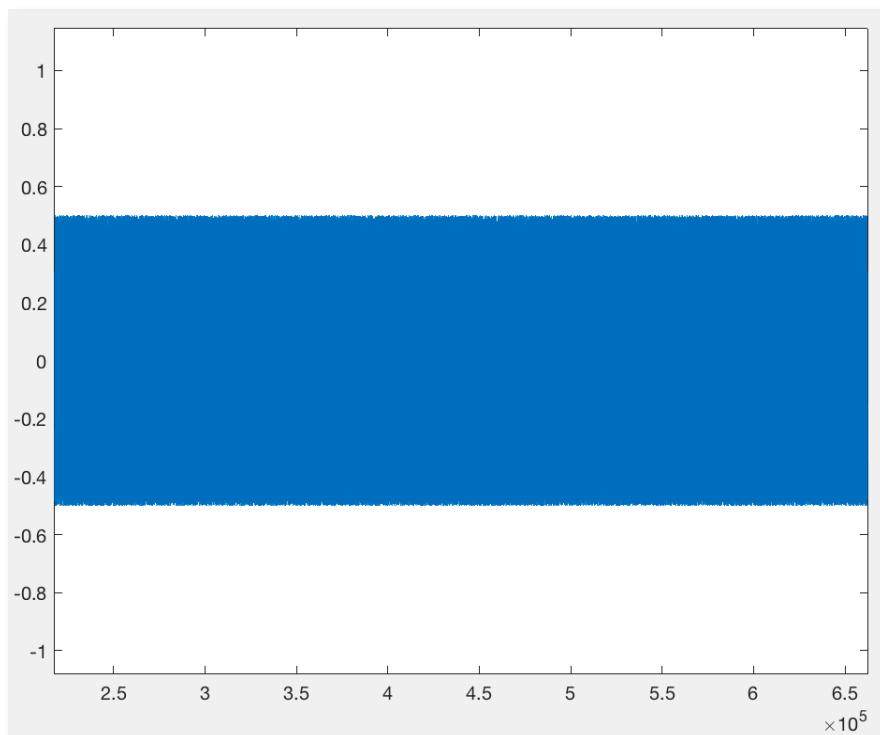
(سوال امتیازی) فایل صوتی فیلتر شده و بدون نویز بدست آمده در سوال ۲ را، با نویز سفید نویزی کنید و آن را پخش کنید و در نهایت نتیجه را ذخیره سازی کنید.

“white noise is a random signal having equal intensity at different frequencies”

با توجه به تعریف نویز سفید (منبع: ویکیپدیا) برای ساخت نویز سفید تنها کافیست آرایه‌ی خودمان را از اعداد رندم پر کنیم برای اینکار از اعداد رندم بین $5.-5.+$ استفاده می‌کنیم و نویز سفیدی به طول صوتمان ایجاد می‌کنیم:

```
whiteNoise = y;  
for R = 1:l  
    whiteNoise(R) = rand - .5;  
end
```

که نویز به صورت زیر خواهد بود و در صورت پخش آن صدای آشنای برفک تلویزون را می‌شنویم!



حال باید صدای بدون نویز قبلی را با این صدا ترکیب کنیم، برای اینکار دو آرایه را با هم جمع می‌زنیم تا هر درایه‌ی آن از جمع درایه‌های دو صوت تشکیل شود.

```
whiteNoise = whiteNoise + noisyLess;
```

که نتیجه شکل زیر می‌شود که با پخش آن به درستی روش مان پی می‌بریم!

