MIDDLE EAST TECHNICAL UNIVERSITY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

EE430 TERM PROJECT

FINAL REPORT

Yiğit TOPOĞLU 2094548

Emre DOĞAN 2093656

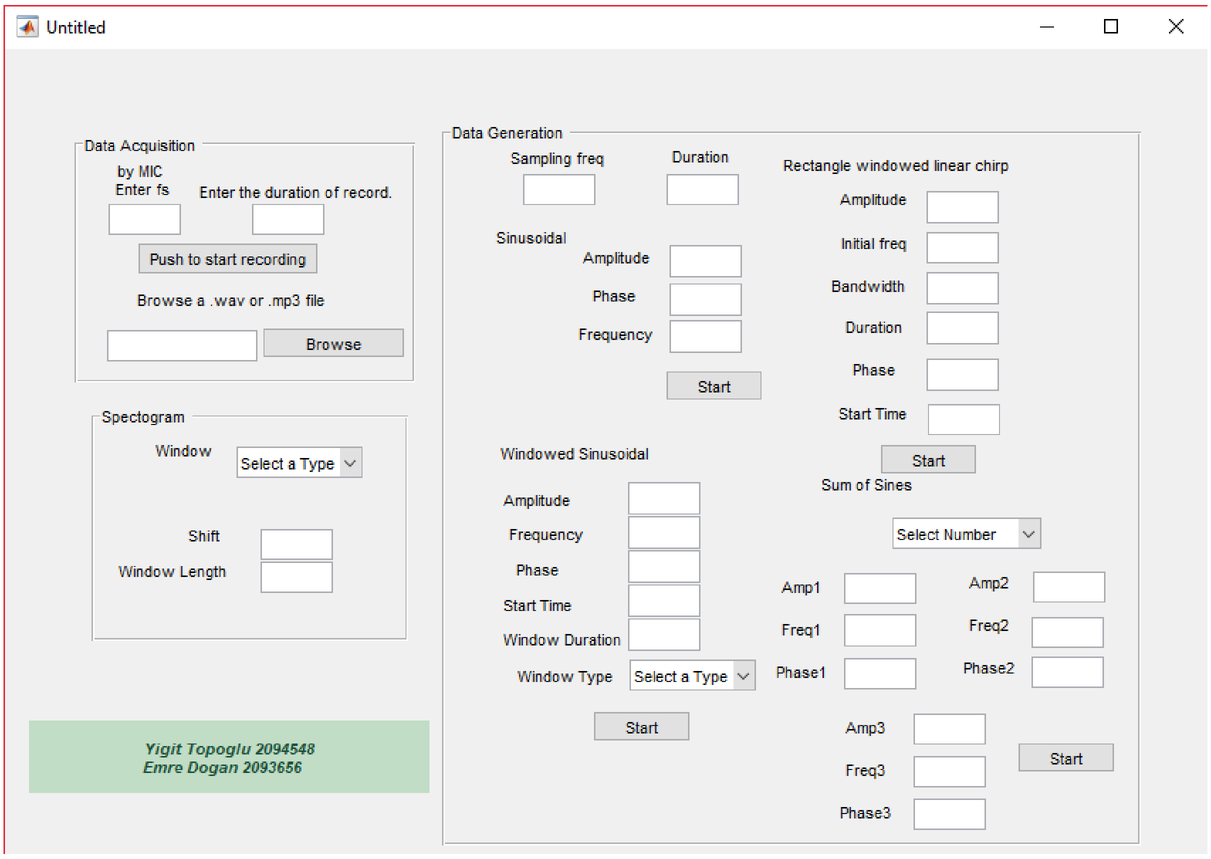
TABLE OF CONTENTS

INTRODUCTION

The purpose of this project is to get familiar with the real-time applications of digital signal processing. The term project is divided into two phases. In phase 1, many signals are generated, and STFT and spectrogram of a taken signal is created in MATLAB environment. In the second phase, a decimator, a quantizer, transform coding with DFT and DCT is created and some of the missing parts of a MPEG/Audio encoder to do audio compression via MATLAB. The algorithm and the results are explained in this report.

PHASE-1

In the first phase of the project, we implemented a program that takes STFT of a time-domain signal. The program must be able to work with various inputs such as computer generated data, audio files that are existing, or sound that are recorded with a microphone. For this phase a GUI is prepared via MATLAB for easy accessibility.The interface of GUI is shown in Figure 1.



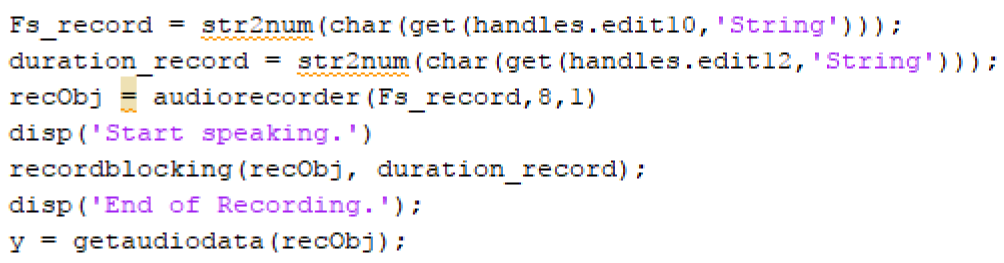
Yigit Topoglu 2094548
Emre Dogan 2093656

Figure 1. The GUI.

The phase is divided into three parts, data acquisition, signal generation and spectrogram.

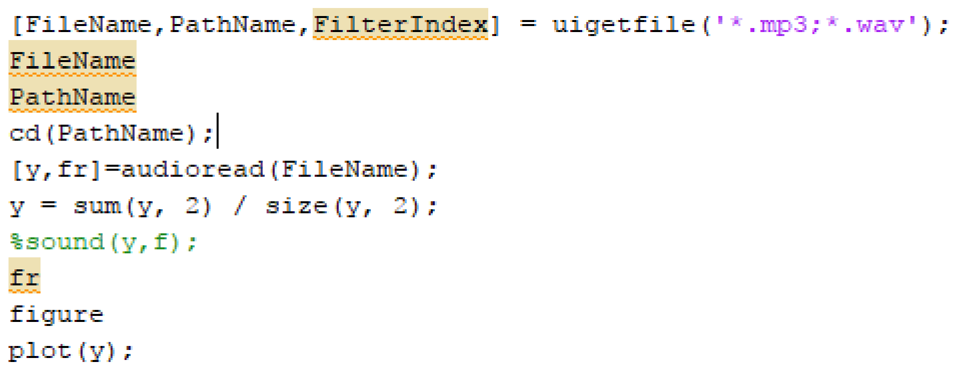
Data Acquisition

As the first part, the program will be able to work with recorded sound with microphone and an existing audio file. For this, the MATLAB code is shown in Figure 2 and 3.



```
Fs_record = str2num(char(get(handles.edit10,'String')));
duration_record = str2num(char(get(handles.edit12,'String')));
recObj = audiorecorder(Fs_record,8,1)
disp('Start speaking.')
recordblocking(recObj, duration_record);
disp('End of Recording.');
y = getaudiodata(recObj);
```

Figure 2. Data acquisition from the microphone.



```
[FileName,PathName,FilterIndex] = uigetfile('*.mp3;*.wav');
FileName
PathName
cd(PathName);
[y,fr]=audioread(FileName);
y = sum(y, 2) / size(y, 2);
%sound(y,f);
fr
figure
plot(y);
```

Figure 3. Data acquisition from an existing file on the computer.

As seen from Figure 2, there are two inputs when we acquire data from the microphone, the desired sample rate and the recording duration. After the recording is done, you can hear the recorded signal. As for the data acquisition from an existing file on the computer, files with .mp3 and .wav format can be read.The acquired example signals are shown in Figure 4 and 5.

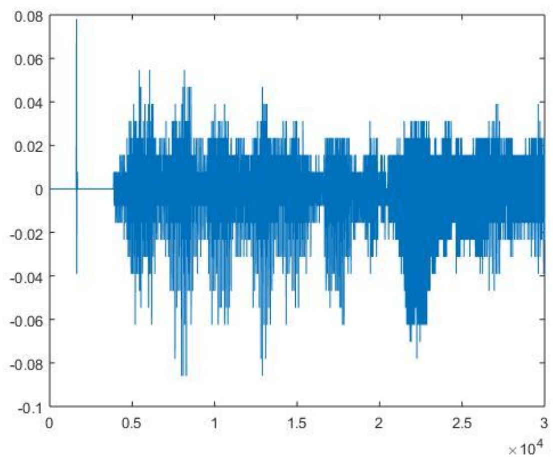


Figure 4. Signal coming from a microphone.

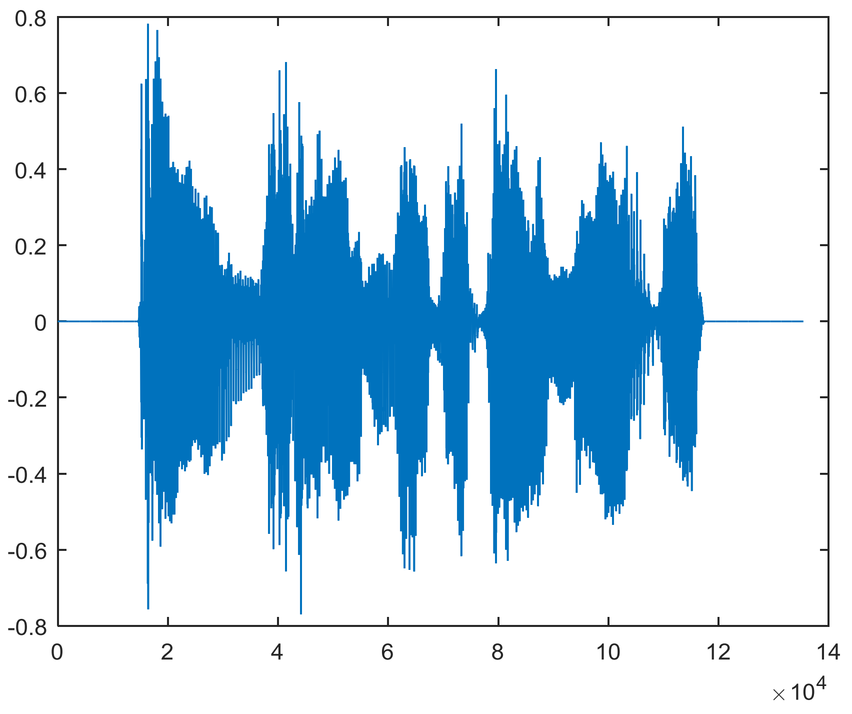


Figure 4. Signal coming from an existing .mp3 file.

Signal Generation

For the second part of the project, signals are generated. With the GUI, we can generate sinusoidal signals, windowed sinusoidal, rectangle windowed linear chirp and sum of up to 3 sinusoidal signals. The generated time signals are shown in Figure 5 to 10. The generated signal have two common inputs, sample frequency and duration. In windowed sinusoidal mode, 8 types of windows can be used.

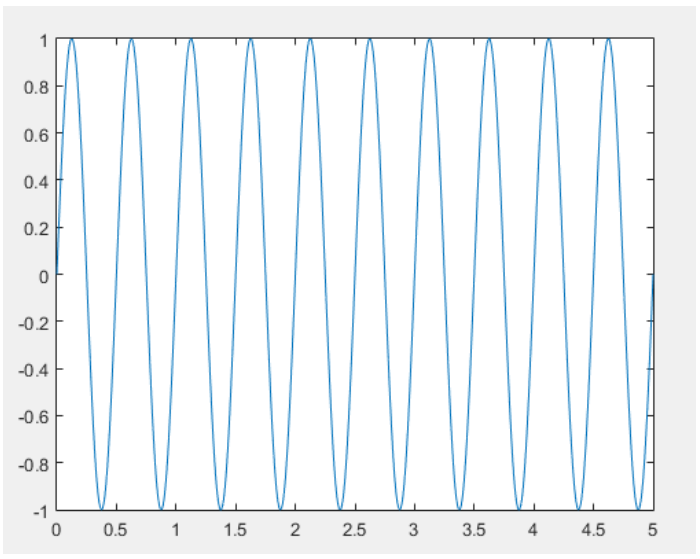


Figure 5. Sinusoidal signal.

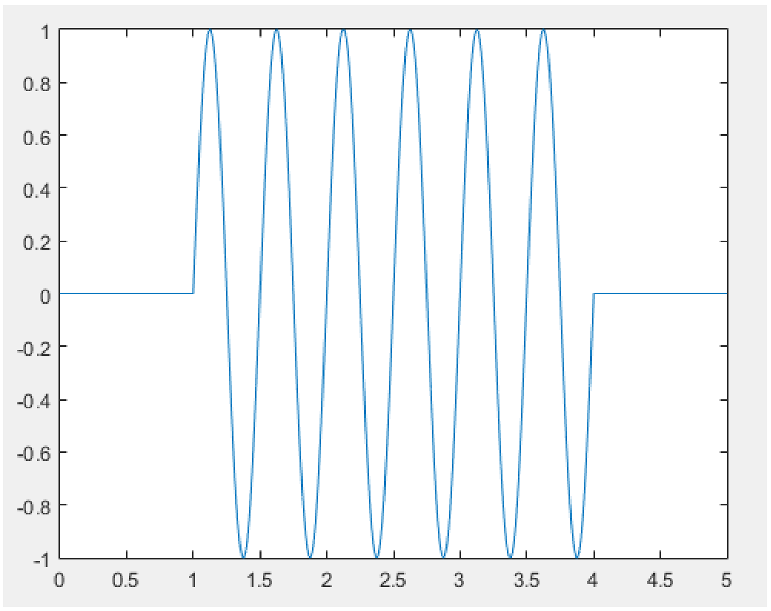


Figure 6. Rectangle windowed sinusoidal signal.

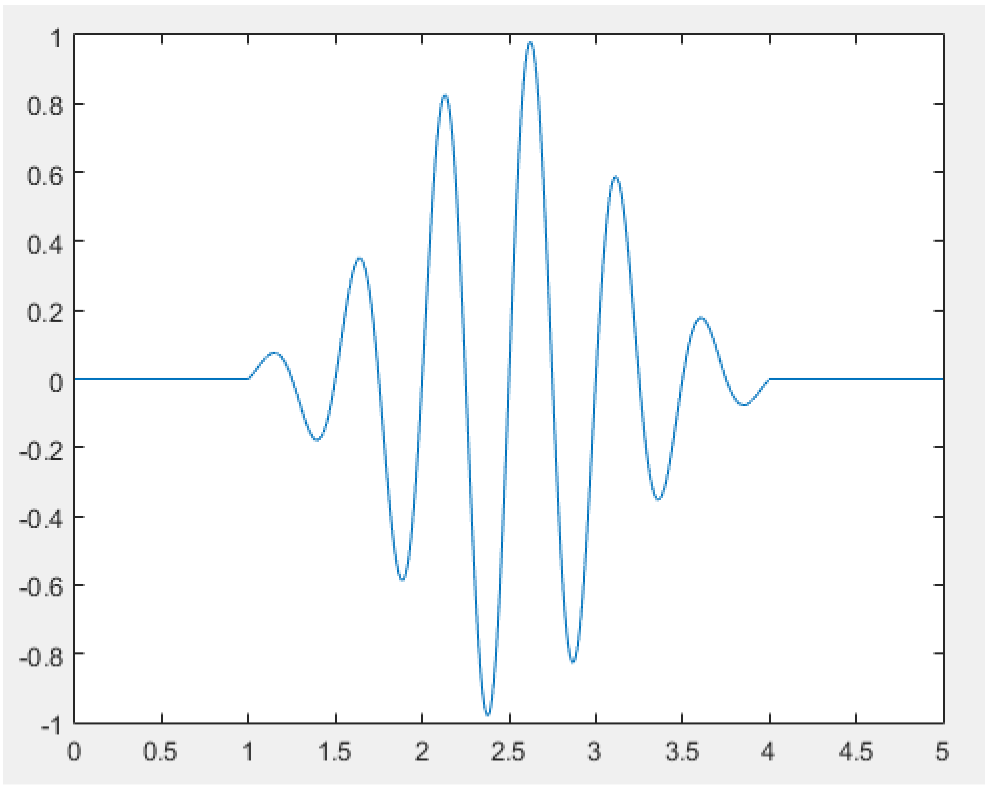


Figure 7. Gaussian windowed sinusoidal signal.

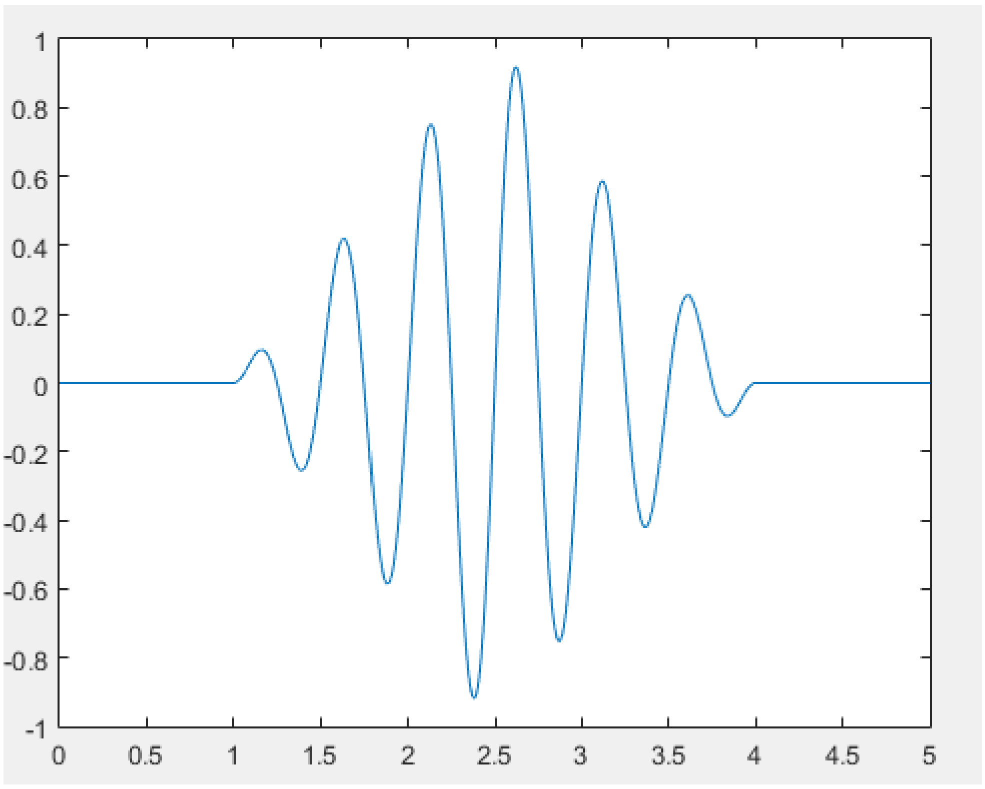


Figure 8. Triangular windowed sinusoidal signal.

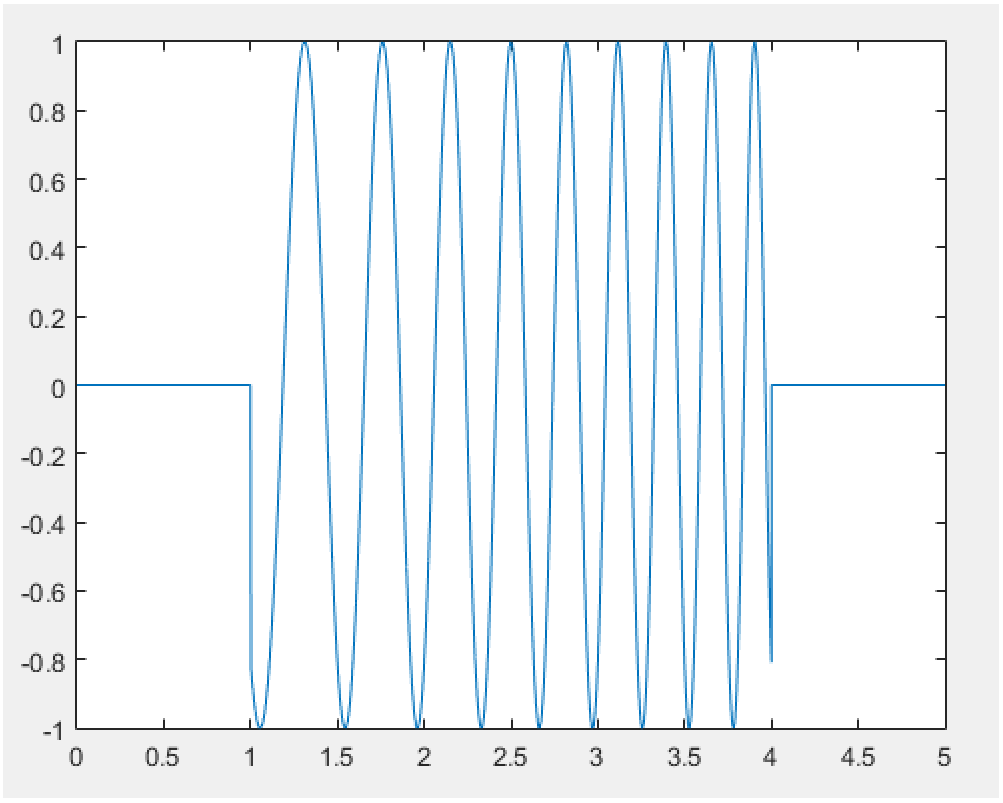


Figure 9. Rectangular windowed linear chirp (from 1 Hz to 6Hz).

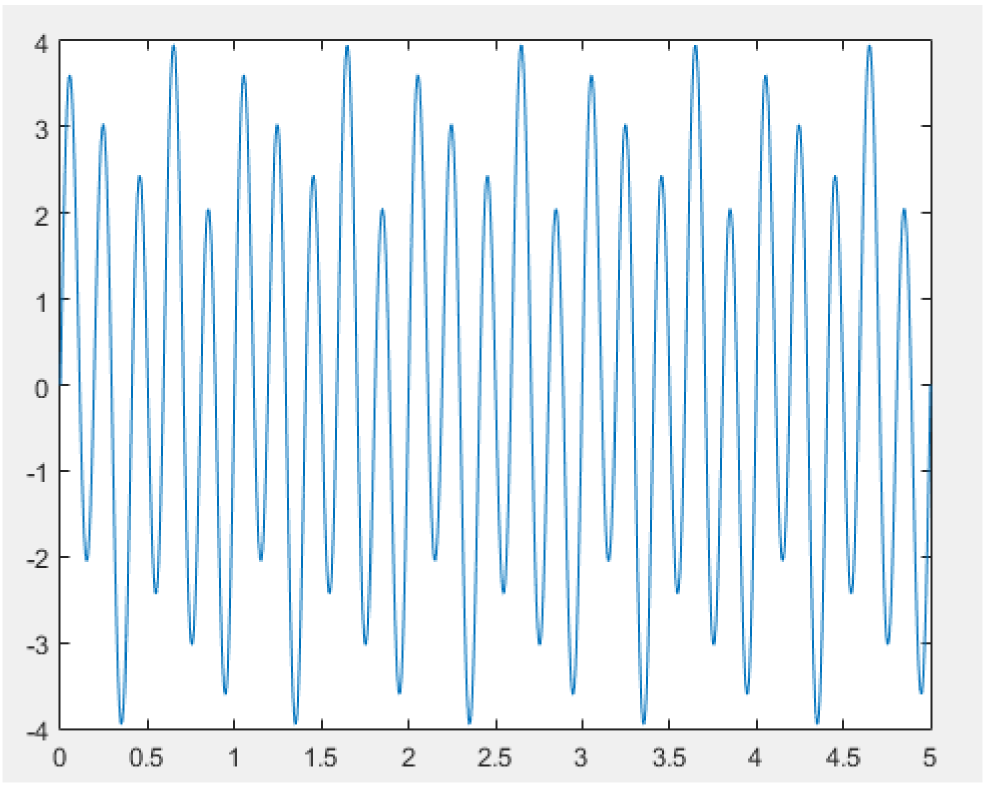


Figure 10. Sum of two sines (f1=2Hz f2=5 Hz).

STFT and Spectrogram

As the third part of the first phase we need to take the Short Time Fourier Transform (STFT) and the spectrogram of a signal. A spectrogram is a plot of magnitude of the STFT on time-frequency plane. We do STFT by windowing a portion the input signal,taking the DFT of windowed signal and shifting the window to take another portion of the signal, repeating until the end of the signal. The shift of the window must be at most equal to the window length not to miss even a little portion of the signal. The GUI designed can do STFT with 9 different window functions (Kaiser, Tukey , Chebyshev, Gaussian, triangular, rectangular, Hamming, Hanning and Taylor). The inputs of the STFT are shift amount and window length. The results for all signals are shown in Figure 11 to 25. The code for one case is shown in Figure 26.

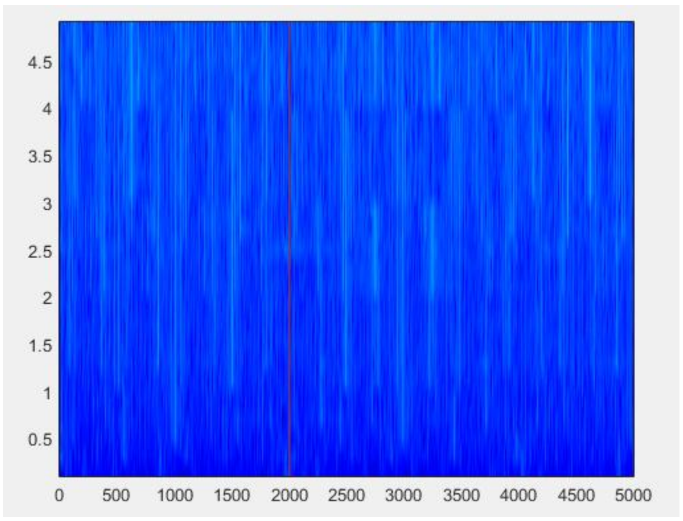


Figure 11. The spectrogram of a sinusoidal signal with frequency 2000Hz with rectangular window (window length=1000) .

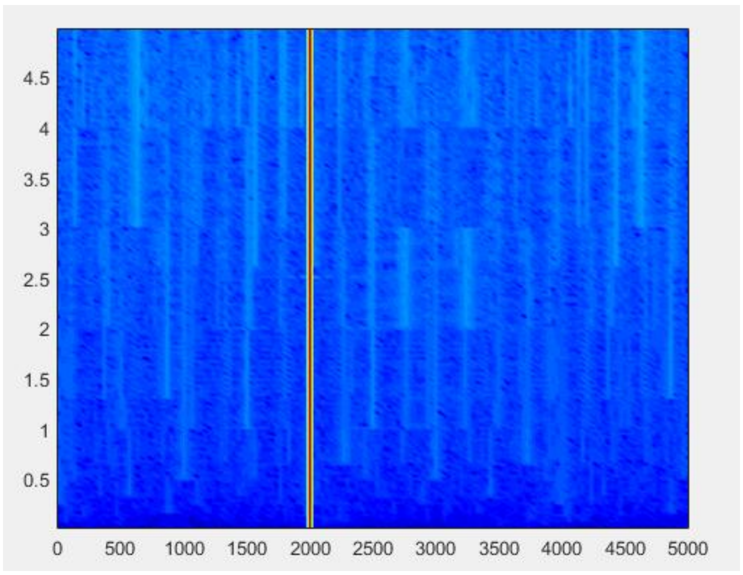


Figure 12. The spectrogram of a sinusoidal signal with frequency 2000Hz with rectangular window (window length=300) .

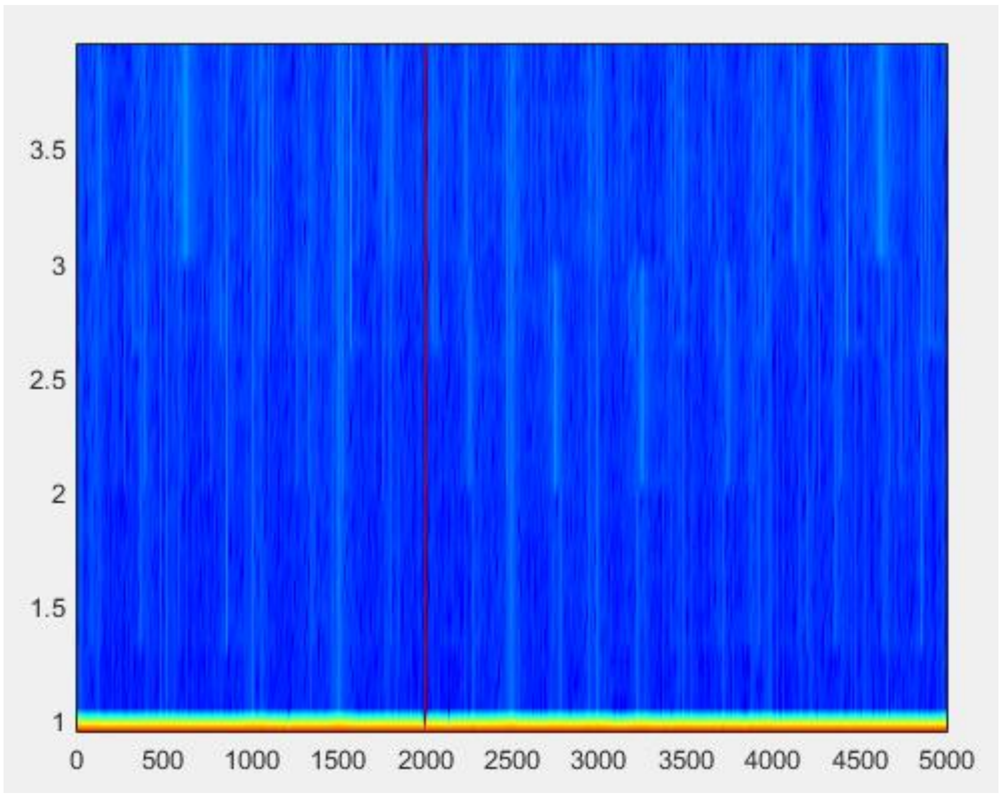


Figure 13. The spectrogram of a windowed sinusoidal signal with frequency 2000Hz with rectangular window (window length=1000) .

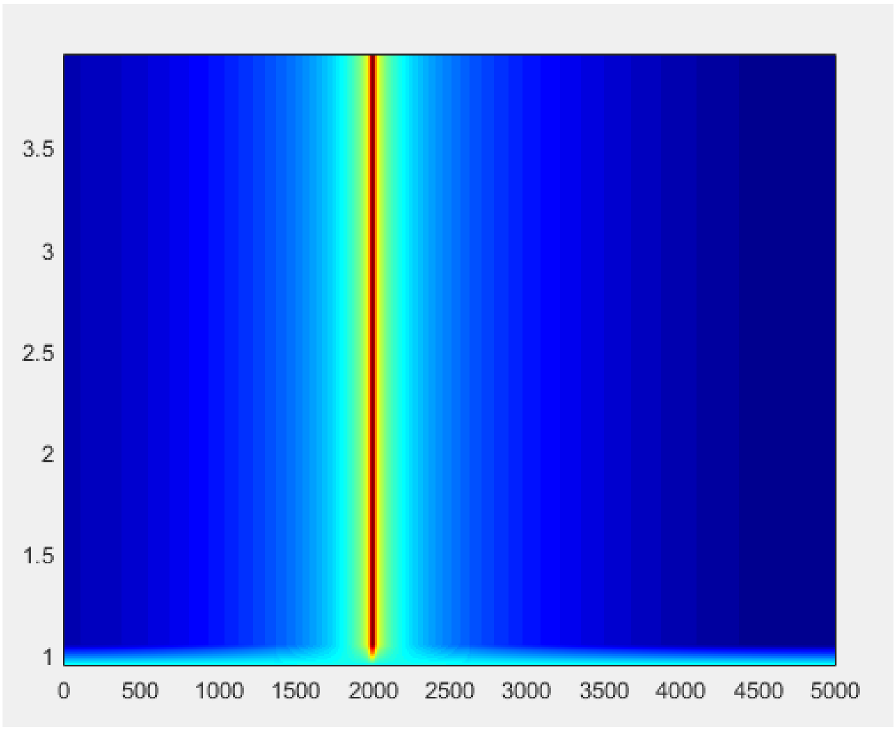


Figure 14. The spectrogram of a windowed sinusoidal signal with frequency 2000Hz with Gaussian window (window length=1000) .

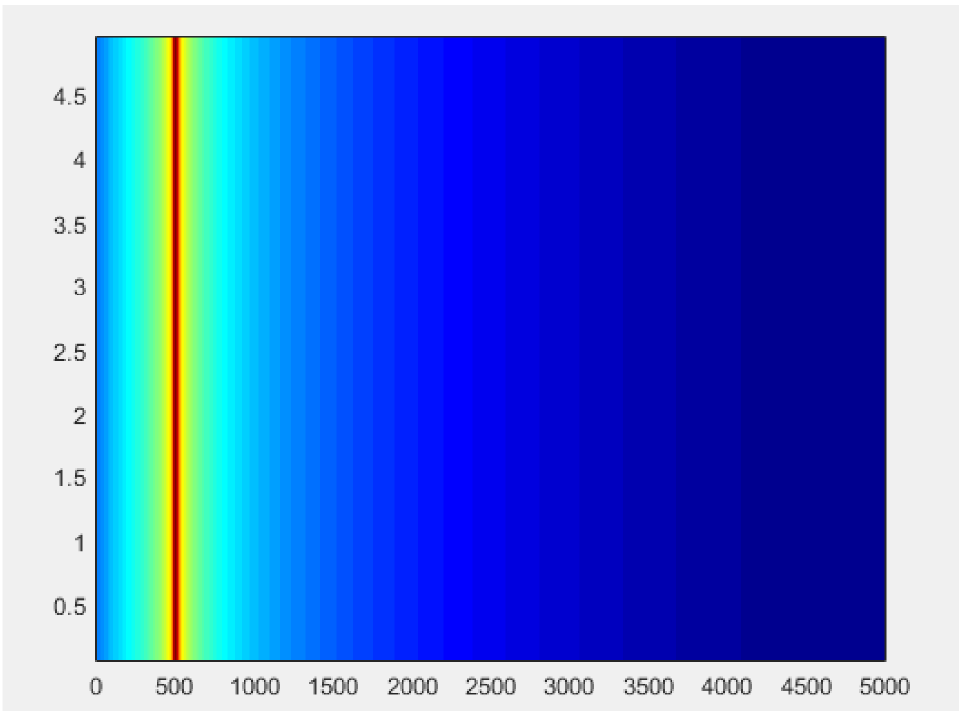


Figure 15. The spectrogram of a sinusoidal signal with frequency 500Hz with Gaussian window (window length=1000) .

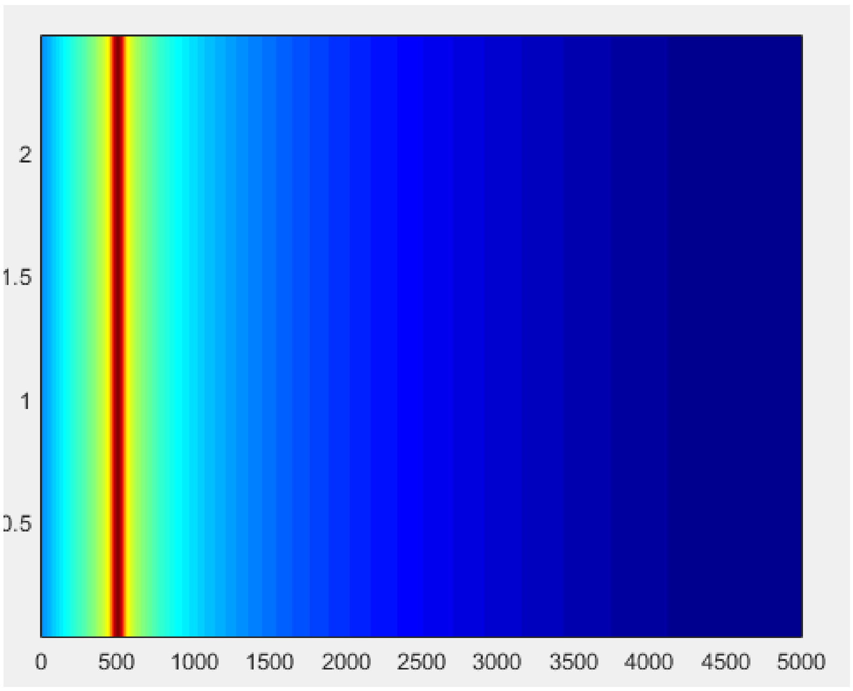


Figure 16. The spectrogram of a sinusoidal signal with frequency 500Hz with Gaussian window (window length=500) .

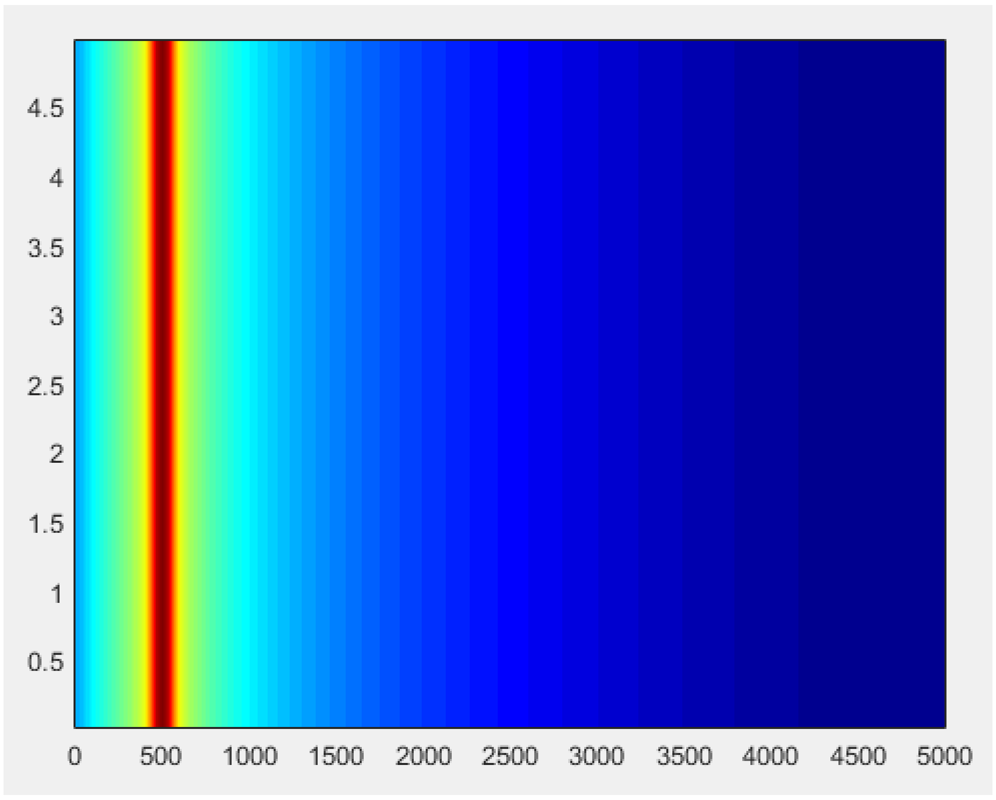


Figure 17. The spectrogram of a sinusoidal signal with frequency 500Hz with Gaussian window (window length=300) .

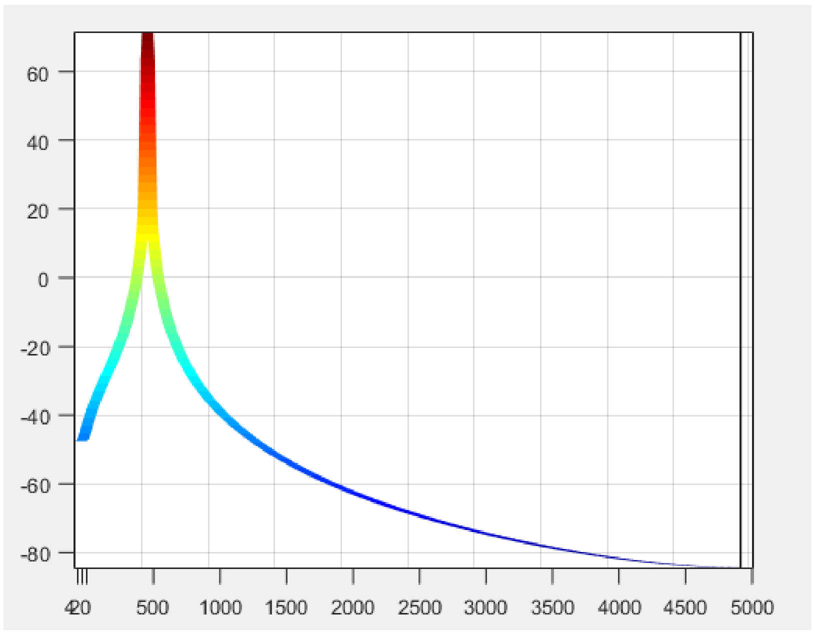


Figure 18. The STFT- frequency graph of the signal in Figure 10 with Gaussian filter.

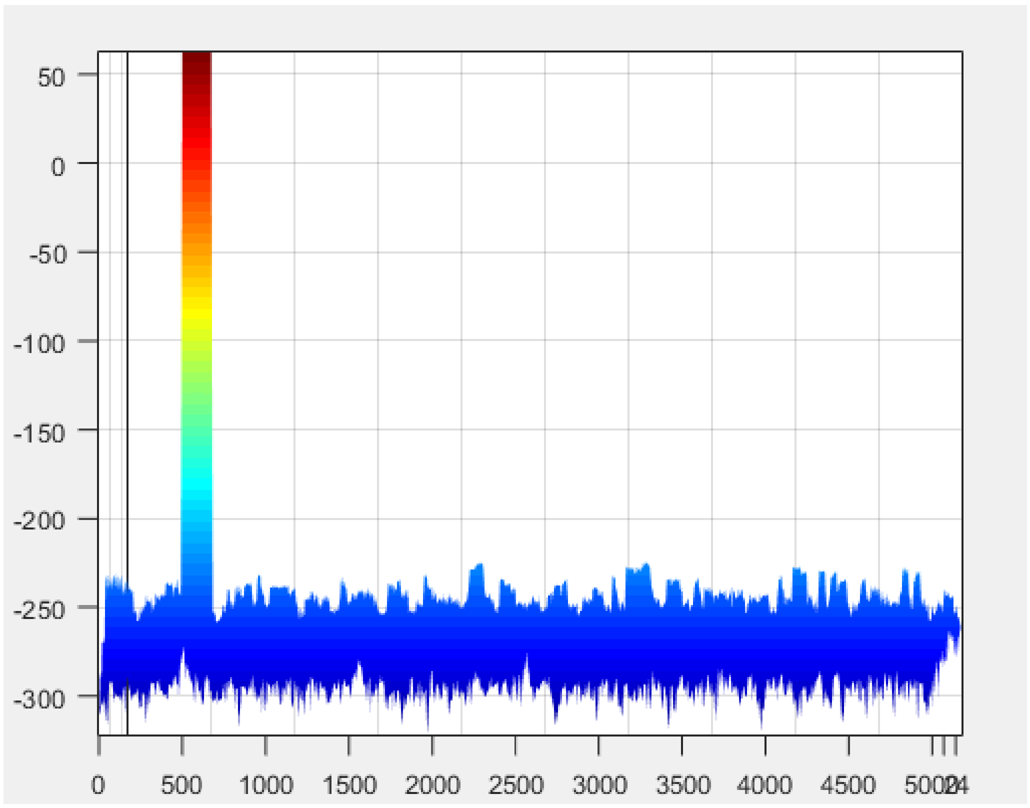


Figure 19. The STFT- frequency graph of the signal in Figure 10 with rectangular filter.

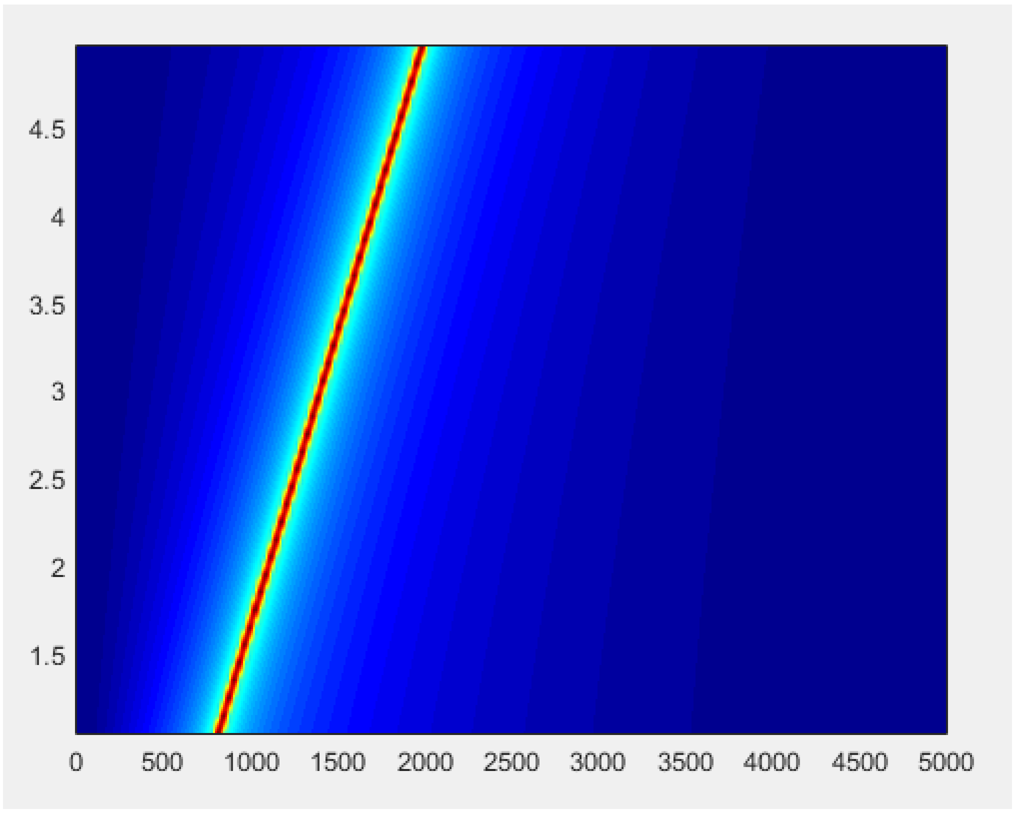


Figure 20. The spectrogram of a linear chirp with initial frequency 500Hz to 1500Hz with Gaussian window (window length=1000) .

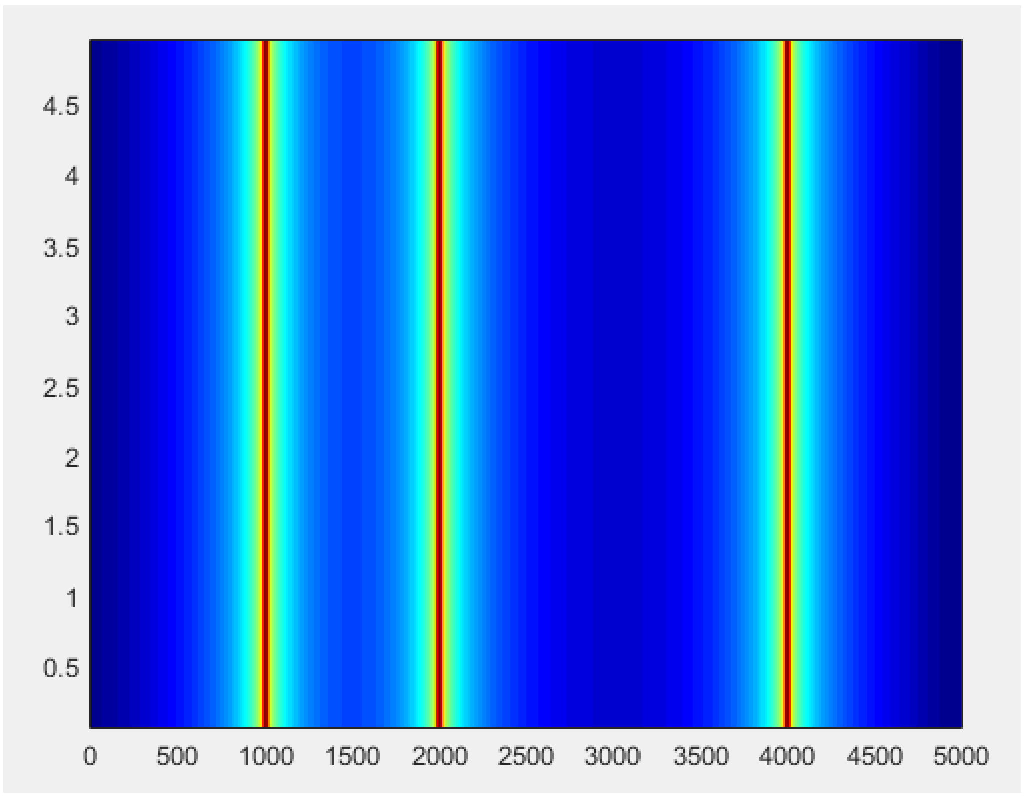


Figure 21. The spectrogram of a sum of 3 sines which have frequencies 1000,2000 and 4000 Hz with Gaussian window (window length=1000) .

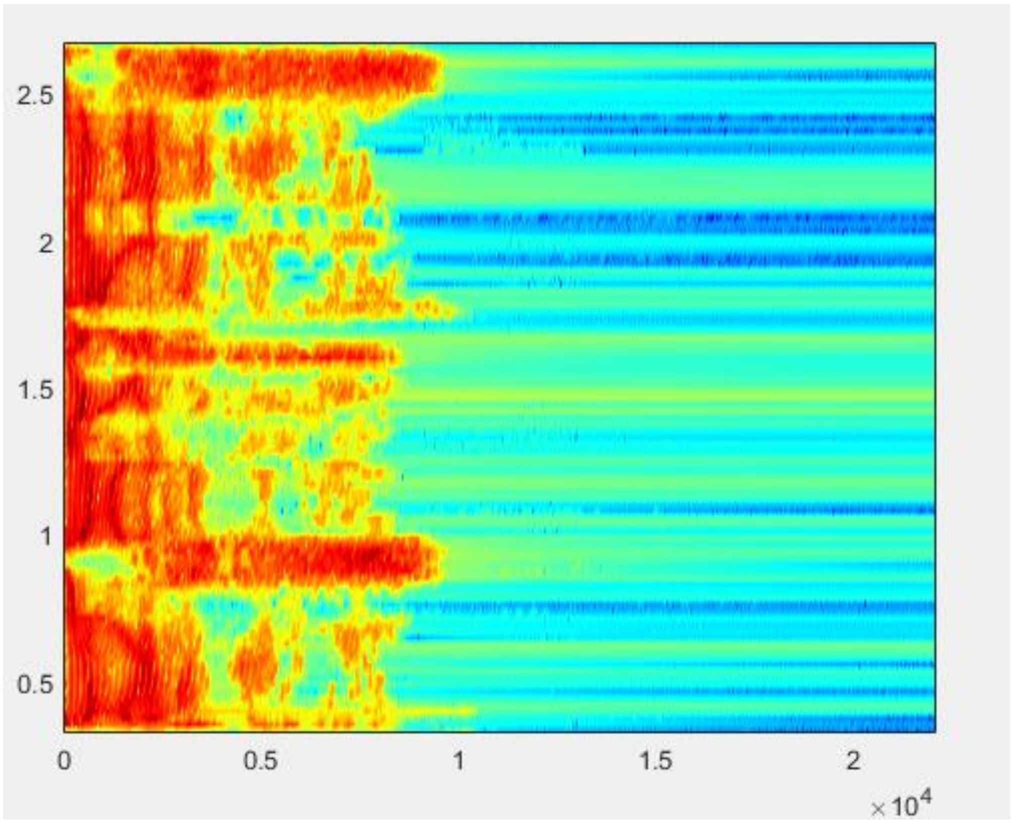


Figure 22. The spectrogram of a n existing audio file with Gaussian window (window length=1000) .

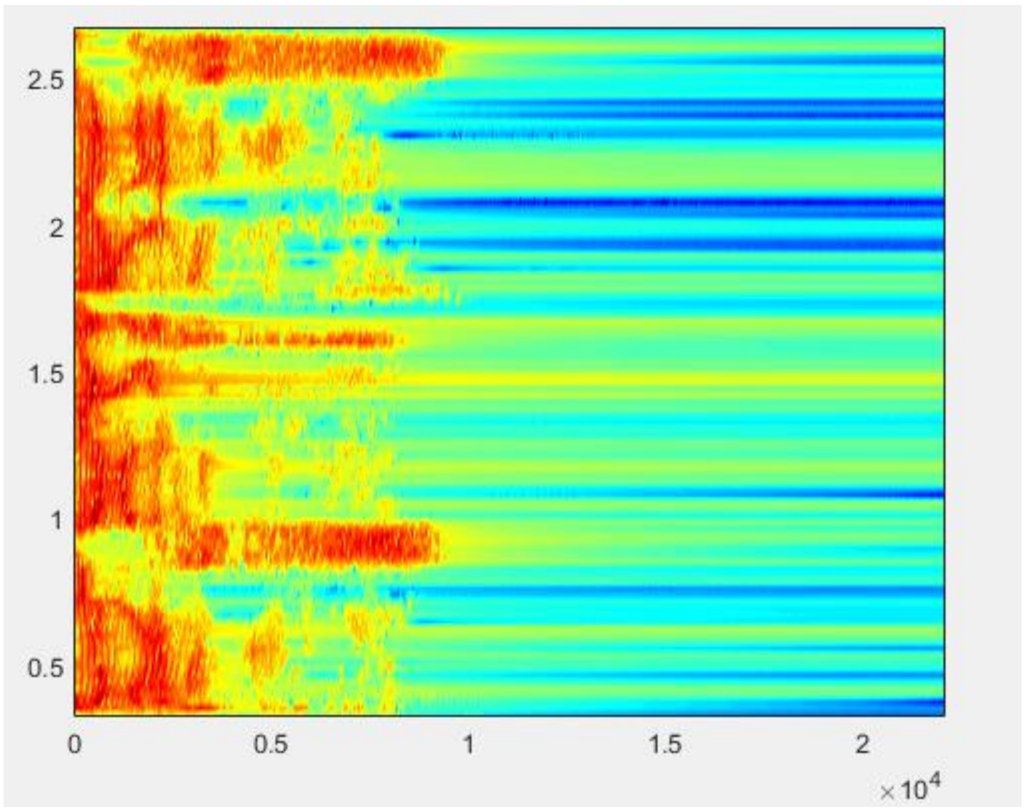


Figure 23. The spectrogram of a n existing audio file with rectangular window (window length=1000) .

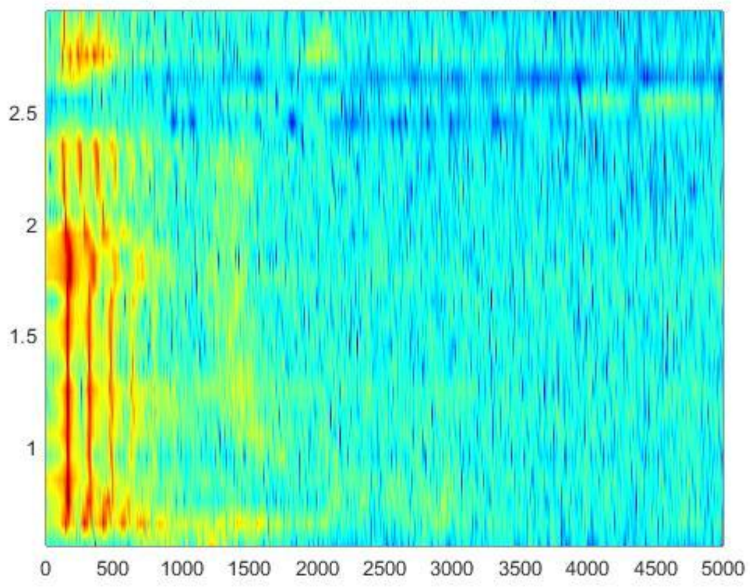


Figure 24. The taken signal from microphone in rectangular window.

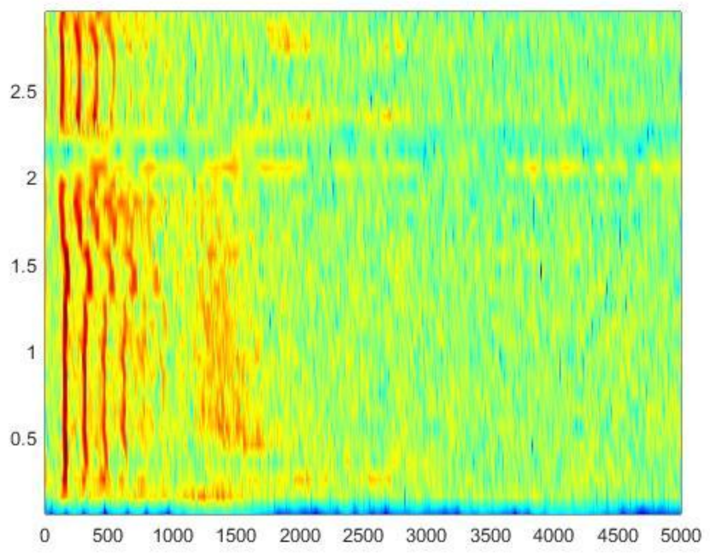
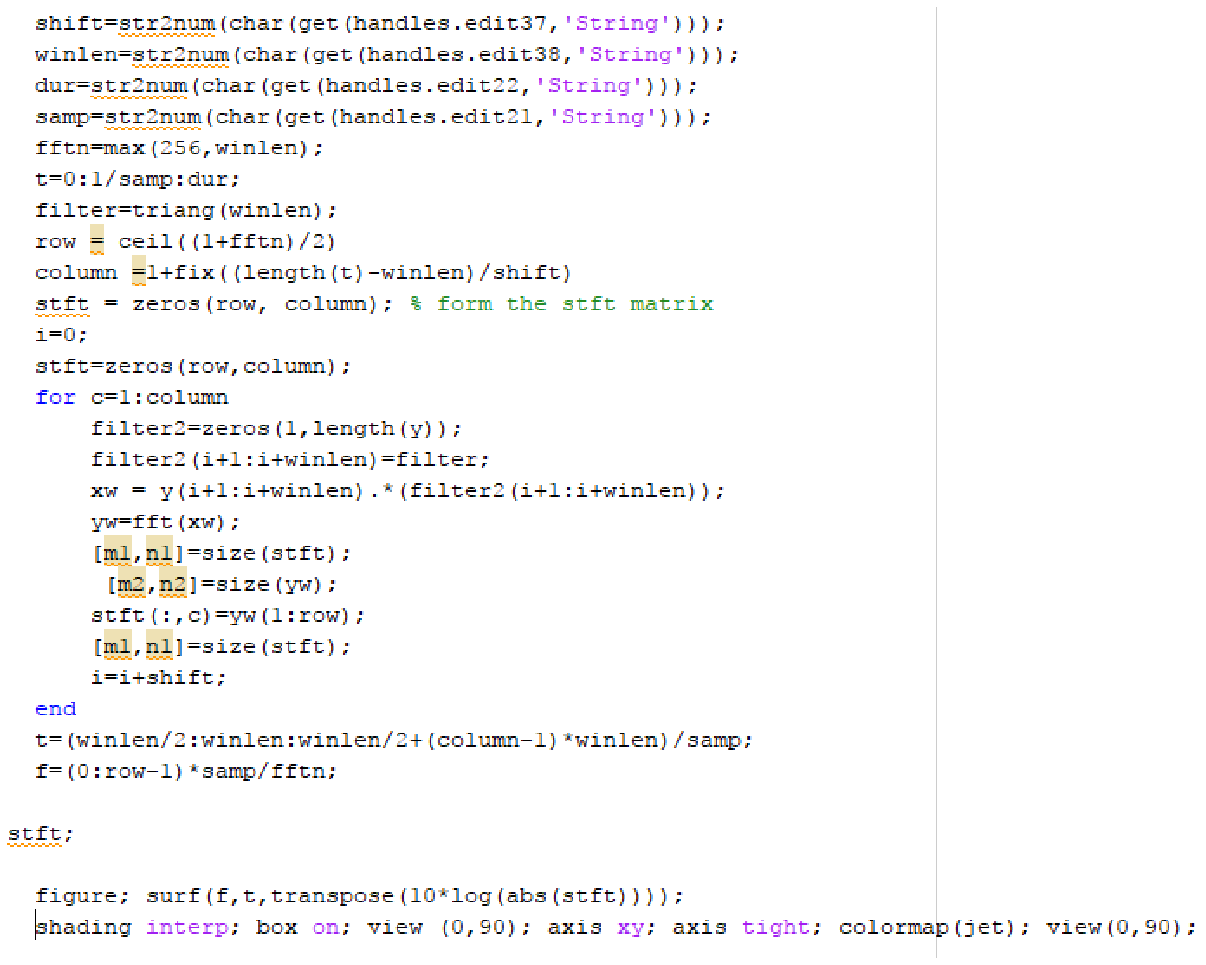


Figure 25. The taken signal from microphone in Gaussian window.



```
  shift=str2num(char(get(handles.edit37,'String')));
  winlen=str2num(char(get(handles.edit38,'String')));
  dur=str2num(char(get(handles.edit22,'String')));
  samp=str2num(char(get(handles.edit21,'String')));
  fftn=max(256,winlen);
  t=0:1/samp:dur;
  filter=triang(winlen);
  row = ceil((1+fftn)/2)
  column =1+fix((length(t)-winlen)/shift)
  stft = zeros(row, column); % form the stft matrix
  i=0;
  stft=zeros(row,column);
  for c=1:column
      filter2=zeros(1,length(y));
      filter2(i+1:i+winlen)=filter;
      xw = y(i+1:i+winlen).*(filter2(i+1:i+winlen));
      yw=fft(xw);
      [m1,n1]=size(stft);
       [m2,n2]=size(yw);
      stft(:,c)=yw(1:row);
      [m1,n1]=size(stft);
      i=i+shift;
  end
  t=(winlen/2:winlen:winlen/2+(column-1)*winlen)/samp;
  f=(0:row-1)*samp/fftn;

stft;

  figure; surf(f,t,transpose(10*log(abs(stft))));
  shading interp; box on; view (0,90); axis xy; axis tight; colormap(jet); view(0,90);
```

Figure 26. MATLAB code for STFT and spectrogram.

- As seen in the Figures, our spectrogram and STFT gives the expected results.
- As seen in Figure 15,16 and 17, aswindow length increases the range of peak at the frequency of the sinusoidal signals decrease because when the window length increases the FFT data we found is much closer to signal. However, as we increase our window length, the noise becomes visible, which is undesired, shown in Figure 11 and 12.
- Gaussian windowed signals give a smoother STFT than rectangular windowed signals. This is becauserectangular windowing causes ripples when windowing. Even though rectangular filter gives exactly what we desire in theory, these ripples create noise, which distorts our signal, shown in Figure 18 and 19.
- In Figure 14, windowed signal have a light line at the bottom of the spectrogram, unlike in the spectrogram of a normal sinusoidal. This is because of the sudden increase when the windowed signal starts.
- In Figure 20, the linear chirp gives us a STFT with sloped line. This is expected as the frequency increases with linear chirp.

PHASE-2
Introduction

In this phase of the project, we studied on the implementation of an audio compressor. There are two possible methods to compress an audio such that,

- Entropy coding (lossless process)
- Quantizing some frequency components (lossy process)

In this term project, we investigated the second option.

We examined this project in 4 main steps.

1. Reducing the number of time samples.
2. Reducing the bits per sample on time domain.
3. Transform coding.
4. MPEG/ Audio encoder.

These steps will be investigated in the given order.

Part 1. Reducing the number of time samples

In this step, we were supposed to reduce the number of time samples. For this purpose, we used `decimate` and `interp` functions together to obtain fractional downsampling. Decimation reduces the original sampling rate of a sequence to a lower rate where interpolation increases sampling rate by an integer factor of its input.

Before decimating, we have to do lowpass filtering to the signal because as we down sample the signal widens in frequency domain and this can cause aliasing, making the perfect reconstruction of original signal impossible. We do filtering to prevent aliasing.

To reduce the number of time samples by a factor of integer, we set interpolation factor equal to 1. So the total reduction rate of the time samples will be like,

$$Reduction\ Rate = \frac{decimation\ factor}{interpolation\ factor} = \frac{decimation\ factor}{1} = decimation\ factor$$

To test the code, we downsampled the input by factors 2, 3, 6 and 3.5 .

After listening both the original and downsampled versions with increasing factors, the sound quality difference became more obvious. With the reduction rate of 2, there was not a clear difference in listening experiences. But in the reduction with rate 6, the sound quality decreased significantly.

The spectrogram outputs of original and downsampled signals can be observed in the following figures.

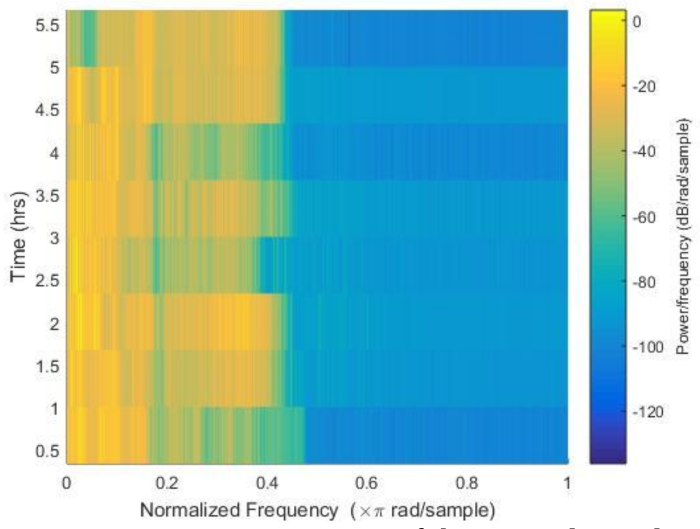


Figure 26. Spectrogram output of the original signal

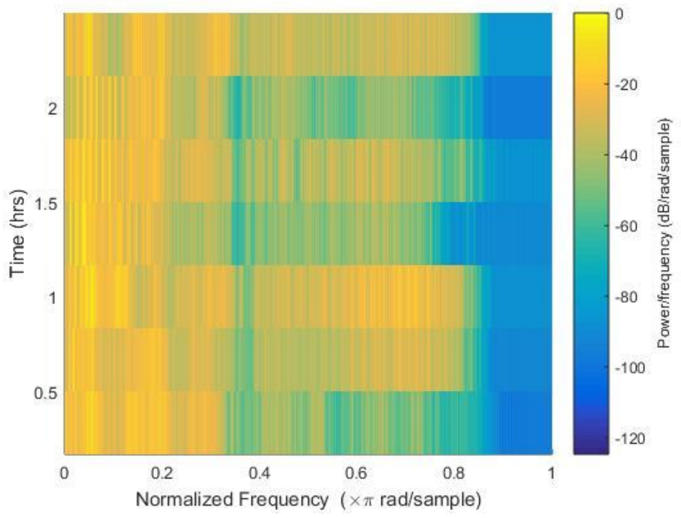


Figure 27. Spectrogram output of the decimated signal with ratio 2.

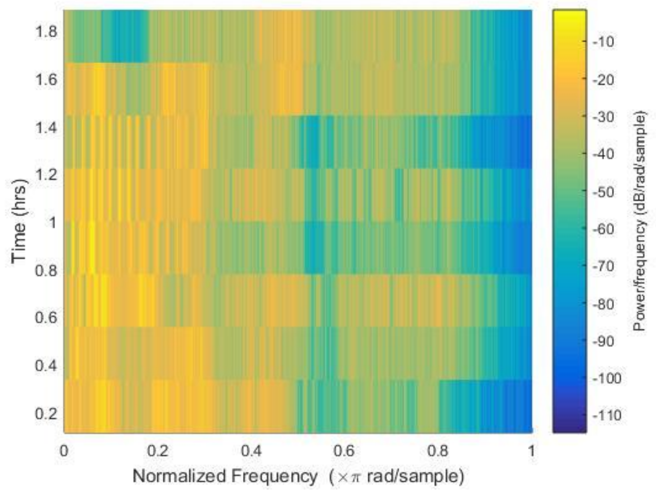


Figure 28. Spectrogram output of the decimated signal with ratio 3.

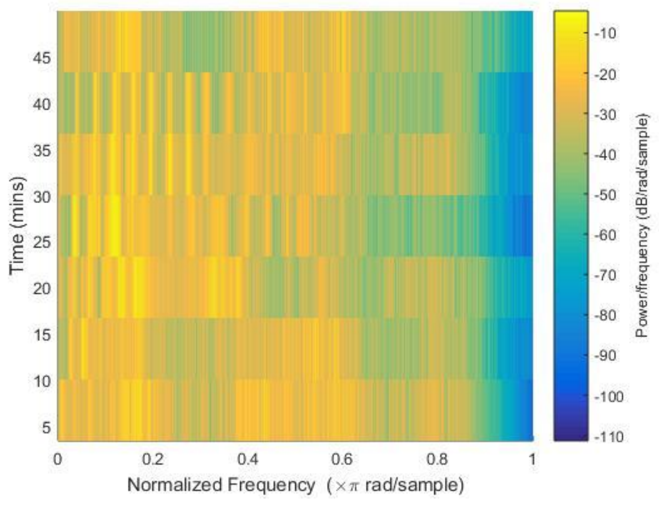


Figure 29. Spectrogram output of the decimated signal with ratio 6.

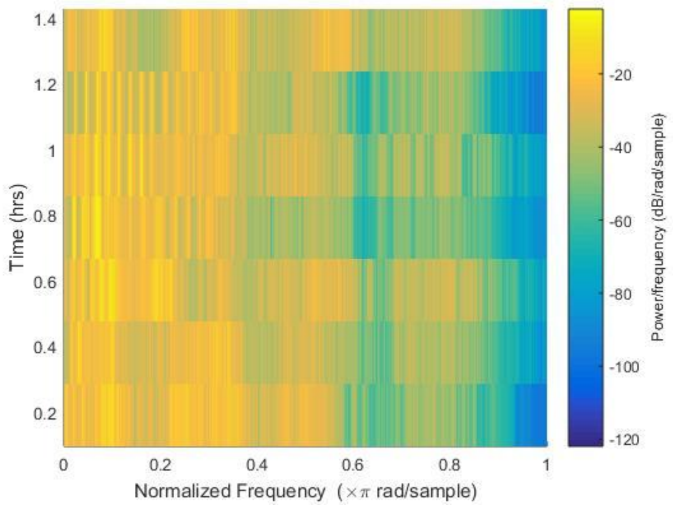


Figure 30. Spectrogram output of the decimated signal with ratio 3.5 (decimated by7, interpolated by 2).

The time domain representations of the original and downsampled signals are given below. It is obvious that the length of the time domain decreases with a ratio of downsampling ratio.

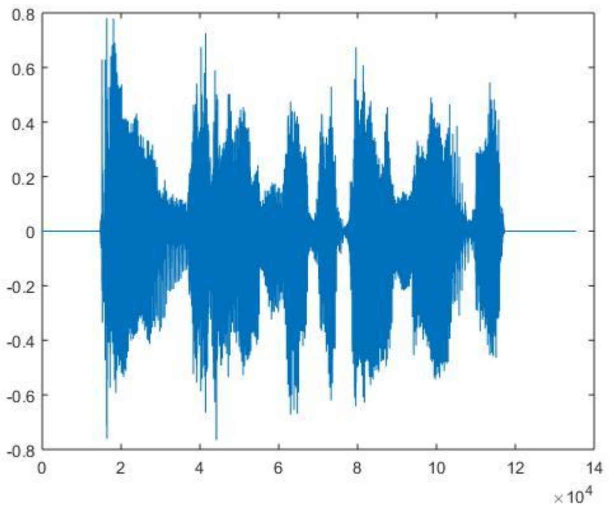


Figure 31. Spectrogram output of the original signal

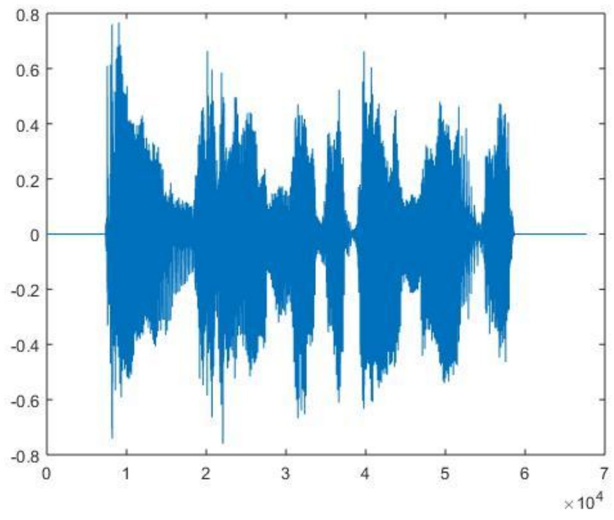


Figure 32. Spectrogram output of the decimated signal with ratio 2.

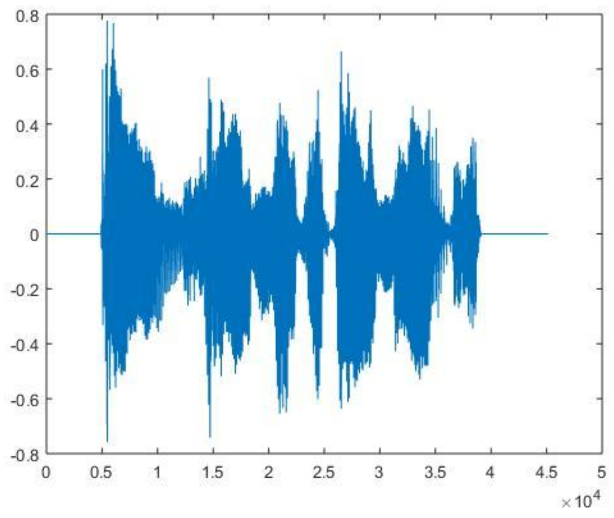


Figure 33. Spectrogram output of the decimated signal with ratio 3.

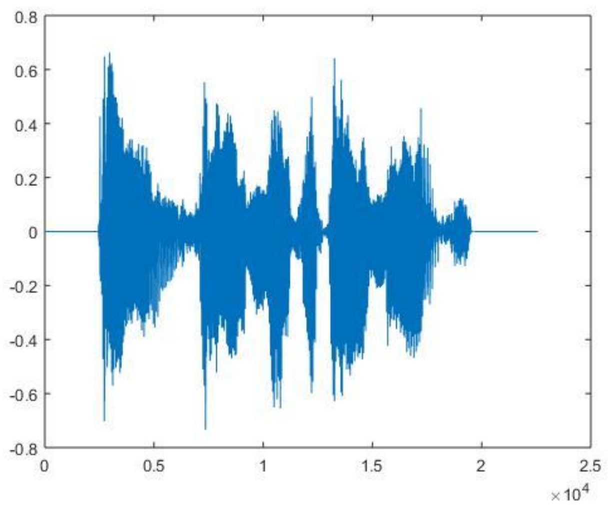


Figure 34. Spectrogram output of the decimated signal with ratio 6.

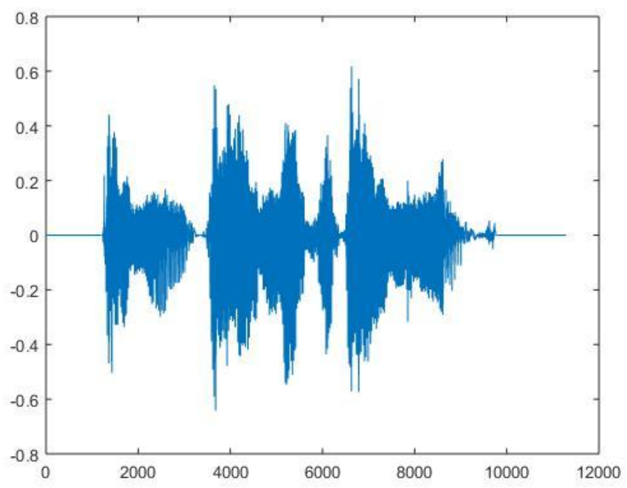


Figure 35. Spectrogram output of the decimated signal with ratio 12.

Decimation Factor	Number of Original Samples	Number of Decimated Samples	Reduction Rate
2	135408	67704	50%
3	135408	45136	33%
6	135408	22568	17%

Table 1. Number of samples for different decimation factors.

Part 2. Reducing the bits per sample on time domain

Reducing the number of bits/sample ratio of original signal is another important way to compress the file. This obviously helps us to compress our input but the loss of information is inevitable.

So, we investigated the quantization process of a mono audio file which has originally 16bit/sample signal and quantized it with 2, 4, 8 and 16 bits/sample systems.

As it can be seen from Table 2, for a mono audio file which has originally 16 bit/sec, there is no error in the 16 bits/sample case as there is no loss of information. But when we decrease the bit number per sample, the error begins to become a problem and the message is mostly distorted. In the case of 2 bits per sample, the message cannot even be realizable. So, there is a trade-off between the loss of information and number of bits used. If the information is low-tolerant to any kind of error, it is not sensible to apply this step. But, if the important issue is to compress the input as much as possible, then this method can be applied.

As in the table given below, there is not a serious loss of original energy until n=2. But when n=2, the message is almost unrecognizable due to the error.

# of bits/sample	Original Energy	Quantized Energy	Error Energy
2	0.0635	0.1361	0.0856
4	0.0635	0.0677	0.0052
6	0.0635	0.0638	3.24e-4
8	0.0635	0.0635	2.02e-5
16	0.0635	0.0635	0

Table 2. Energies of original and quantized signals.

The plot of the original signal is given below in Figure 36.

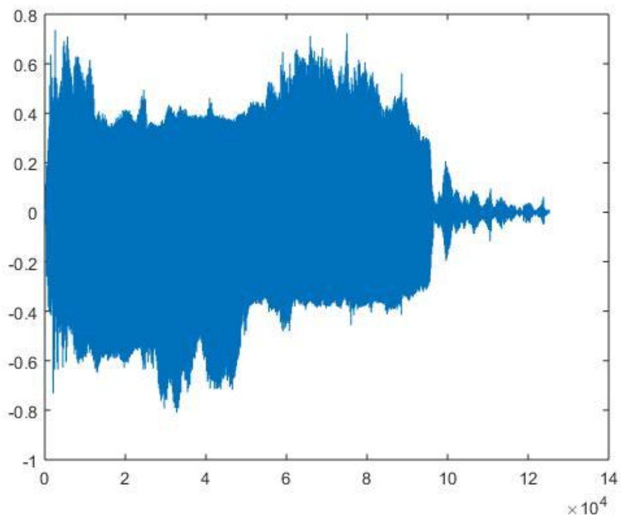


Figure 36. The original signal vs time

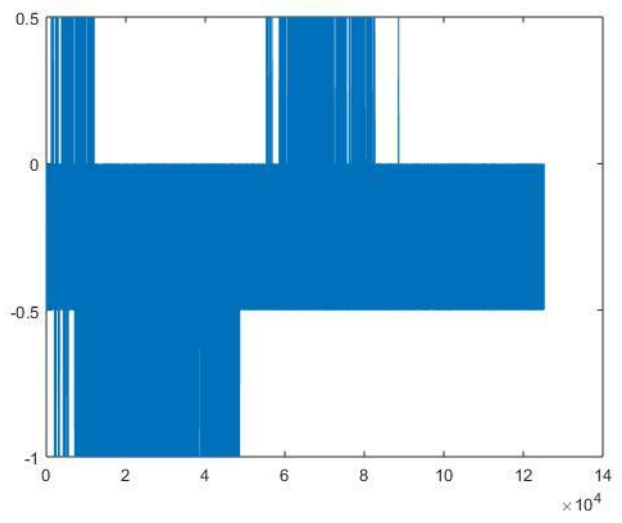


Figure 37. Quantized version of the signal with n=2.

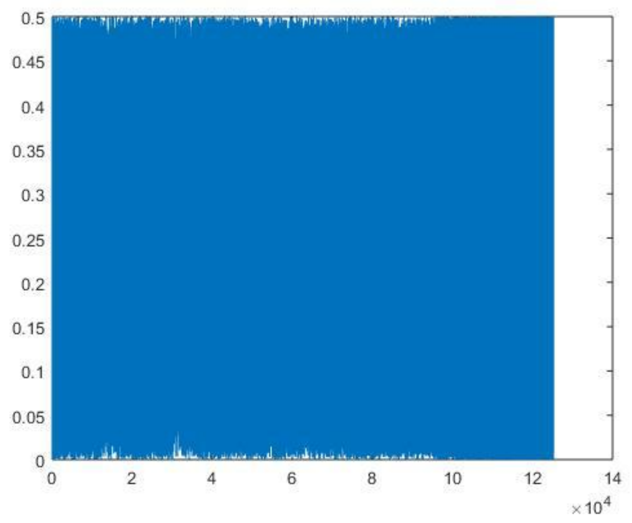


Figure 38. Error function when n=2.

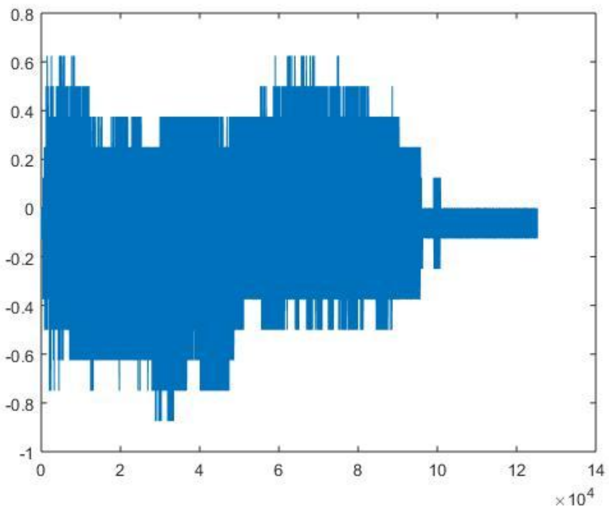


Figure 39. Quantized version of the signal with n=4.

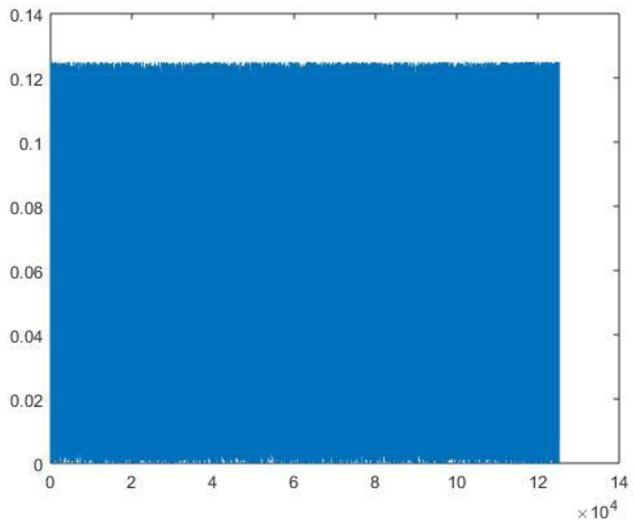


Figure 40. Error function when n=4.

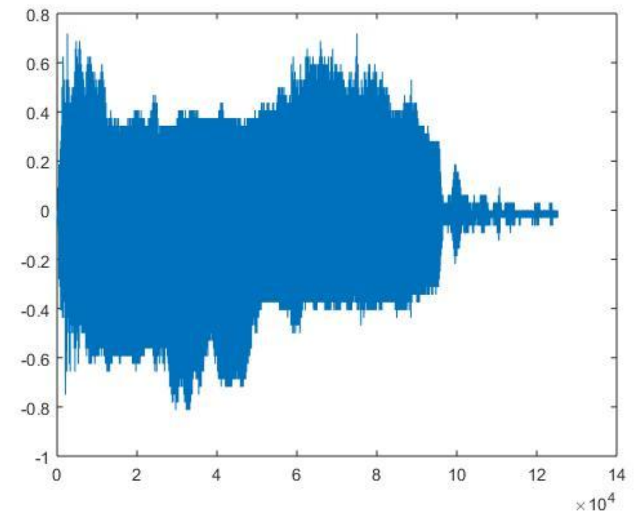


Figure 41. Quantized version of the signal with n=6.

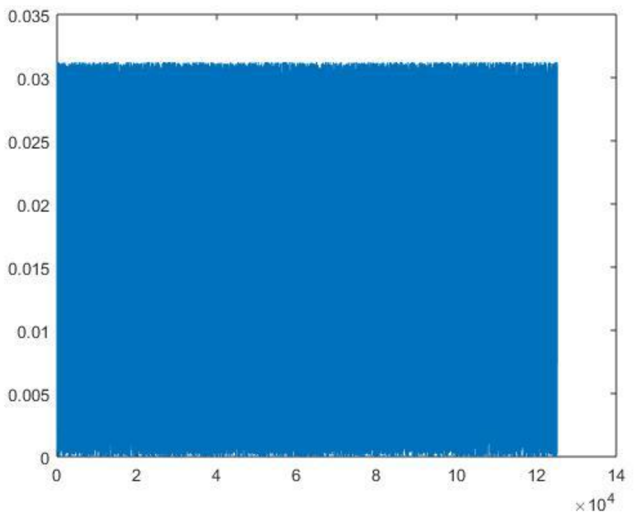


Figure 42. Error function when n=6.

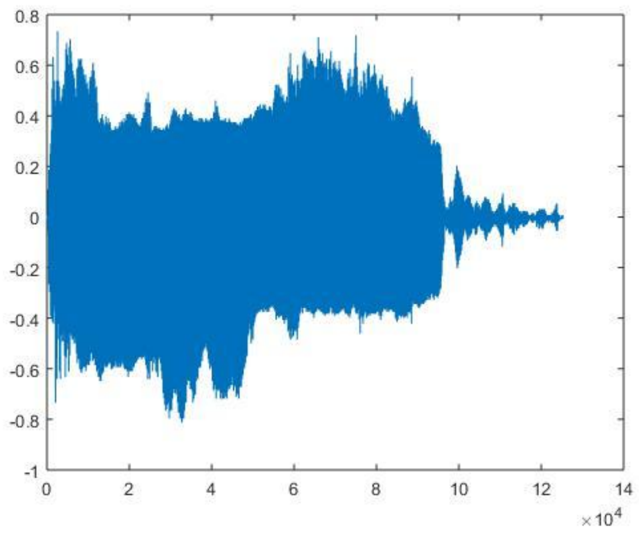


Figure 43. Quantized version of the signal with n=8.

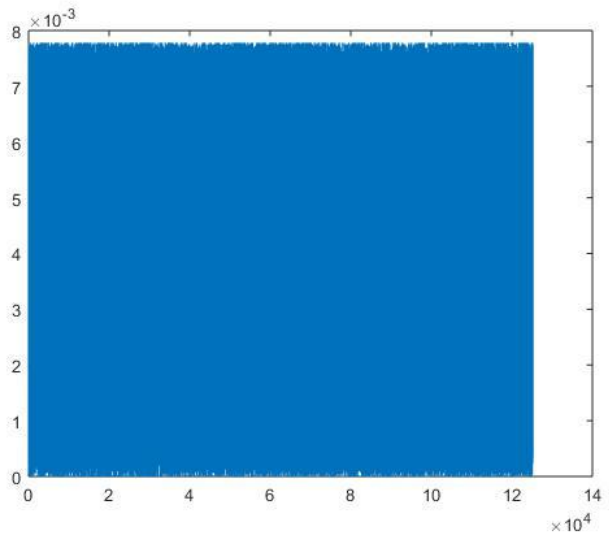


Figure 44. Error function when n=8.

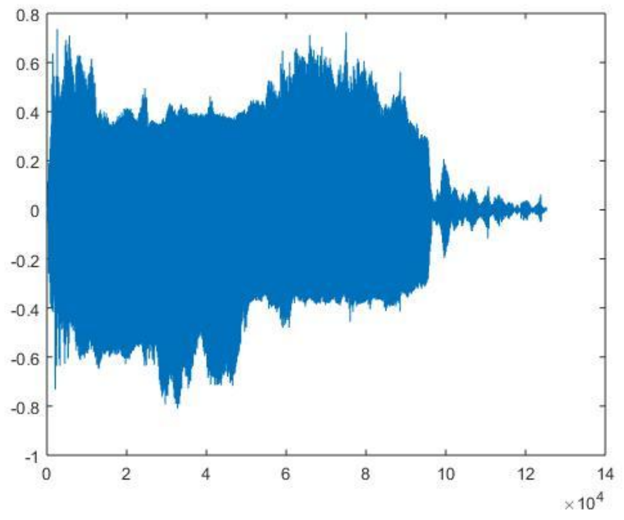


Figure 45. Quantized version of the signal with n=16.

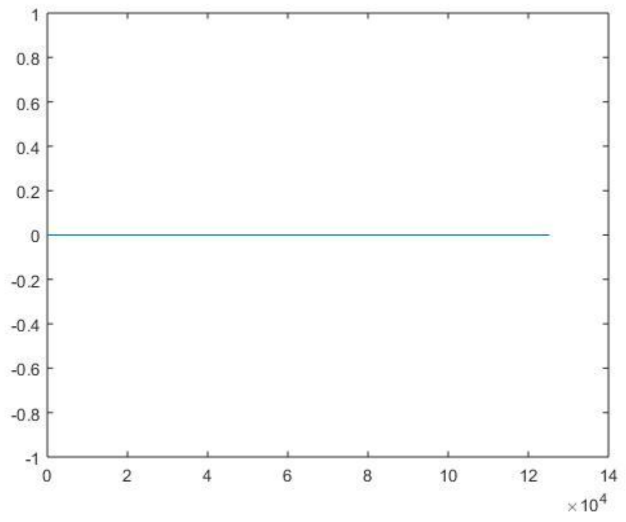


Figure 46. Error function when n=16.

As It can be seen from these figures, the error decreases and approaches to zero as the number of bits per sample increases. When it is equal to 16 (as in the original file), the error is equal to zero. (perfect reconstruction case)

But when n=6, the sound is reconstructed successfully. But n=4 gives the result with a serious distortion and noise. So, any number greater than 4 will be suitable to quantize the input signal for the file we used.

Part 3. Transform Coding

Instead of applying quantization in time domain, we can transform the input into frequency domain and then, eliminate some values below a threshold.

The methods to apply transform coding was used,

- Discrete Fourier Transform
- Discrete Cosine Transform

We apply these two methods with the window size defined in the code. The window size is an important parameter for the success of the system. As it gets larger, there will be more data be processed and so the system becomes less successful.

For window size=2000, the power coefficients due to the power of original signal, output signal and error signal can be observed in Table 3 when the frequency domain coefficients are quantized with DFT.

Threshold Value	Original Power	ISFT Output Power	Error
Max/5000	0.0635	0.0642	1.02e-7
Max/1000	0.0635	0.0642	6.44e-6
Max/200	0.0635	0.0641	7.12e-5
Max/50	0.0635	0.0637	5.07e-4
Max/10	0.0635	0.06	0.0042
Max/5	0.0635	0.0558	0.0084
Max/200 when the winlen is 5	0.0635	0.0635	4.8e-7

Table 3. Power Values of Original Signal, Output Signal and Error Signal.

It is obvious that when the threshold value increases, less coefficients are taken into account which causes a worse output and a larger error energy. So, the threshold value should not be increased too much to be able to recover signal successfully.

Another important parameter for the process is the window length. As given in the last row of the table above, we can balance the effect of larger threshold values with smaller window lengths.

When DFT operation is replaced with DCT, the following values for energy were taken. Note that the window length is taken as 2000 again.

Threshold Value	Original Power	IDCT Output Power	Error
Max/5000	0.0635	0.0642	1.43e-7
Max/1000	0.0635	0.0642	2.61e-6
Max/200	0.0635	0.0641	5.62e-5
Max/50	0.0635	0.0636	5.28e-4
Max/10	0.0635	0.0596	0.0046
Max/5	0.0635	0.0552	0.0089
Max/200 when winlen is 5	0.635	0.0635	5.79e-7

Table 4. Power Values of Original Signal, Output Signal and Error Signal.

The same facts of DCT can be said for DCT, too. When the threshold value increases, we take less coefficients into account which gives less accurate results. But, this bad effect can be compensated with lowering the window length.

The plots corresponding to different threshold values (max(dft of signal)/5 and max(dft of signal)/5000) in DFT method can be observed below.

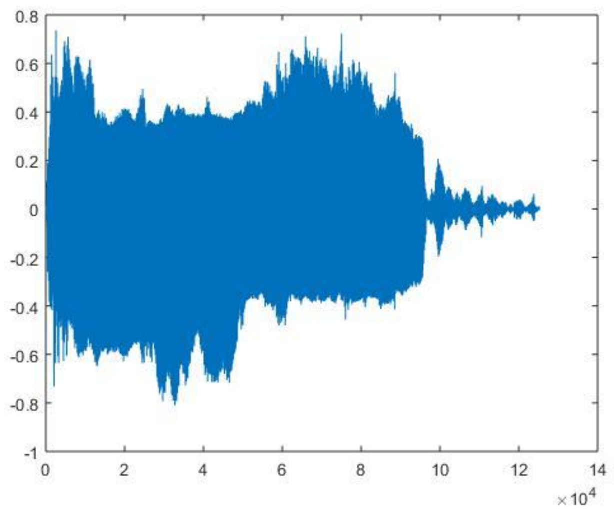


Figure 47. Plot of the original signal versus time

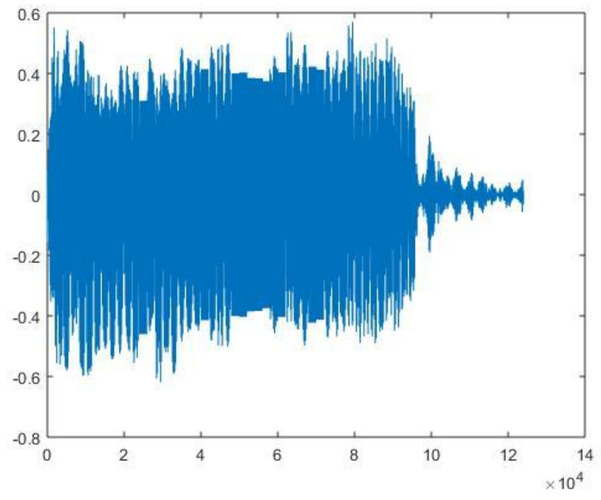


Figure 48. Plot of the quantized signal in DFT with the threshold value max(dft of signal)/5

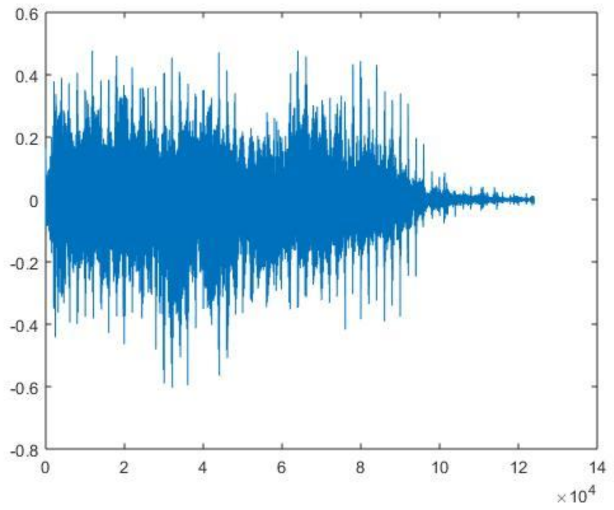


Figure 49. Error signal versus time when DFT is applied with the threshold value max(dft of signal)/5.

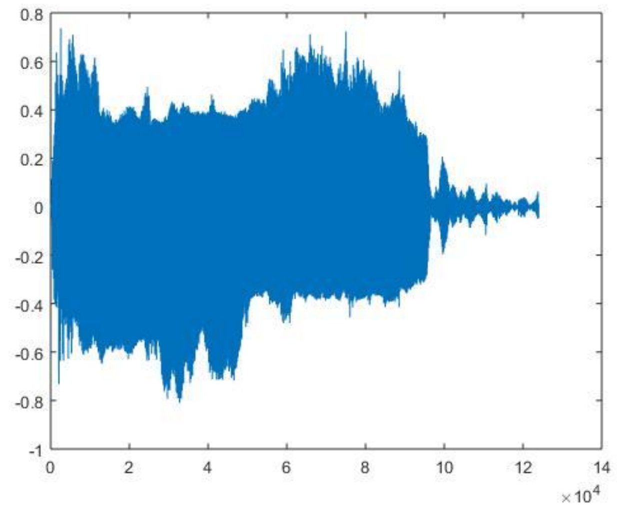


Figure 50. Plot of the quantized signal in DFT with the threshold value max(dft of signal)/5000

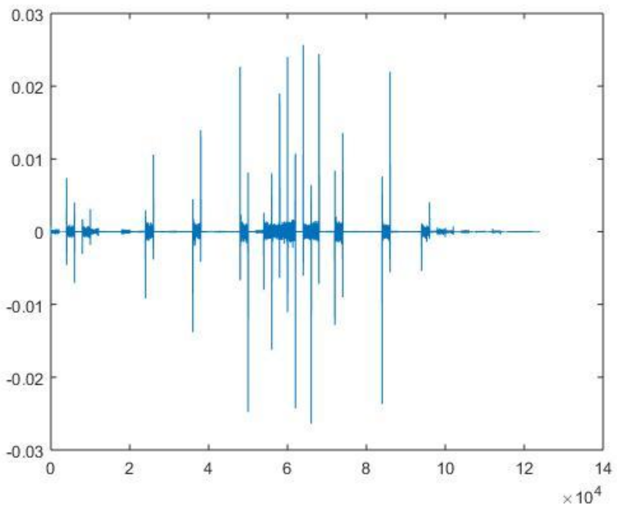


Figure 51. Error signal vs time when DFT is applied with the threshold value max(dft of signal)/5000

The same procedure is applied for DCT process for the threshold values of

max(dft of signal)/5 and max(dft of signal)/5000.

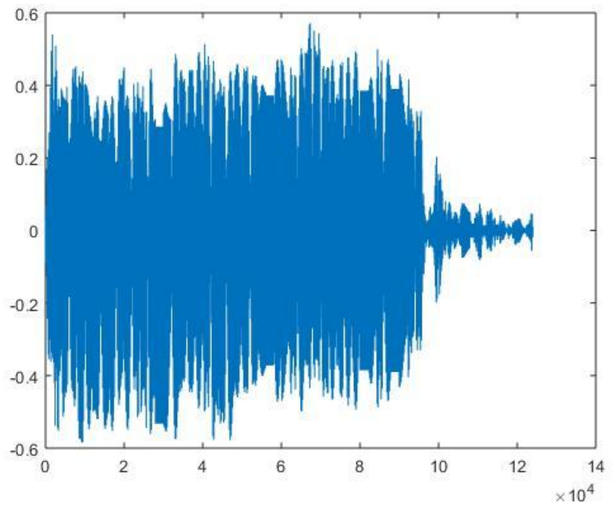


Figure 52. Plot of the quantized signal in DCT with the threshold value max(dct of signal)/5

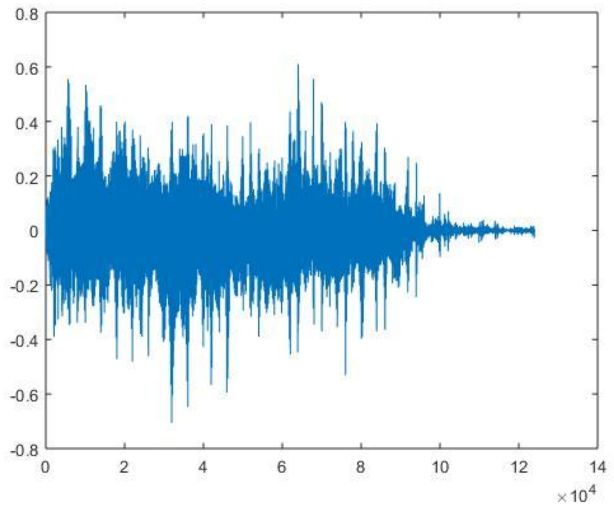


Figure 53. Error signal vs time when DCT is applied with the threshold value max(dct of signal)/5

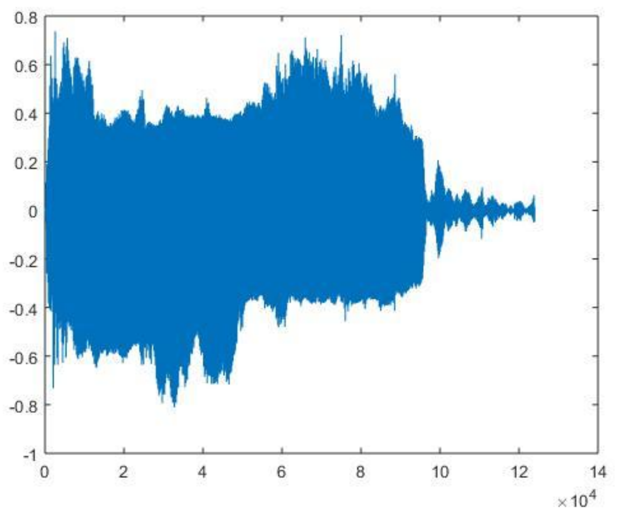


Figure 54. Plot of the quantized signal in DCT with the threshold value max(dct of signal)/5000

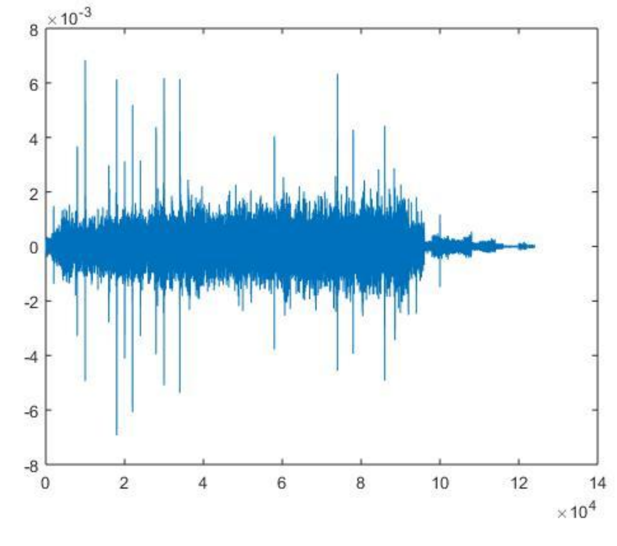


Figure 55. Error signal vs time when DCT is applied with the threshold value max(dct of signal)/5000.

As a result, to have a qualified transform coding, one should be careful about these two facts to optimize the process,

- Lowering the window length.
- Decreasing the threshold value.

Part 4. MPEG/Audio Encoder

The MPEG/Audio encoder algorithm quantizes the frequency domain coefficients. The number of bits is the limiting factor for our system. This algorithm is based on the psychoacoustic model of human hearing system. It is based on the idea of discarding some frequency components that human hears cannot recognize.

In this system, there are 4 main subsystems such that,

i. Prototype Filter Design
ii. Sub Band Filtering
iii. Spectrum Estimation
iv. Quantizatiton

I. Prototype Filter Design

In this step, first we created a 512-tap FIR low pass filter to become the prototype filter for the sub band filters. This filter's impulse response and frequency response can be observed below in Figure 56, 57 and 58.

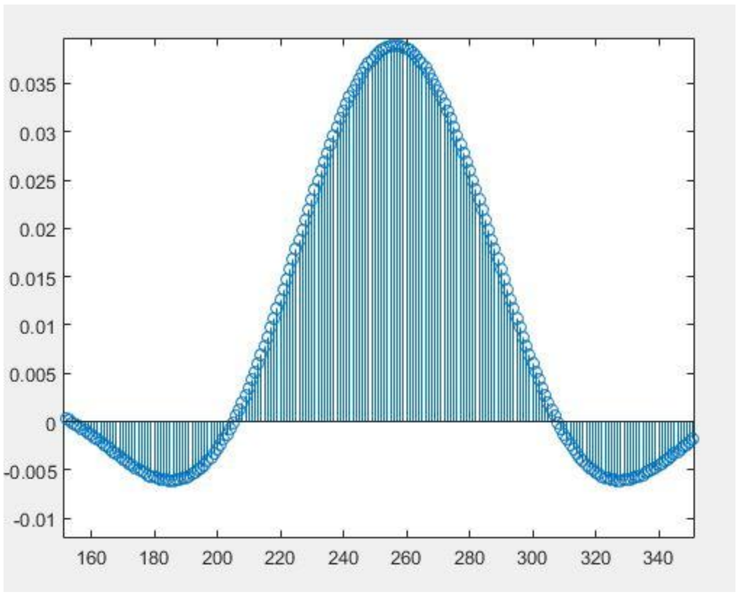


Figure 56. Impulse Response of Prototype Filter (h[n])

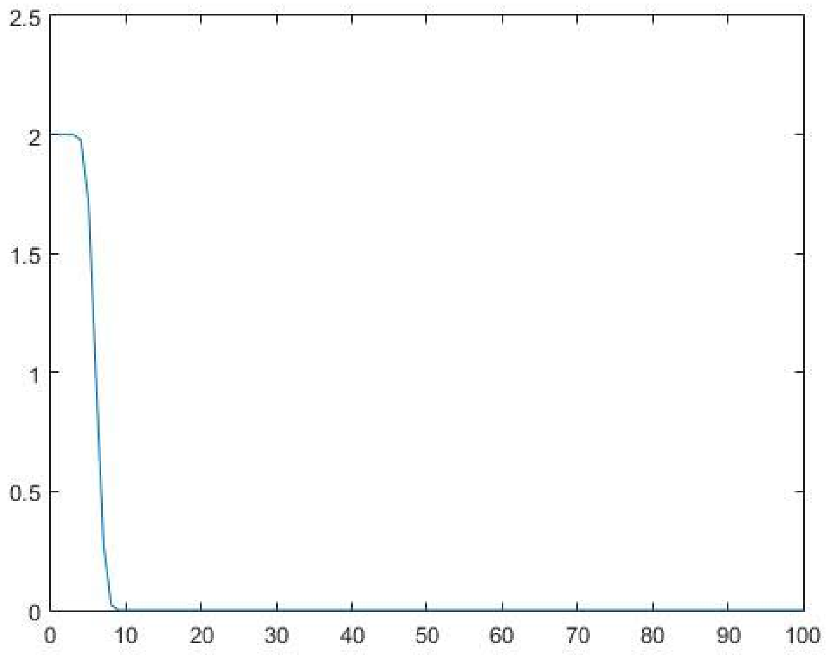


Figure 57. Magnitude Response of the Prototype Filter (|H(e^{jw})|)

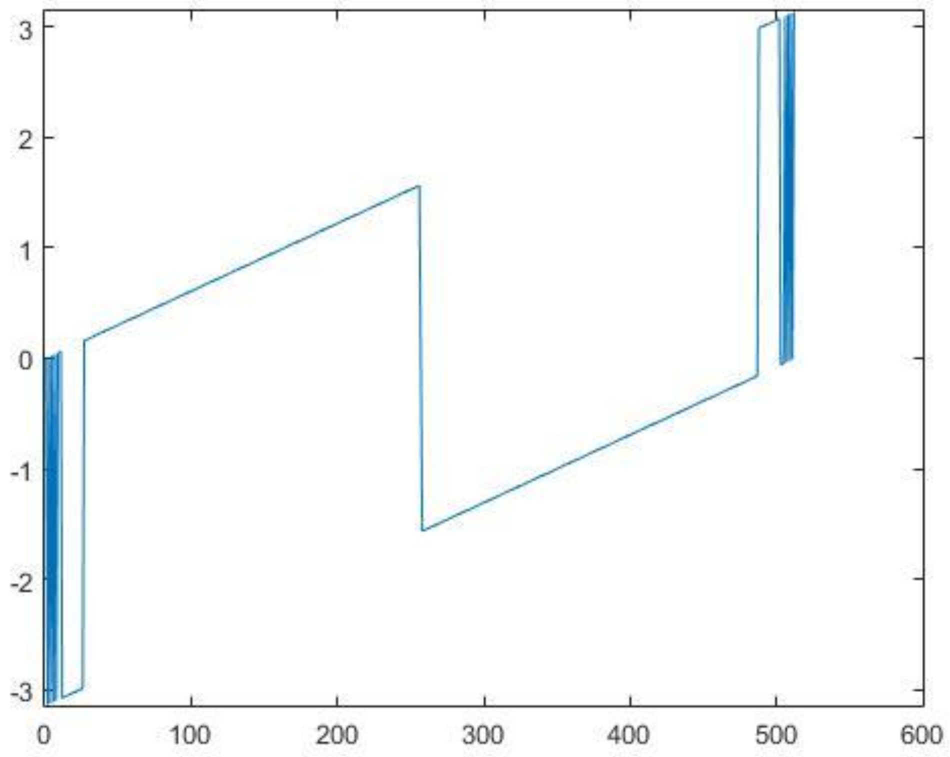


Figure 58. Phase Response of the Prototype Filter (|H(e^{jw})|)

II. Sub Band Filtering

What we created in the prototype filter design was a low pass filter. But our system requires these filters in a row with their shifted versions to create h0[n] to h31[n](sub band filtering). The full implementation is available in subband_filtering.m file. The frequency responses from h0[n] to h31[n] is shown in Figure 59.

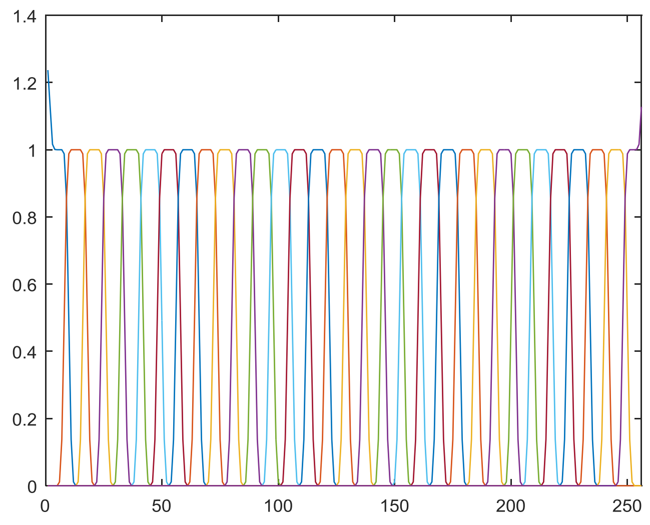


Figure 59. Frequency response of all subband filters.

III. Spectrum Estimation

This part is implemented in scaled_fft_db.m file. We first window the input signal with Hamming window and simply normalize the frequency domain (FFT) components such that the maximum point is 96 dB and take their magnitude values into account to send them into the psychoacoustic model as input. the spectrum estimation determines the masking level of the psychoacoustic model of the MPEG/Audio encoder. The scaled version of fft of input signal is shown in Figure 60.

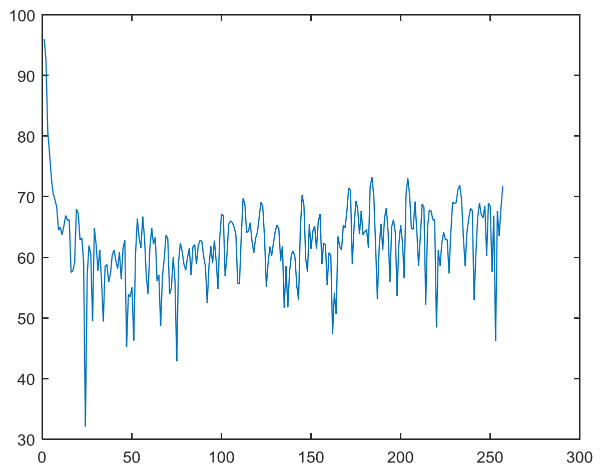


Figure 60. Scaled FFT in dB. of the input.

IV. Quantization

In this step, we conduct the quantization of the samples. We implement the quantization by using the formula:

$$q = \left\lfloor \left(Q_a \frac{s}{S_f} + Q_b \right) 2^{R-1} \right\rfloor$$

where s is the sample to quantize, Sf is the scaling factor, R is the number of bits for the sample and Qa, Qb are the quantization parameters. The code for quantization can be found in quantization.m file.

With different bitrates, different results were achieved. We tried to process a .wav file with 32, 96 and 128 bitrates. And the conclusion was not surprise for us as the number of bits increases, the sound quality of the output becomes better.

We took a sample .wav file whose time domain graph is given in Figure 61.

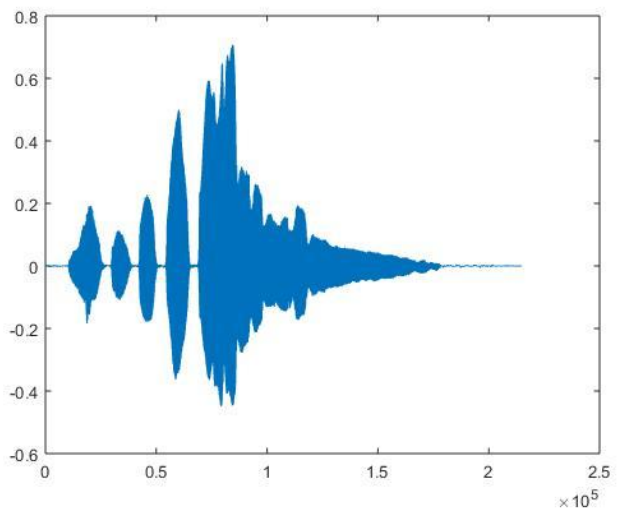


Figure 61. Input signal versus time

By processing this file with 32, 96 and 128 bitrates, the corresponding signals are achieved given in Figure 62, 63 and 64.

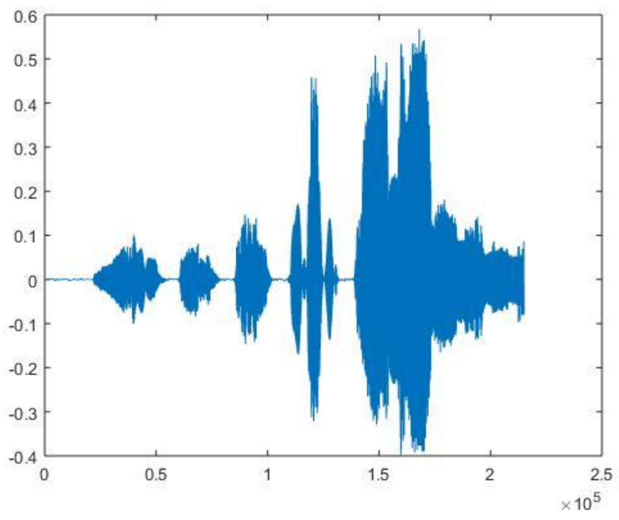


Figure 62. Output signal processed with 32 bitrates.

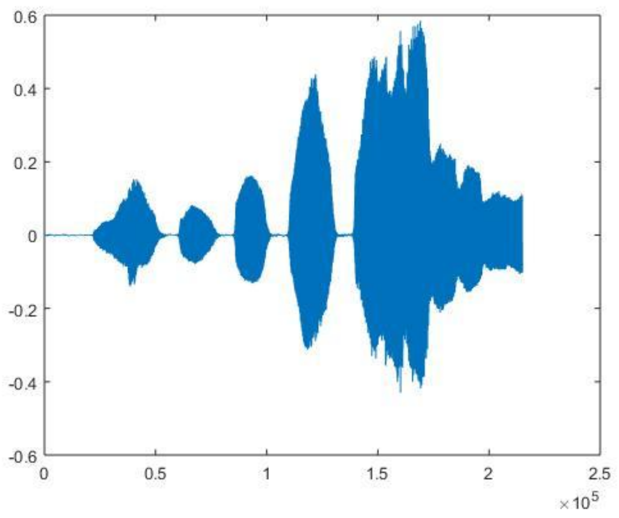


Figure 63. Output signal processed with 96 bitrates.

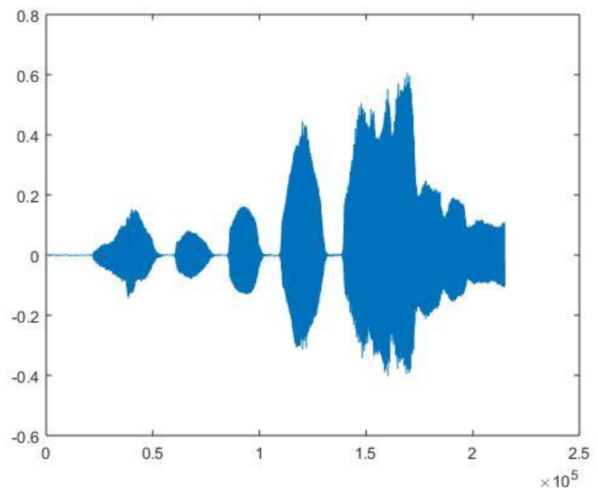


Figure 64. Output signal processed with 128 bitrates.

CONCLUSION

The purpose of this term project is to get familiar and implement the real-time applications of digital signal processing, such as quantization, STFT, decimation-in-time, DFT and DCT coding and MPEG/audio compression. We highly believe that the experience we got throughout the semester while doing this project will be very beneficial in our future careers as engineers.