دانشکده مهندسی برق

گزارش تمرین کامپیوتری 1

سیستم های مخابراتی

تهیه‌کننده:

زهرا ملکی

بهار 1403

سوال 1)

.1

By setting the frequency sample and bit depth according to the question's requirements and using the audiorecorder function along with other necessary functions, I recorded my sound for 10 seconds. I saved the recorded sound in both recorded_audio.mat and recorded_audio.wav files.

```
recorder = audiorecorder(fs, bitDepth, 1);
%record
recordblocking(recorder, duration);
%to data
data = getaudiodata(recorder);
```

.2

The Short-Time Fourier Transform (STFT) is a representation of power/frequency in the time and frequency domains.

Spectrogram returns the short-time Fourier transform of the input signal, x. Each column of s contains an estimate of the short-term, time-localized frequency content of x.

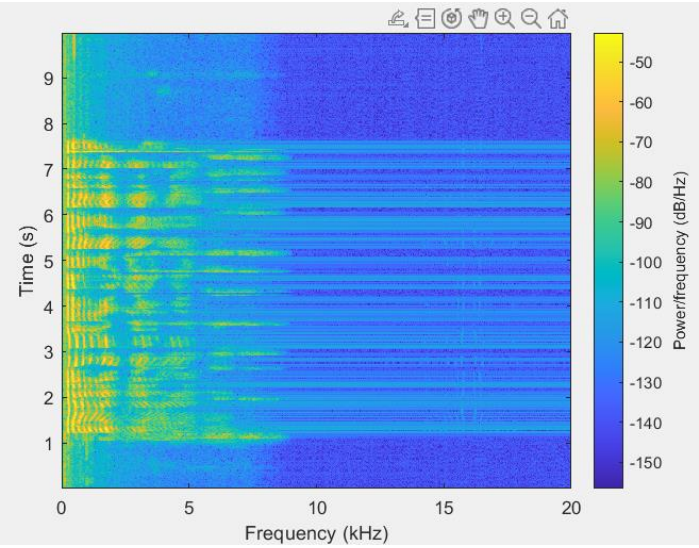
-window to divide the signal into segments and perform windowing, is applied to the signal and has an effect on the resolution.

-noverlap samples of overlap between adjoining segments

-nfft sampling points to calculate the discrete Fourier transform and returns a vector of cyclical frequencies,

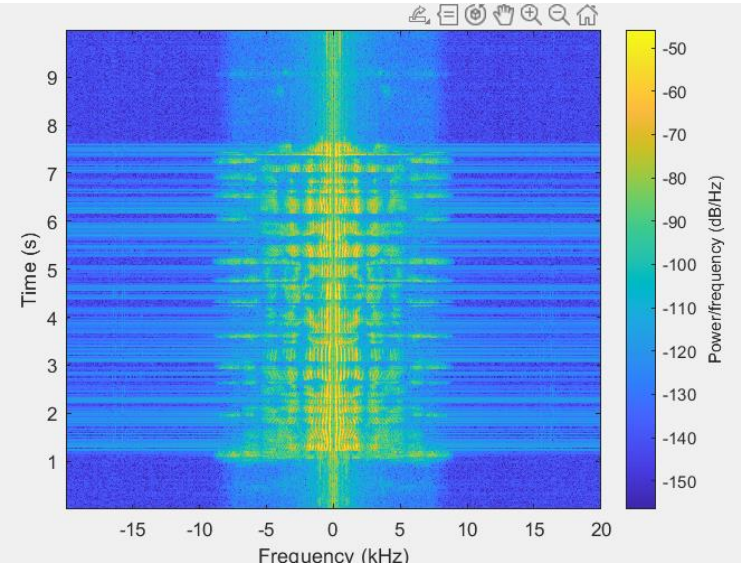
-fs, expressed in terms of the sample rate.

```
spectrogram(data, window, noverlap, nfft, fs);
```



.3

This phenomenon occurs due to the real nature of an audio signal. An audio signal is a real signal, and as a result, its frequency spectrum always lies on the even frequency axis.

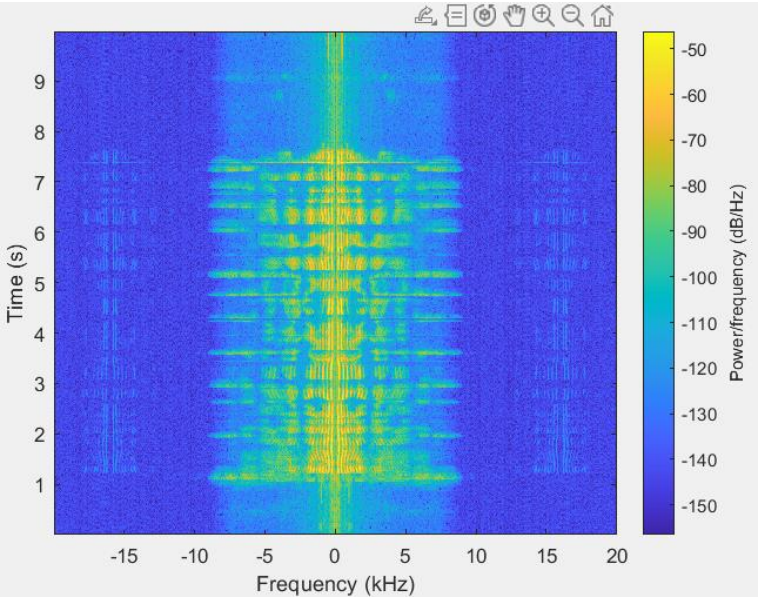


.4

windowTypes = {'rectwin', 'hann', 'hamming', 'blackman', 'gaussian'};

window = hann(nfft);

A popular window that reduces spectral leakage and provides better frequency resolution.



.5

To obtain power, we can use auto-correlation of the signal. Since our signal is a digital signal and is shifted, the point where it has maximum self-correlation gives us the power value.

Power: 423.1691

$$R_{v(\alpha)} = \lim_{T\to\infty} \frac{1}{T} \int v(t)v^*(t-\alpha)dt$$

$$R_{v(0)} = < v(t), v(t) > = P$$

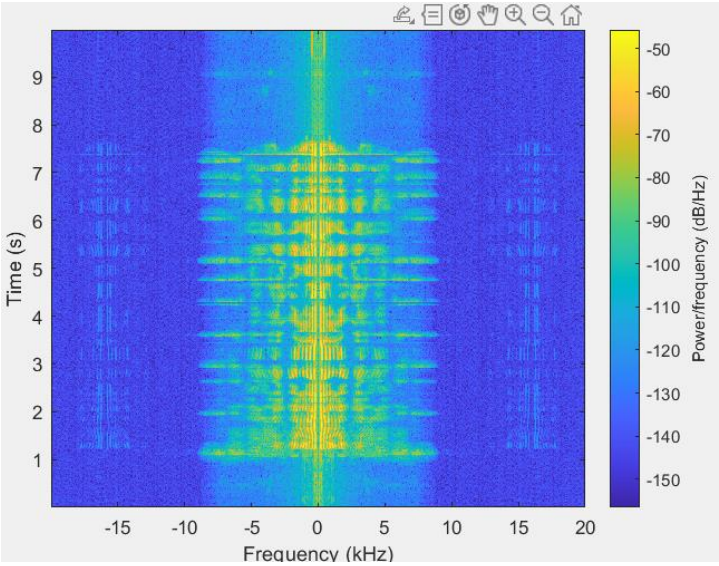
$$if\ R_{v(0)} \geq R_{v(t)}$$

.6

nfft is the number of samples of the input signal used to compute the frequency spectrum.

As it increases the resolution of frequency and time domain increases. Increasing nfft improves time and frequency clarity of the spectrum. It also improves frequency resolution. However, larger nfft values require more resources and may not always provide significant improvements.

Nfft=10000



.7

To normalized the data in a way that normalized power became 1, I used:

normalized_data = data / sqrt(max(autocorr_result));

-result: normalized power= 1

Then I used soundsc(normalized_data,fs) to hear my normalized voice.

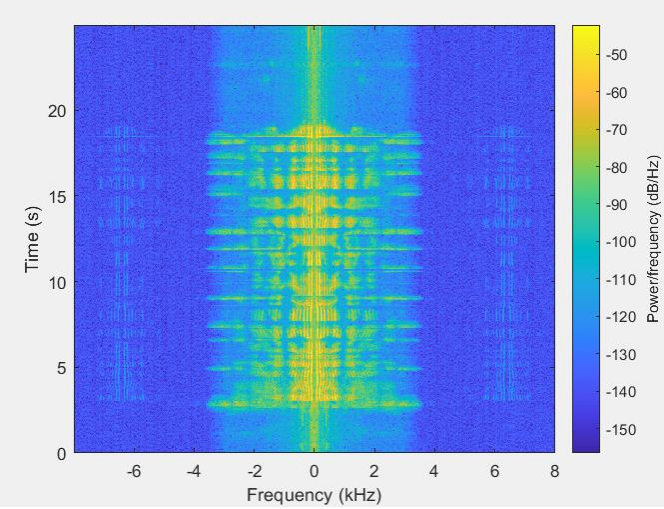
.8

Nyquist–Shannon sampling theorem : If a function contains no frequencies higher than *w*, then it can be completely determined from its ordinates at a sequence of points spaced if ws>2w

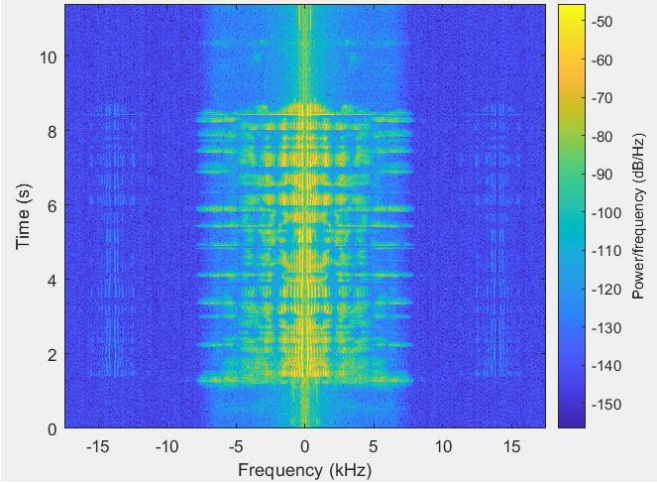
Based on the image we can assume that w=16k Hz and higher frequencies is the background noise.

Fs > 32 kHz

fs_min = 32000;



And yes, the band width changed a little bit because we are in the limit of Nyquist–Shannon sampling theorem. But, if we go a bit higher band width with would not change and we can drive the actual signal.
fs = 35000;



We can see that the band width stays the same.

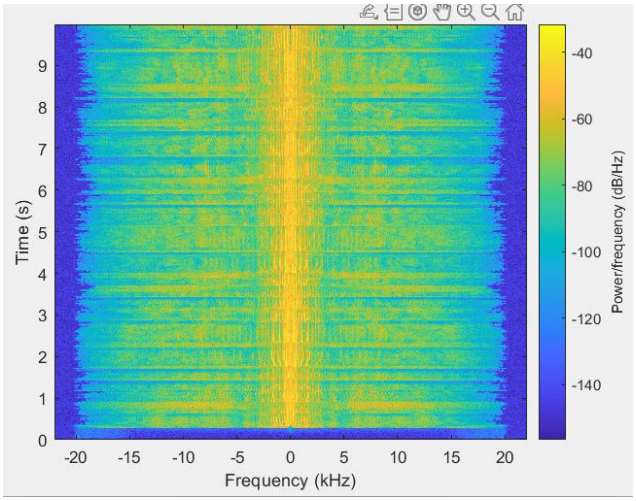
.9

The most common and widely used sample rate in audio is 44kHz, followed by 48kHz and 96kHz.

Sampling rates are based on the idea that to record and recreate a frequency in the human hearing range you must be able to sample it at least twice every wave cycle. Since the human hearing range is between 20 Hz and 20 kHz, for the best quality we should record at a sample rate of at least 40 kHz.

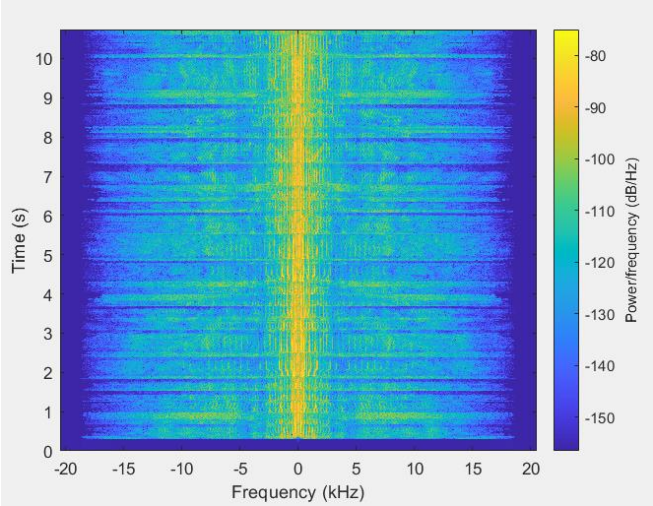
In the 1980s, 44.1kHz was established as the de facto sample rate for Compact Discs (CDs). It has since become the standard for most digital audio formats, including MP3 and WAV files. The higher you set your sample rate, the more data your computer will have to store.
44.1 kHz offers a nice tradeoff between file size and audio quality. It's important to note that although 96 kHz captures more data than 44.1, it doesn't necessarily produce better audio quality.
Why not?
-Since an ADC captures more samples per second at higher sample rates, your computer's CPU will have to work harder to process more data. You may experience glitches, dropouts, or even crashes if your CPU isn't equipped to handle an increased workload. Because it can accurately reproduce frequencies up to 22.05 kHz – slightly above the highest frequency audible to the human ear.

For my music Fs = 44100 :



Based on the spectrum w is 20kHz approximately. Fs>40Hz. For down sampling, I first normalized the music in channel 1 and reduce the sampling rate to 41Hz.

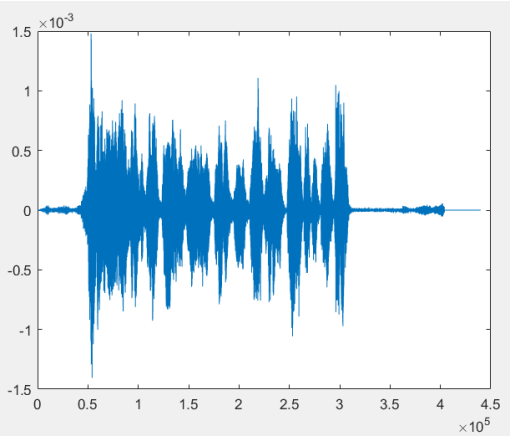
```
nfft = 1024;
window = hann(nfft);
noverlap = 0;
```

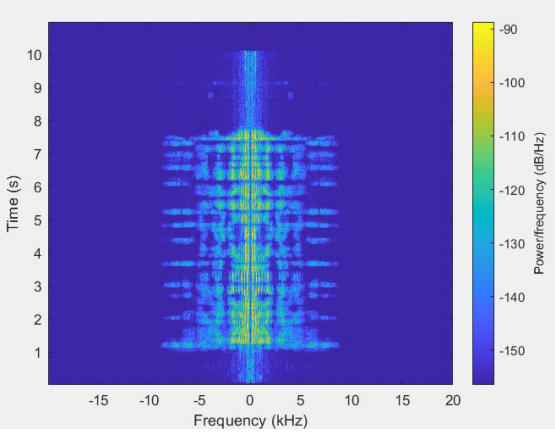


سوال 2)

.1

To do this, first I used syms to define Hand get the z inverse transom hz. I needed to convert hz to numeric arrayto use conv between the normalized voice and the channel. The spectrogram is as bellow:



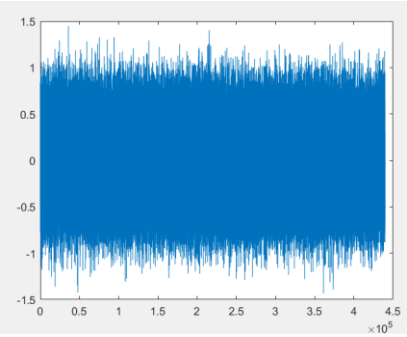


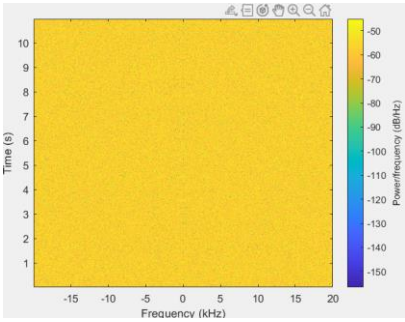
power_channel = 1.455

we have eco(distortion) as our bandwidth is larger than the distance between the time shifts and reduce of the amplitude.

.2

I use: sqrt(0.1) * randn(1, 40000) to produce normal noise.
noise_added=outputSignal + noise';
power_channel_noise = 43877.4572





.3

We use equalizer to cancel multipath and eco:

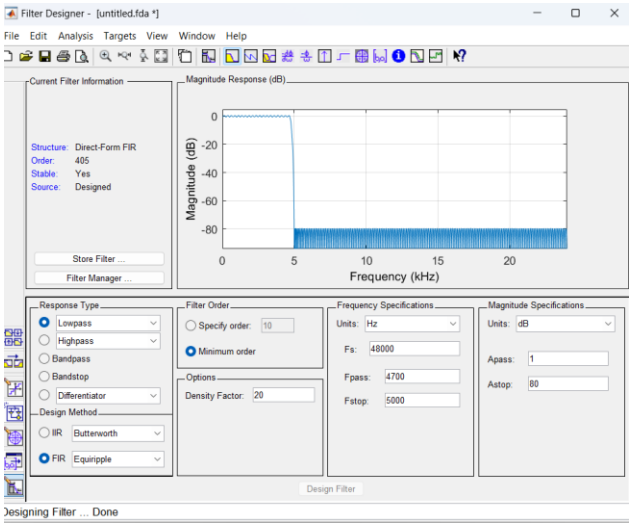
$$H_{eq} = \frac{20 * 0.05 = 1}{H\_c} \cong 20(1 + z^{-800} - z^{-1600} - z^{-2400} + z^{-3200} - z^{-4000})$$

Heq = 20* (1+z^-800-z^-1600-z^-2400+z^-3200-z^-4000);

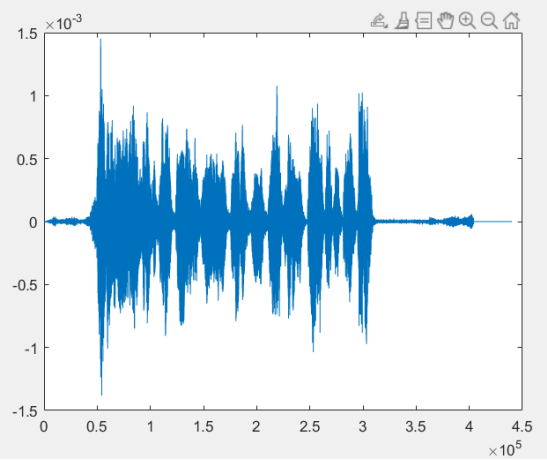
We do the same thing that we did in the part 1 and apply this Heq to the noise added signal.

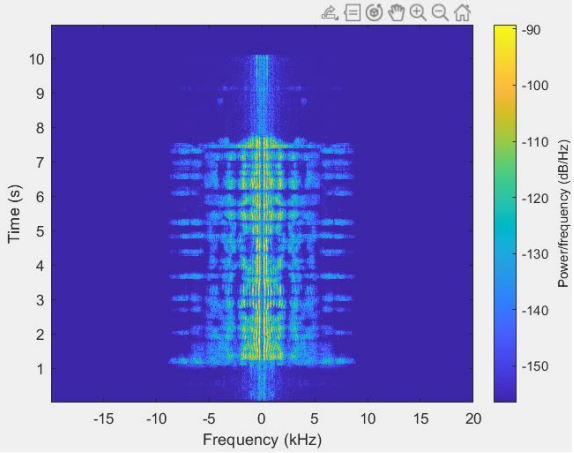
outputSignal_eq = (conv(noise_added', hz_num_eq))';

.4



```
filterDesign = designfilt('lowpassfir', 'PassbandFrequency', .47, ...
                    'StopbandFrequency', .50, 'PassbandRipple', ...
                    1, 'StopbandAttenuation', 80);
filterCoeffs = filterDesign.Coefficients;
filteredOutput = conv(noise_added, filterCoeffs);
filteredOutput:
```



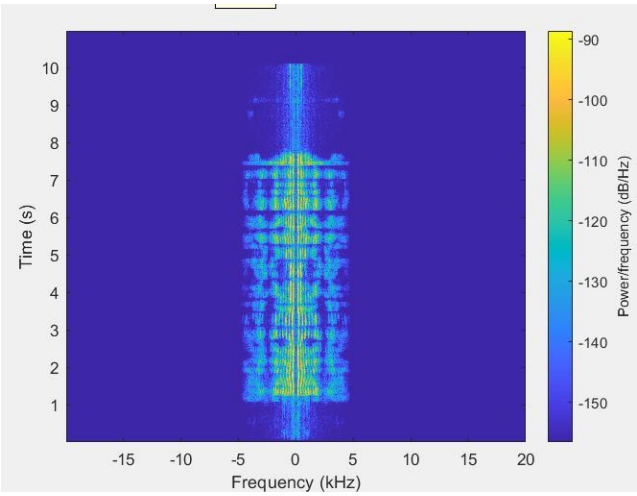


power_channel_fil = 0.014176

We can see that noise has been completely removed from the signal.

.5

No, we cannot reduce the stop band frequency as much as we want and it should be more than the bandwidth of the original signl. It is possible to loss some of the data of the original signal in the higher frequencies. Stop frequency =2k Hz :



As you can see, the band with reduced and we lost some of the data in the higher frequencies.