به نام خدا

گزارش پروژه ی آزمایشی درس سیگنال و سیستم ها

نام استاد: جناب آقای دکتر رحمتی

نام دانشجو: فرشید نوشی

شماره ی دانشجویی: 9831068

گزارش

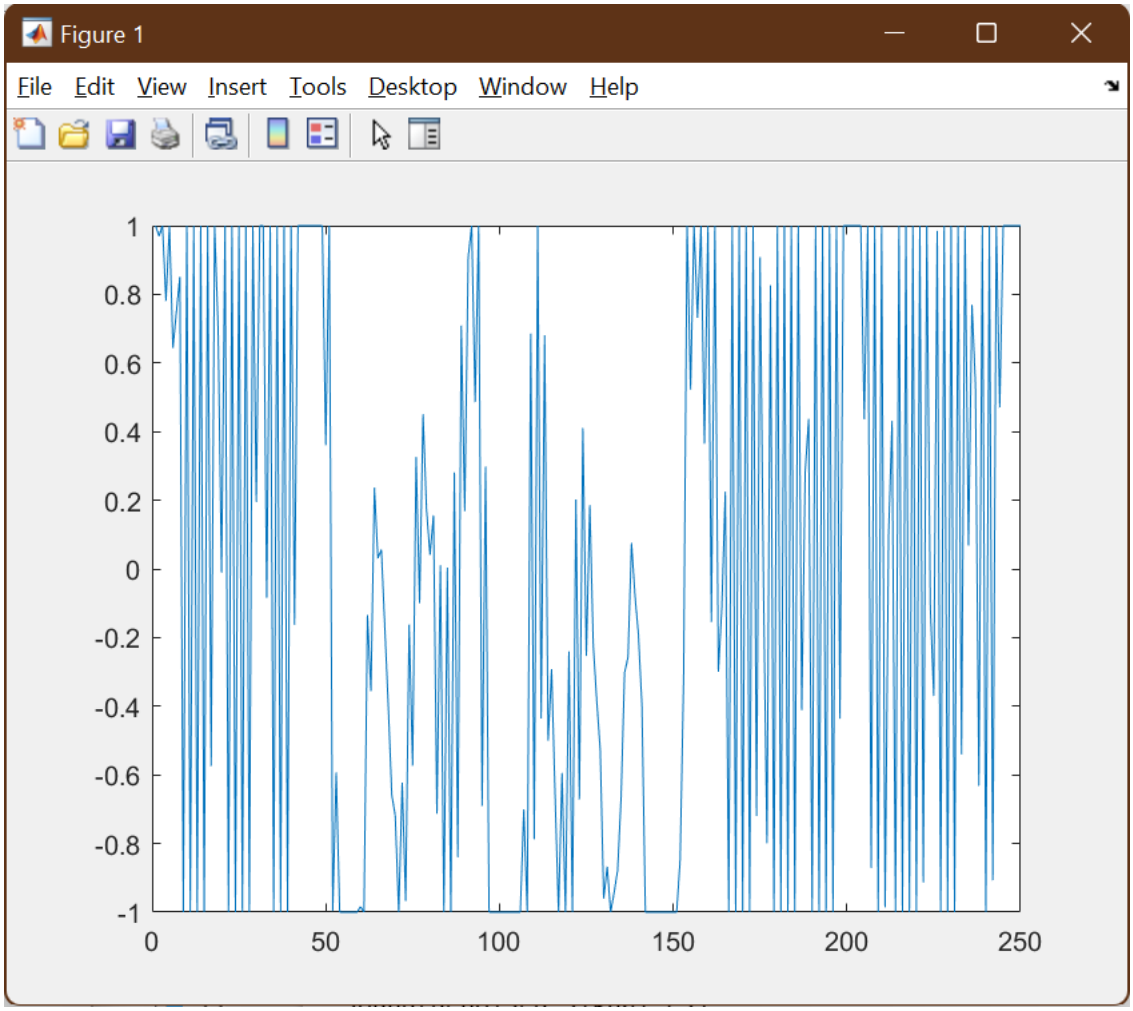
با توجه به صورت پروژه اقدام به پیاده سازی دستور پروژه در دو فایل جداگانه ی MATLAB شد. که هر دو در فایل ارسالی قرار گرفته اند.

گزارش فاز اول

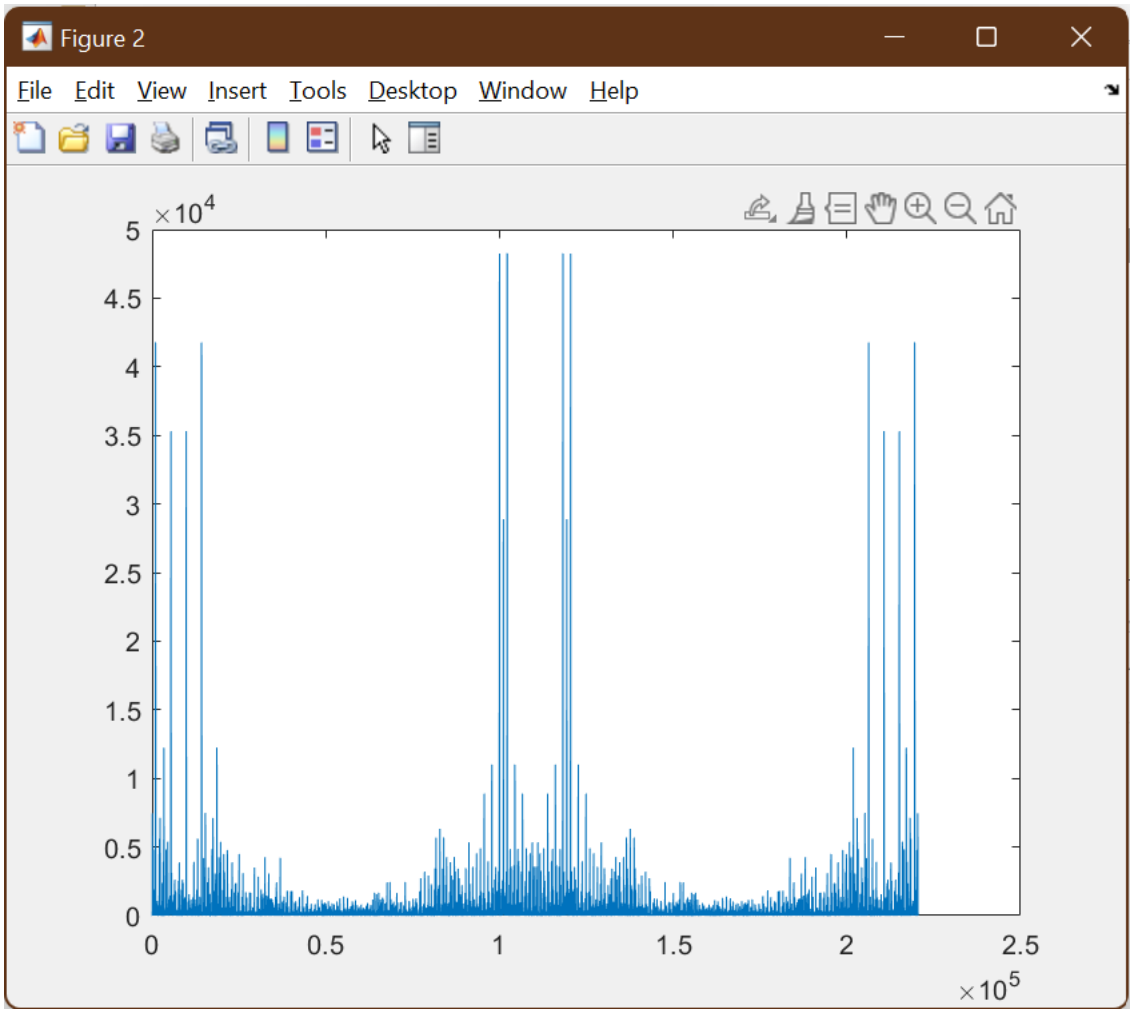
```
%% Phase one ==============================
[phase1sample,Fs]=audioread('phase1sample.wav');
% plot signal
figure
plot(phase1sample(1:250))
% fourier transform
figure
f_phase1sample = abs(fft(phase1sample));
plot(f_phase1sample)
% filter the noise
figure
b = 14000 ;
f_phase1sample = f_phase1sample .* [ones(b,1);zeros(length(f_phase1s
plot(f_phase1sample)
% ifft and play signal
figure
denoised_signal = abs(ifft(f_phase1sample));
sound(denoised_signal,Fs);
audiowrite('denoised_signal.wav',denoised_signal,Fs);
plot(denoised_signal(1:250))
```

در این قسمت در خط اول اقدام به خواندن فایل صوتی و فرکانس نمونه برداری میکنیم.

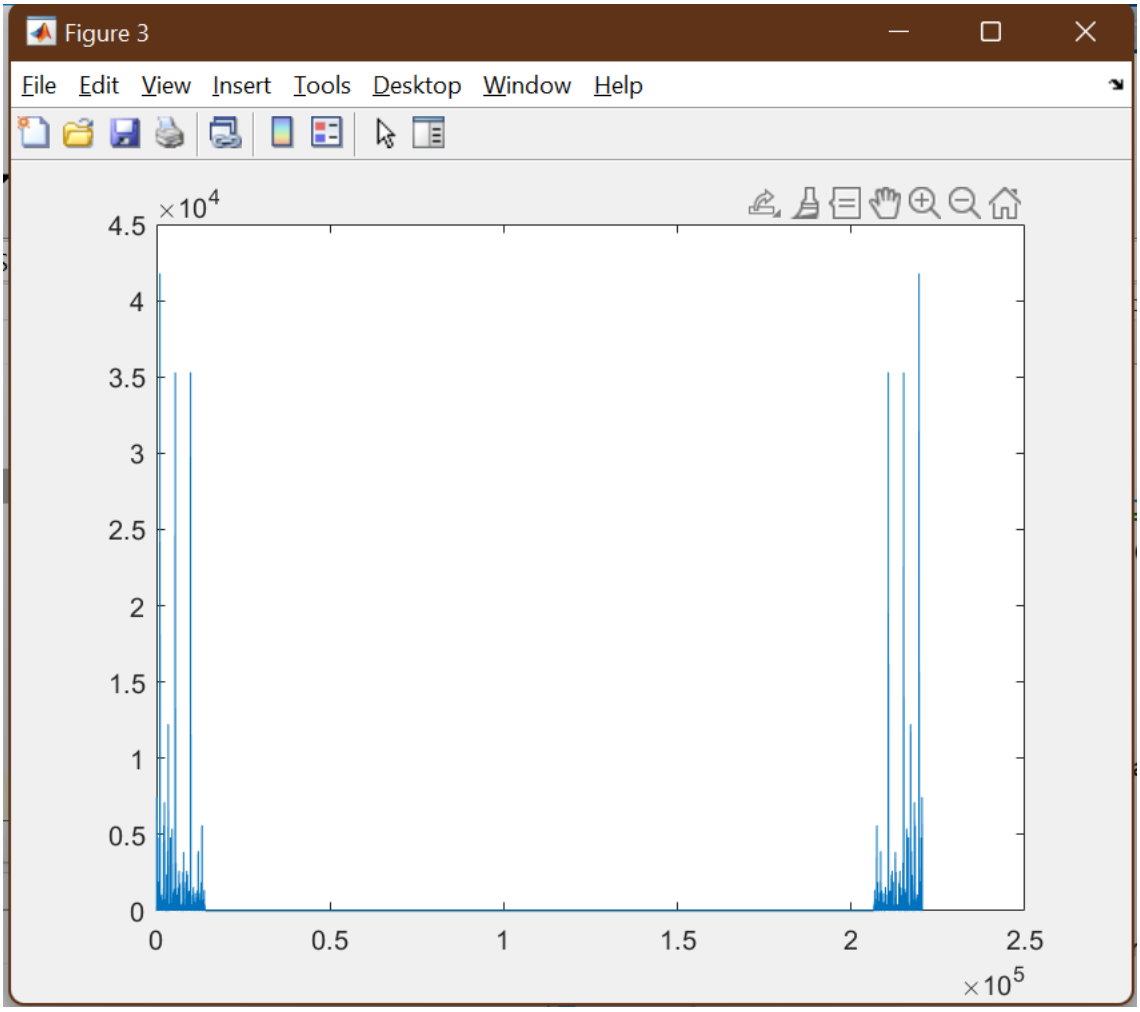
در ادامه در یک شکل فایل صوتی خوانده شده مان را در فاصله ی یک تا 250 را به روی نمودار میبریم تا نمایش داده شود.



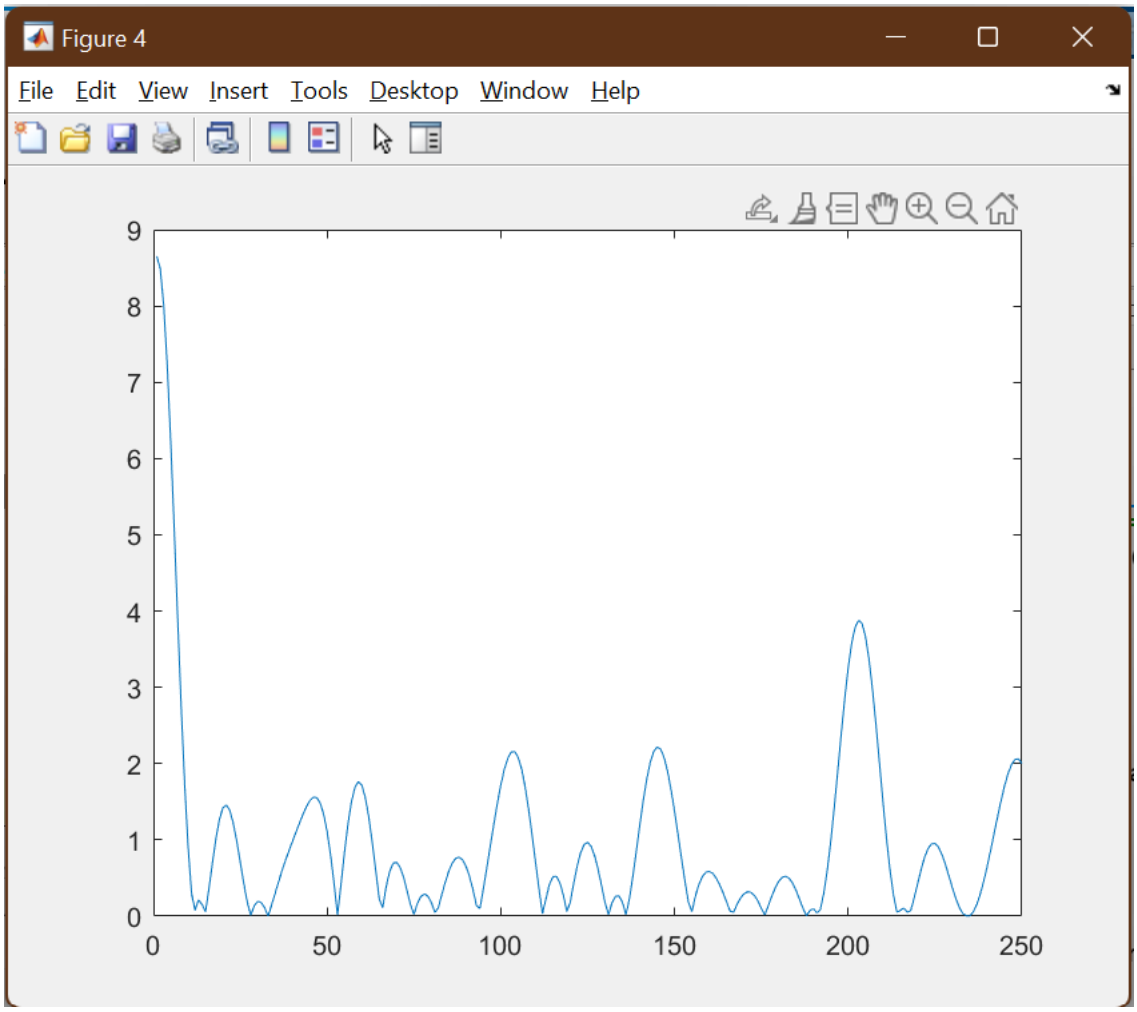
در نمودار بالا بدون هیچ تغییری تا کنون فایل صوتی را نشان داده ایم. در ادامه با استفاده از تابع fft تبدیل فوریه سیگنال مان را بدست آوردیم و آن را در شکلی دیگر در حوزه ی فرکانس نمایش داده ایم.



که همانطور که در شکل مشاهده میکنید بر روی صفحه پلات شده است. پس از آزمون و خطا و بررسی صوت های خروجی در خروجی برنامه، آستانه ی فرکانسی 14000 را برای فیلتر کردن قرار دادیم و فرکانس های بالا تر از ان را از سمت صفر حذف نمودیم، هم چنین فرکانس های بالای نمودار(14000 فرکانس بالا را نیز حفظ کردیم) در نهایت نمودار فیلتر شده ی ما به صورت شکل زیر در آمده است.(اینکار به طریق ضربی که در خط 17 کد میبینید میان ضرب دو ماتریسی که ماتریس دوم خاصیت فیلتر کنندگی دارد انجام شده است)



که همانطور که میبینید فرکانس های دو قسمت پایینی و بالایی در آن باقی ماندند و باقی فرکانس ها فیلتر شدند. در قسمت آخر کد به ترتیب با تابع معکوس تبدیل فوریه(ifft) تبدیل معکوس فوریه ی سیگنال تبدیل شده و فیلتر شده ی مرحله ی قبل مان را میدهیم و با پخش صدا(sound) آن را پخش کردیم و با ذخیره ی صدا (audiowrite) آن را در یک فایل با نام denoised_signal ذخیره کردیم و در نهایت با نشان دادن سیگنال denoised شده بر روی محور اولیه اش با همان طول دفعه ی اول تاثیر فیلترینگمان بر سیگنال ورودی مشهود است.



تصویر نهایی بعد از اعمال تغییرات

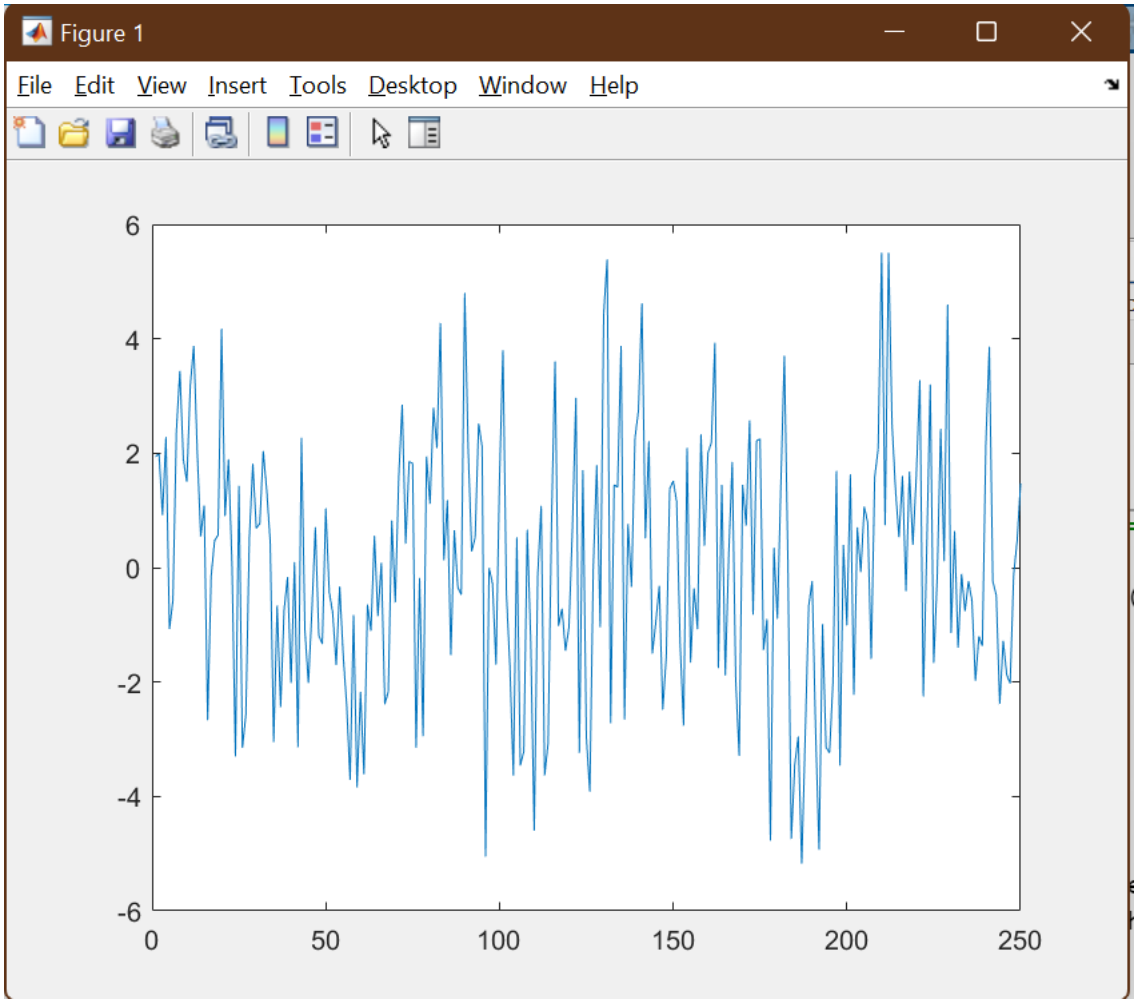
گزارش فاز دوم

```
%% Phase two ==============================

[phase2sample,Fs]=audioread('phase2sample.wav');

% plot signal
figure
plot(phase2sample(1:250))

% fourier transform
figure
fourier_signal=abs(fft(phase2sample));
plot(fourier_signal(1:length(fourier_signal)/2+1))

% denoise
figure
b = 300;
fourier_signal(fourier_signal < b) = 0;
plot(fourier_signal)

% inverse fourier
figure
output_signal=abs(ifft([fourier_signal fourier_signal]));
plot(output_signal(1:250))

```

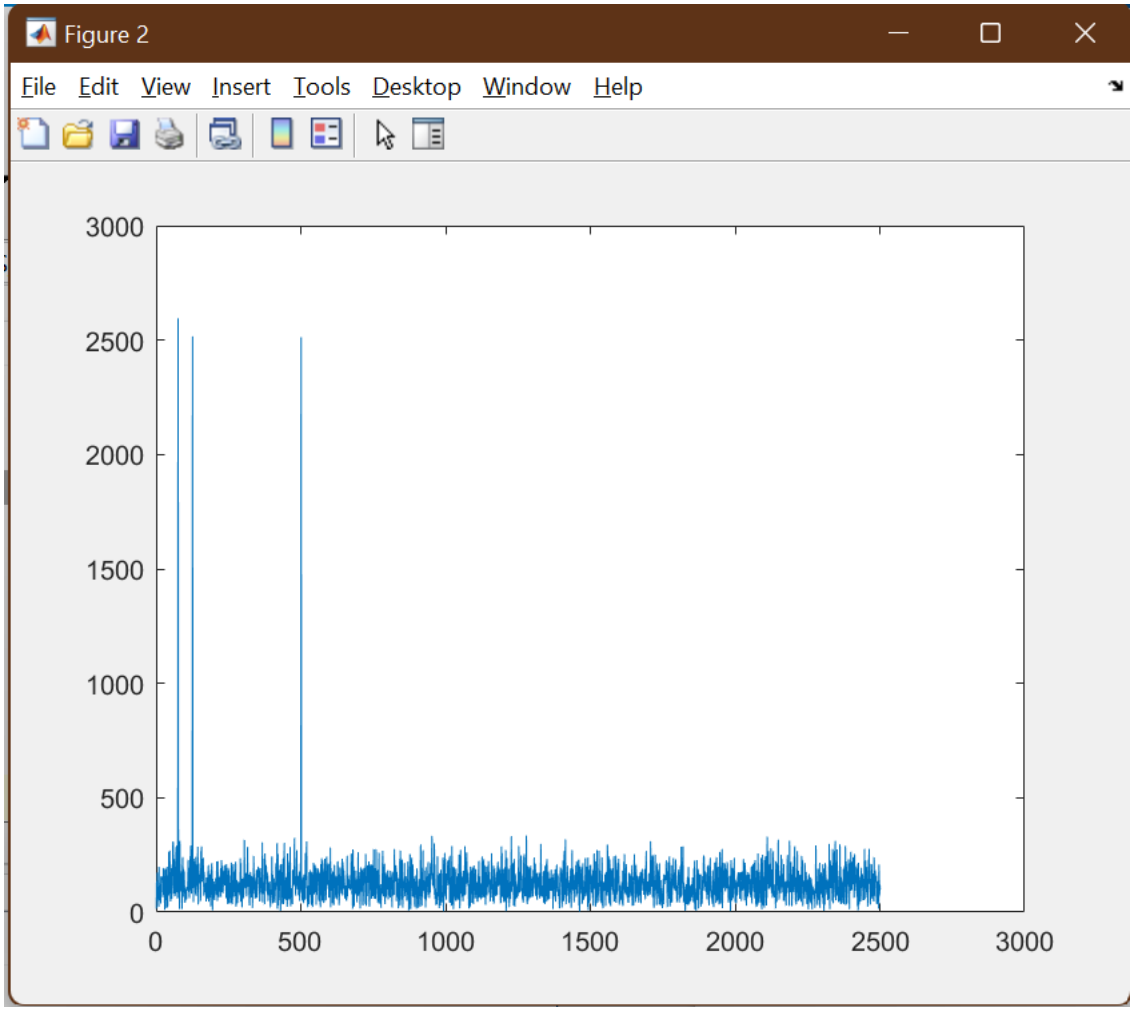
```
% save
audiowrite('second_phase_first_denoised_signal.wav',output_signal(2:

% fourier transform
figure
fourier_signal_2=abs(fft(output_signal));
plot(fourier_signal_2(2:length(fourier_signal_2)/2+1))

% under 1200
under_1200_audio = abs(ifft(equalizer_function(phase2sample,Fs,[10,1
audiowrite('under1200.wav',under_1200_audio,Fs)

% above 2000
above_2000_audio = abs(ifft(equalizer_function(phase2sample,Fs,[0.1,
audiowrite('above2000.wav',above_2000_audio,Fs)

%% Function:
function [fourier_signal,f] = equalizer_function(signal,Fs,amplify_c
    fourier_signal=pwelch(signal,[],[],[],Fs);
    f=1:Fs/2;
    for i = 1:length(fourier_signal)
        if(f(i) >= 20 && f(i) < 50)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(1);
        elseif (f(i) >= 50 && f(i) < 100)
```

```
    for i = 1:length(fourier_signal)
        if(f(i) >= 20 && f(i) < 50)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(1);
        elseif (f(i) >= 50 && f(i) < 100)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(2);
        elseif (f(i) >= 100 && f(i) < 200)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(3);
        elseif (f(i) >= 200 && f(i) < 500)
             fourier_signal(i)=fourier_signal(i)*amplify_cof(4);
         elseif (f(i) >= 500 && f(i) < 1000)
             fourier_signal(i)=fourier_signal(i)*amplify_cof(5);
        elseif (f(i) >= 1000 && f(i) < 2000)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(6);
        elseif (f(i) >= 2000 && f(i) < 4000)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(7);
         elseif (f(i) >= 4000 && f(i) < 8000)
             fourier_signal(i)=fourier_signal(i)*amplify_cof(8);
        elseif (f(i) >= 8000 && f(i) < 12000)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(9);
         elseif (f(i) >= 12000 && f(i) < 20000)
            fourier_signal(i)=fourier_signal(i)*amplify_cof(10);
        end
    end
end
```

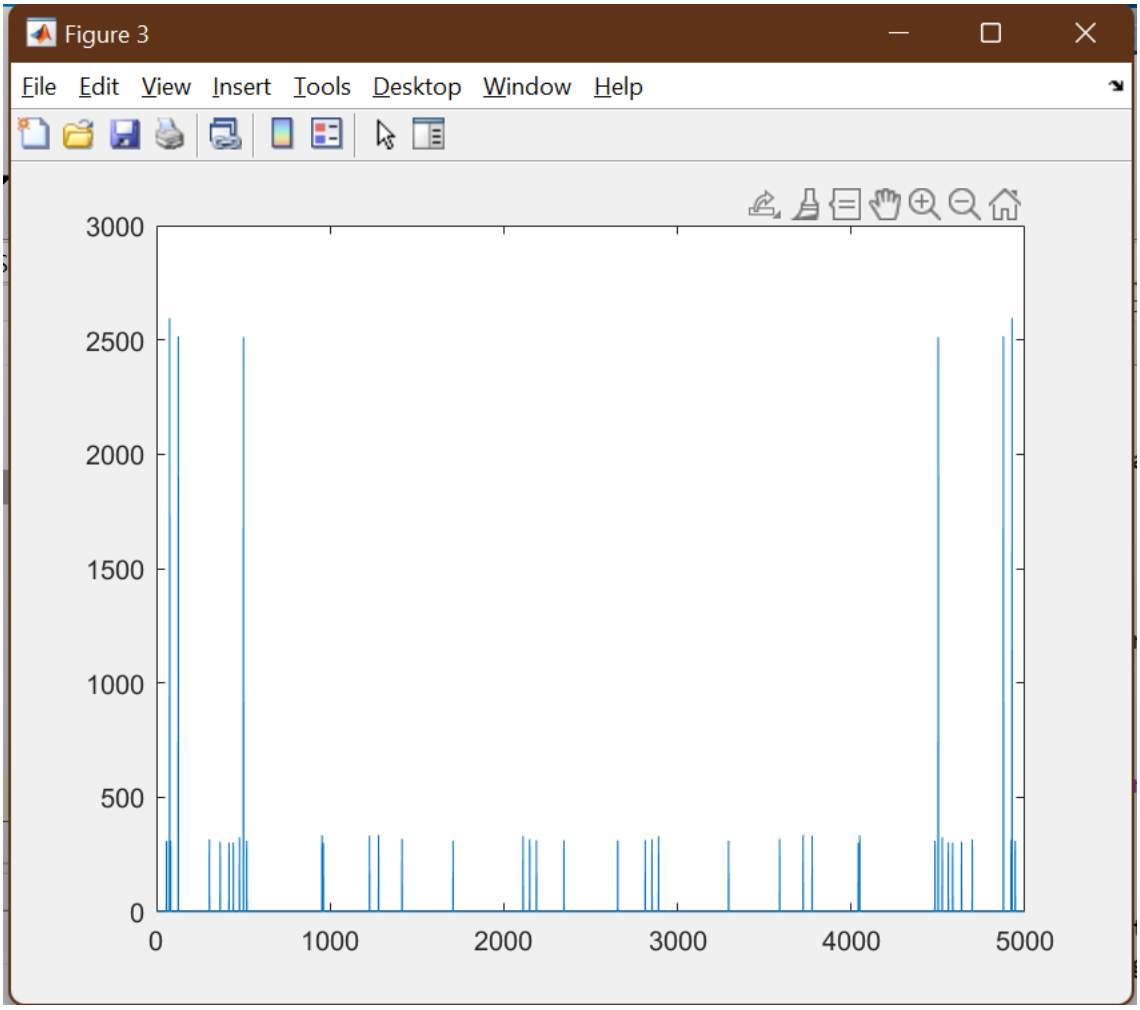
در ابتدای برنامه ی این بخش همانند قسمت قبل با خواندن سیگنال و نمایش آن روی نمودار برنامه آغاز میشود.



در ادامه ی برنامه با استفاده از توابع MATLAB تبدیل فوریه ی سیگنال ورودی را بدست می آوریم و خروجی اش را plot میکنیم.

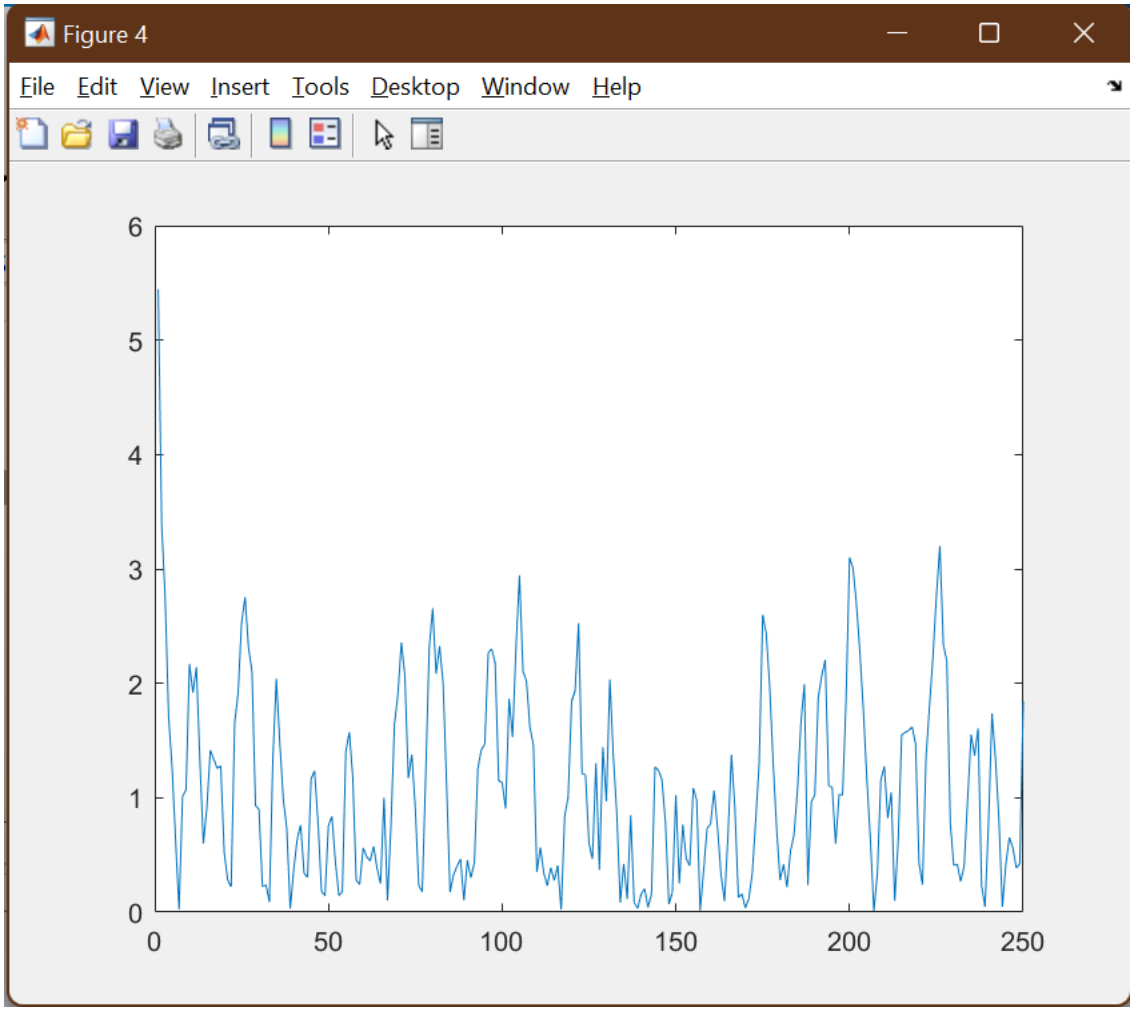


همانطور که میبینیم تعداد بسیاری سیگنال پالس ضربه با اندازه های متغیر و اغلب کمتر از 500 وجود دارند که به احتمال فراوان نویز میباشند و با فیلتر کردنشان به سیگنال زیر رسیدیم(با استفاده از گذاشتن آستانه ی 300 در کد که اندازه های زیر 300 را صفر کند در خط 21 کد)

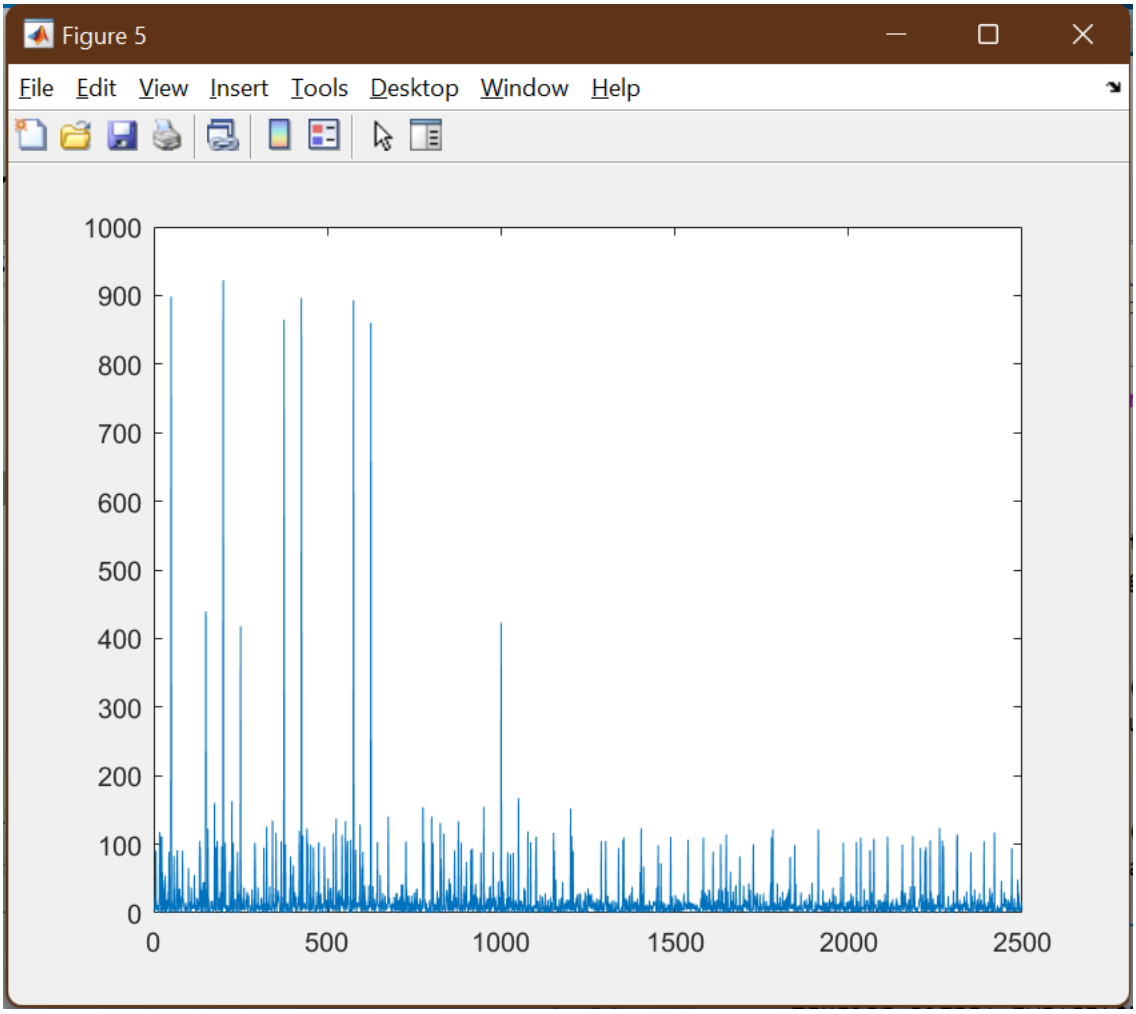


همانطور که در نمودار میبینم نمودار ما اکنون سیگنال های فرکانسی منظم تری دارد و در حوزه ی زمان برای تبدیل طبق مرحله های بعدی پیش رفتیم.

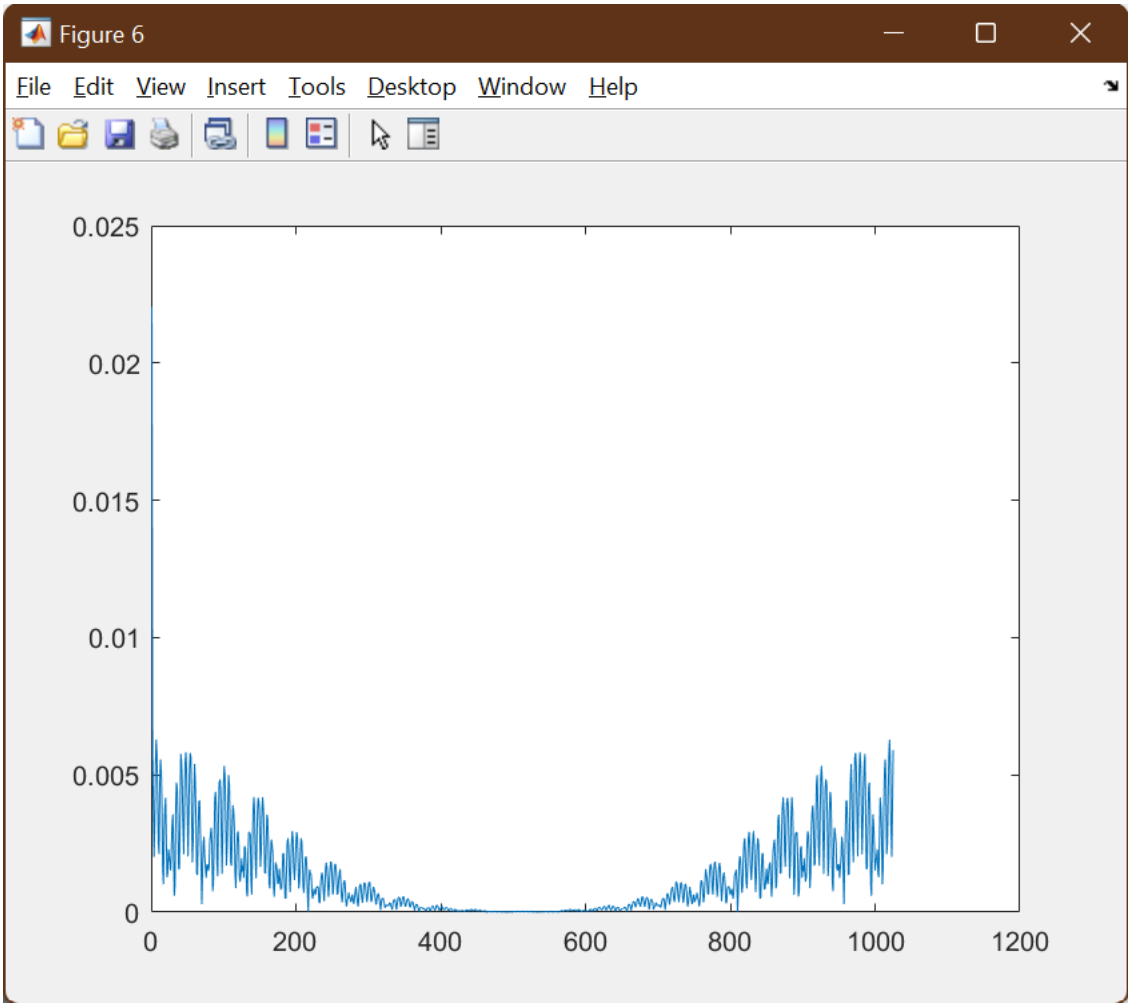
در قسمت بعد کد به مانند بخش قبل به ترتیب با تابع معکوس تبدیل فوریه(ifft) تبدیل معکوس فوریه ی سیگنال تبدیل شده و فیلتر شده ی مرحله ی قبل مان را میدهیم و با نشان دادن سیگنال denoised شده بر روی محور اولیه-اش با همان طول دفعه ی اول تاثیر فیلترینگمان بر سیگنال ورودی مشهود است در ادامه با ذخیره ی صدا (audiowrite) آن را در یک فایل با نام second_phase_first_denoised_signal ذخیره کردیم.



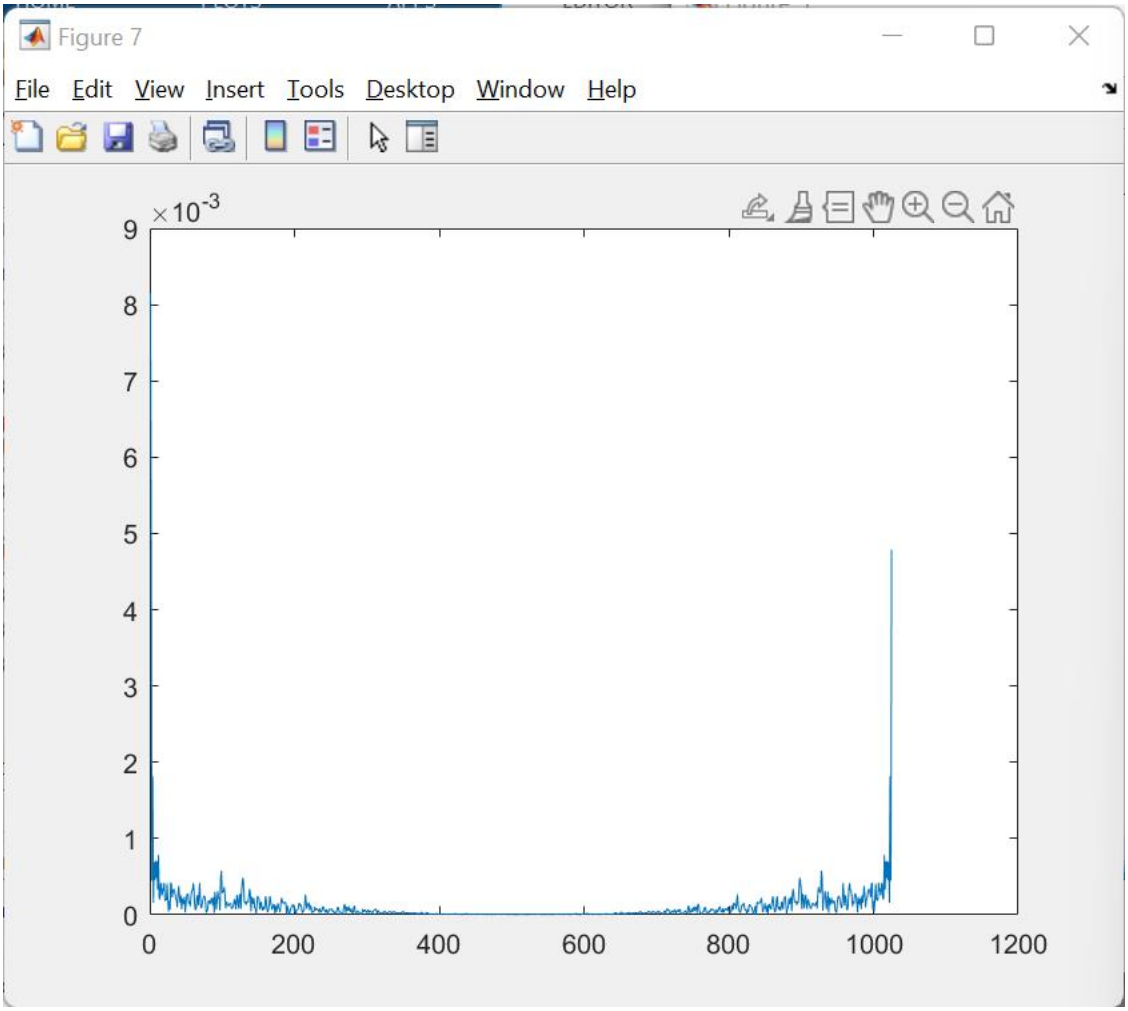
در قسمت بعدی کد بار دیگر از سیگنالی که پس از دینویز شدن به دست آمده تبدیل فوریه گرفتیم که حاصل به شکل زیر در آمد و با شکل اول قبل از فیلتر شدن شباهت زیادی دارد. حال دو حالت equalizer که در صورت پروژه آمده بودند با کمک تابع equalizer_function پیاده سازی شده اند به این صورت که ضرایب تقویت کنندگی برابر 10 هستند و تضعیف کردن برابر 0.1 در این صورت ما در کد تابع بر روی اندیس های مختلف سیگنال حرکت میکنیم و در هر یک چک میکنیم که در کدام یک از بازه های ما قرار میگرفته است و برحسب جایگاهی که از ضرایب ورودی تابع قرار میگرفته است و Band مربوط به آن.(به طور مثال در بازه ی 200 تا 500هرتز) سپس برحسب ضریب مربوطه اندازه ی آن خانه در ضریبش ضرب میشده است و اینکار را برای تمام خانه های سیگنال انجام دادیم.



در دو قسمت پایانی کد تبدیل فوریه سیگنال هایمان را ابتدا برای equalizer برای فرکانس های زیر 1200 هرتز بر روی نمودار کشیده ایم و فایل under1200 را برای صدایش تولید کرده ایم و در ادامه برای equalizer برای فرکانس های بالای 2000هرتز بر روی نمودار کشیده ایم و فایل above2000 را برای صدایش تولید کرده ایم.(در هر دو حالت از ضریب 10 برای تقویت بازه ی مطلوب و 0.1 برای تضعیف بازه ی نامطلوب استفاده کردیم)



شکل برای under1200



نمودار برای above 2000

تفاوت دو سیگنال تقویت شده در این بود که در سیگنال under 1200 به علت اینکه اغلب فرکانس هایمان در آن قرار داشتند اندازه ی فرکانس های صدایمان را بالا میبرد و باعث میشد شدت صدا تقویت بشود و در عوض در سیگنال above 2000 برعکس حالت قبل چون اغلب فرکانس هایمان خارج آن بازه قرار داشتند اندازه ی فرکانس هایمان را کم میکرد(تضعیف میکرد) و باعث میشد شدت صدا تضعیف بشود.