19CCE212 - MULTI-RATE SIGNAL PROCESSING AND SOFTWARE-DEFINED RADIO

CODING ASSIGNMENT I

27/05/2022, Friday

Aim:

Develop a code for audio denoising using wavelets [Haar wavelet]. Analyse the effectiveness of denoising for varying levels of decomposition. Repeat the analysis for varying degrees of noise added to the clean audio signal.

Code:

```
import matplotlib.pyplot as plt # Provides an implicit way of plotting
import numpy as np # Support for large, multi-dimensional arrays and matrices
import scipy.fftpack as sf # Module to calculate discrete fast Fourier
transform
import sounddevice as sd # Play and record NumPy arrays containing audio
signals

# Sketch the spectrum of the signal in frequency domain:
def plot_frequency_domain(signal, plot_name):

    print("\n"); signal_fft = np.fft.fft(signal)
    plt.xlabel("Frequency in Hertz")
    plt.ylabel("A(f) in Volts")
    plt.title("" + str(plot_name) + " in Frequency Domain")
    plt.stem(np.arange(0, len(signal_fft)), signal_fft)
    plt.grid(True)
    plt.show()

# Down-sample the given signal by a factor of "m":-
def down_sample(signal, m):
    down_sampler = []
    # Copy the element value from original signal to down sampler array with
the increment value set to "m":
    for i in range(0, len(signal), m):
        down_sampler.append(signal[i])
    return down_sampler

# Up-sample the given signal by a factor of "l":-
def up_sample(signal, l):
    up_sampler = []
    for i in range(len(signal)):
        up_sampler.append(signal[i]) # Copy the element value from original
signal to up sampler array
        if i != len(signal):
```

```
            for k in range(l-1):
                up_sampler.append(0) # Insert "l-1" zeros between the elements
of the array
        else:
            break
    return up_sampler

# Driver Code: main(); Execution starts here.

# Details for Sound Recording:-
Fs = int(input("\nEnter the sampling frequency in kilo-hertz: "))
Fs = Fs * 1000 # Fs = 48,000 Hz
t = int(input("Enter the time duration in seconds: "))
n = np.arange(0, t, 1/Fs) # Return evenly spaced values within a given
interval

# Give information about available devices and find the default input/output
device(s):-
print("\nThe list of audio devices connected to your system is as follows:\n"
+ str(sd.query_devices()))
print("\nNote: > indicates the default input device and < indicates the
default output device respectively - " + str(sd.default.device))

# Record audio data from your sound device into a NumPy array:-
print("\nRecording has started.")
x = sd.rec(int(t*Fs), Fs, 1, blocking = 'True')
#sd.wait()
x = x.flatten(); # Return a copy of the array (matrix) collapsed into one
dimension.
#t = np.arange(0,d,1/Fs)
#x = np.sin(2*3.14*2000*t)
print("Recording has stopped.")

# Play back a NumPy array containing audio data:-
print("Recording is being played...")
sd.play(x,Fs)

title = "Recorded Sound"
# Plot the Recorded Wave:-
plt.xlabel('Number of Frames (Sampling Frequency * Time Duration)')
plt.ylabel('Amplitude')
plt.title(title)
plt.plot(x)
plt.grid(True)
plt.show()
plot_frequency_domain(x, title)

decomposition = int(input("\nEnter the number of levels of decomposition: "))
```

```
# Generate a Noise:- (Draw random samples from a normal distribution)
y = np.random.normal(0, 0.2, np.size(x)); # Additive Wired Gaussian Noise
x = x + y
plt.title('Histogram Representation - Gaussian Noise')
plt.hist(y) # Plot a Histogram
plt.grid(True) # Configure the grid lines
plt.show()
print("Noisy signal is being played...")
sd.play(x,Fs)

# Plot the Noisy Signal:-
plt.subplot(2,1,1)
plt.xlabel('Time (in Seconds)')
plt.ylabel('Amplitude')
plt.title('Noisy Sinusoidal Wave')
plt.plot(n, x)
plt.tight_layout() # Adjust the padding between and around subplots
plt.grid(True) # Configure the grid lines

# Perform Spectral Analysis:-
X_f = abs(sf.fft(x)) # Return the absolute value of the fast Fourier transform
l = np.size(x)
fr = (Fs/2)*np.linspace(0,1,int(l/2))
xl_m = (2/l)*abs(X_f[0:np.size(fr)]); # Compute Magnitude Spectrum

# Plot Magntiude Spectrum:
plt.subplot(2,1,2)
plt.title('Spectrum of Noisy signal')
plt.xlabel('Frequency (in Hertz)')
plt.ylabel('Magnitude (in dB)')
plt.plot(fr,20*np.log10(xl_m))
plt.tight_layout() # Adjust the padding between and around subplots
plt.grid(True) # Configure the grid lines

plt.show() # Display all open figures

h1= [1/np.sqrt(2), 1/np.sqrt(2)]; g1= [1/np.sqrt(2), 1/np.sqrt(2)] # Same
coefficients
h2= [0, 0]; g2 = [0, 0] # Complementary coefficients
l_m_factor_expander = 2

for i in range(decomposition):
    # Analyze:-
    a = np.convolve(x, h1); p = np.convolve(x, h2)
    b = down_sample(a, l_m_factor_expander); q = down_sample(p,
l_m_factor_expander)
```

```
    # Synthesize:-
    c = up_sample(b, l_m_factor_expander); r = up_sample(q,
l_m_factor_expander)
    d = np.convolve(c, g1); s = np.convolve(r, g2)

    # Add:-
    x = np.add(d, s)

title = "Denoised Audio"
plt.xlabel('Number of Frames (Sampling Frequency * Time Duration)')
plt.ylabel('Amplitude')
plt.title(title);
plt.plot(x)
plt.grid(True)
plt.show()

print("Denoised audio is being played...")
sd.play(x,Fs)
plot_frequency_domain(x, title)
```

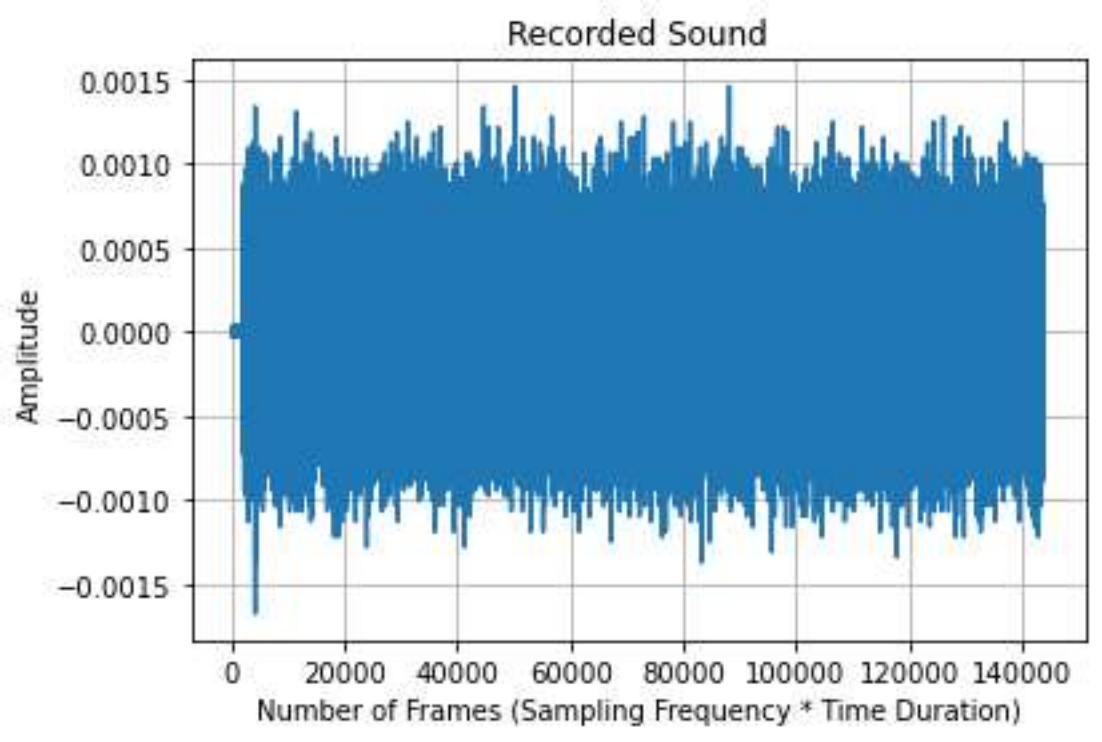
Result:

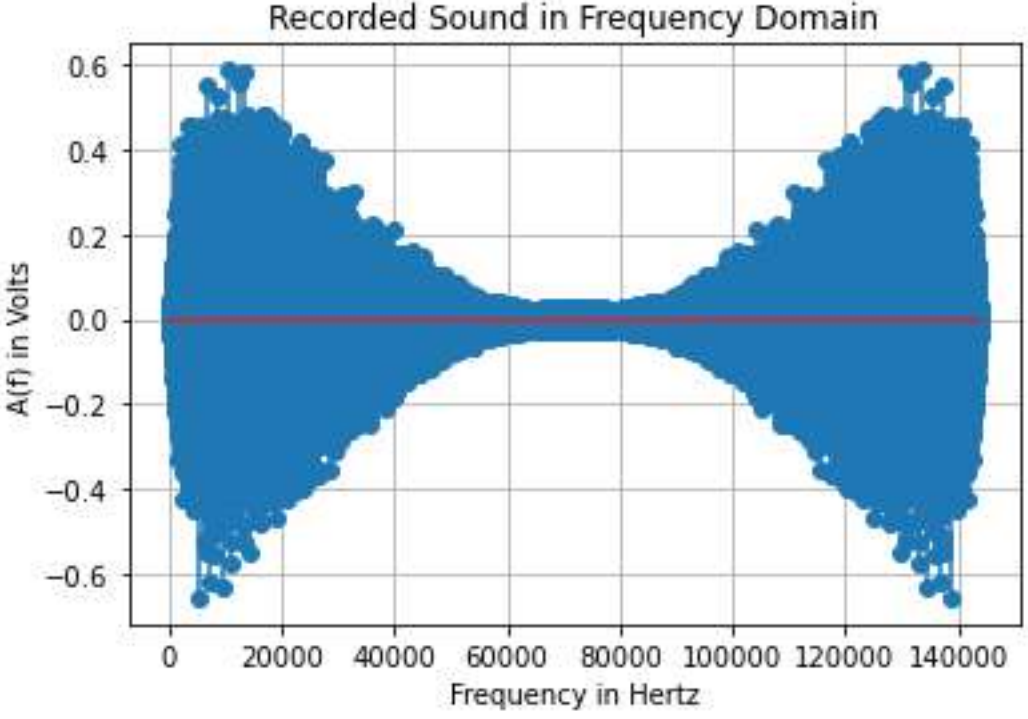
Thus, developed a code for audio denoising using the Haar wavelet technique which can be implemented in multi-rate signal processing applications. All simulation results were verified successfully.

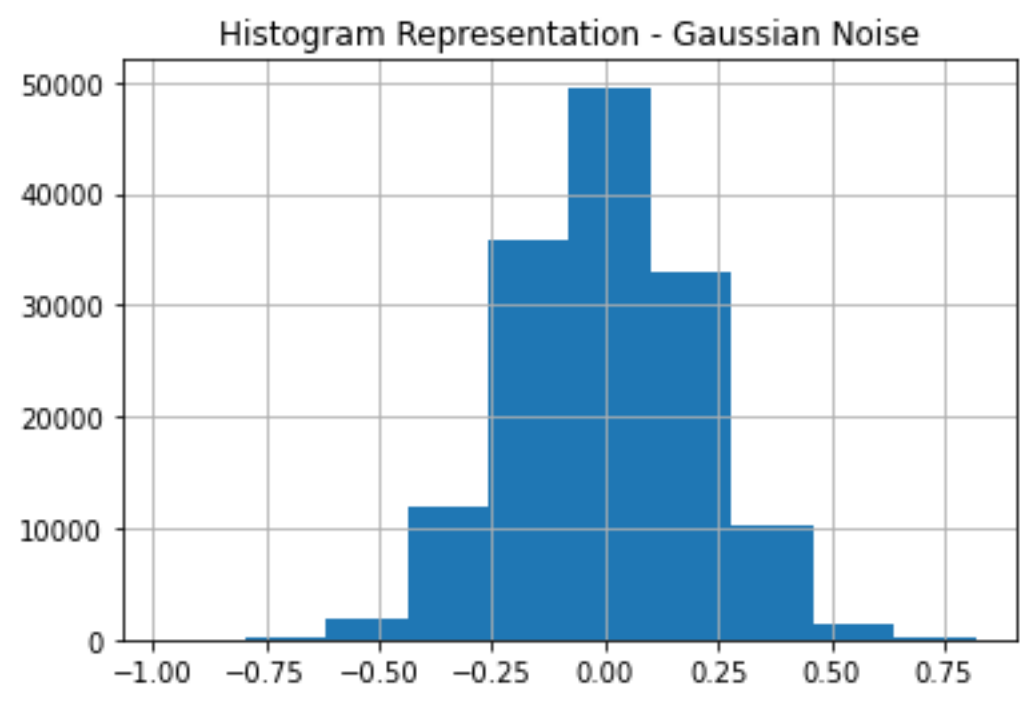
```
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

Restarting kernel...
```

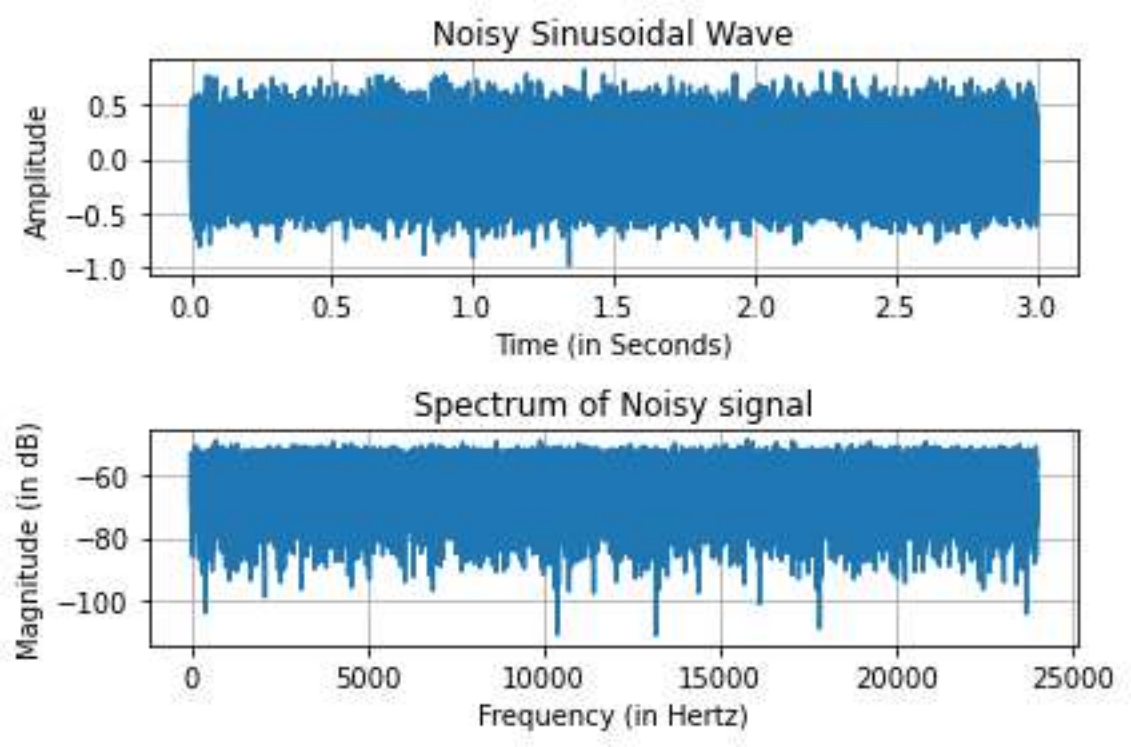
```
In [1]:         'E:/Plan B/Amrita Vishwa Vidyapeetham/Subject Materials/Semester IV/
19CCE212 - Multi-Rate Signal Processing and Software Defined Radio/Assignments/Audio
Denoising/Audio_Denoising_Code_Record.py'      ='E:/Plan B/Amrita Vishwa Vidyapeetham/
Subject Materials/Semester IV/19CCE212 - Multi-Rate Signal Processing and Software Defined
Radio/Assignments/Audio Denoising'


Enter the sampling frequency in kilo-hertz: 48

Enter the time duration in seconds: 3

The list of audio devices connected to your system is as follows:
   0 Microsoft Sound Mapper - Input, MME (2 in, 0 out)
>  1 Microphone (Realtek(R) Audio), MME (2 in, 0 out)
   2 Microsoft Sound Mapper - Output, MME (0 in, 2 out)
<  3 Headphones (Realtek(R) Audio), MME (0 in, 2 out)
   4 Speakers (Realtek(R) Audio), MME (0 in, 2 out)
   5 Primary Sound Capture Driver, Windows DirectSound (2 in, 0 out)
   6 Microphone (Realtek(R) Audio), Windows DirectSound (2 in, 0 out)
   7 Primary Sound Driver, Windows DirectSound (0 in, 2 out)
   8 Headphones (Realtek(R) Audio), Windows DirectSound (0 in, 2 out)
   9 Speakers (Realtek(R) Audio), Windows DirectSound (0 in, 2 out)
  10 Headphones (Realtek(R) Audio), Windows WASAPI (0 in, 2 out)
  11 Speakers (Realtek(R) Audio), Windows WASAPI (0 in, 2 out)
  12 Microphone (Realtek(R) Audio), Windows WASAPI (2 in, 0 out)
  13 Speakers 1 (Realtek HD Audio output with SST), Windows WDM-KS (0 in, 2 out)
  14 Speakers 2 (Realtek HD Audio output with SST), Windows WDM-KS (0 in, 2 out)
  15 PC Speaker (Realtek HD Audio output with SST), Windows WDM-KS (2 in, 0 out)
  16 Headphones 1 (Realtek HD Audio 2nd output with SST), Windows WDM-KS (0 in, 2 out)
  17 Headphones 2 (Realtek HD Audio 2nd output with SST), Windows WDM-KS (0 in, 2 out)
  18 PC Speaker (Realtek HD Audio 2nd output with SST), Windows WDM-KS (2 in, 0 out)
  19 Stereo Mix (Realtek HD Audio Stereo input), Windows WDM-KS (2 in, 0 out)
  20 Microphone (Realtek Digital Microphone), Windows WDM-KS (2 in, 0 out)
  21 Microphone (Realtek HD Audio Mic input), Windows WDM-KS (2 in, 0 out)

Note: > indicates the default input device and < indicates the default output device
respectively - [1, 3]

Recording has started.
Recording has stopped.
Recording is being played...


C:\ProgramData\Anaconda3\lib\site-packages\numpy\ma\core.py:3375: ComplexWarning: Casting
complex values to real discards the imaginary part
  _data[indx] = dval
```

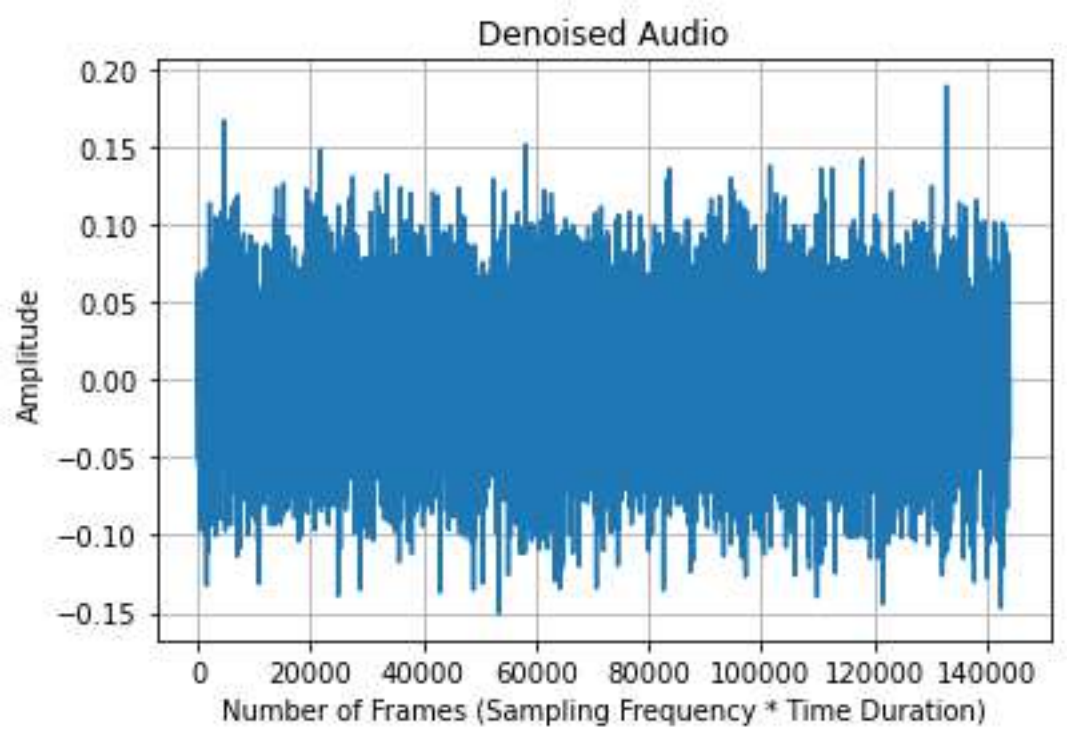
```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py:102: ComplexWarning:
Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py:102: ComplexWarning:
Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)


Enter the number of levels of decomposition: 50
Noisy signal is being played...
Denoised audio is being played...


C:\ProgramData\Anaconda3\lib\site-packages\numpy\ma\core.py:3375: ComplexWarning: Casting
complex values to real discards the imaginary part
  _data[indx] = dval
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py:102: ComplexWarning:
Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\_asarray.py:102: ComplexWarning:
Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)

In [2]:
```

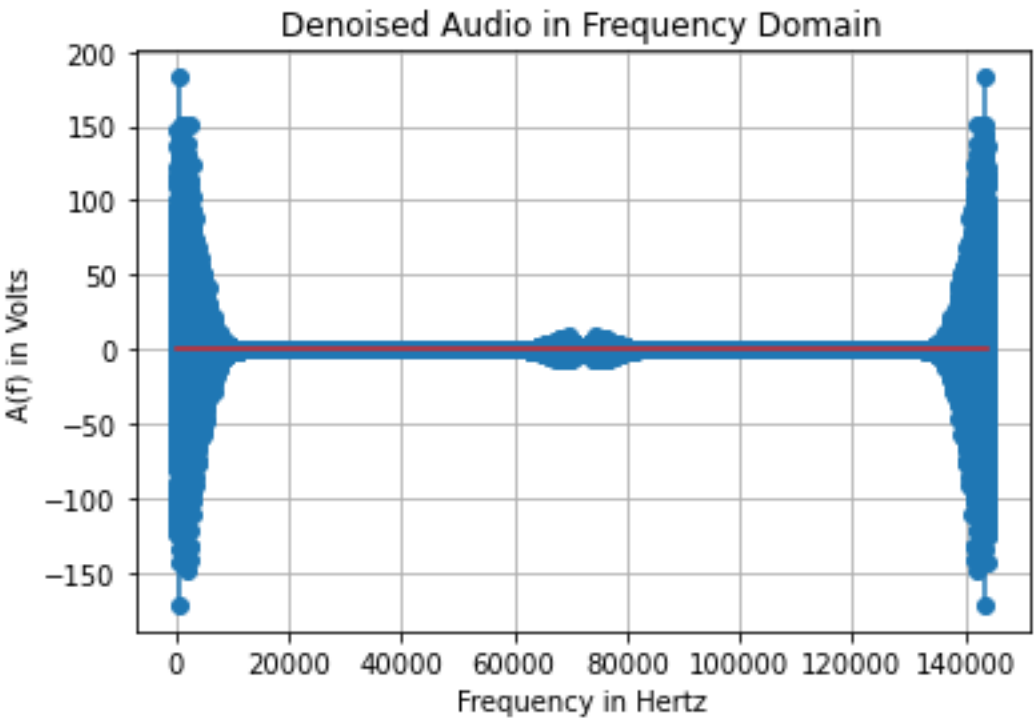

Recorded Sound
0.0015
0.0010
0.0005
0.0000
−0.0005
−0.0010
−0.0015
Amplitude
0
20000
40000
60000
80000
100000
120000
140000
Number of Frames (Sampling Frequency * Time Duration)


Recorded Sound in Frequency Domain
A(f) in Volts
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
20000
40000
60000
80000
100000
120000
140000
Frequency in Hertz


Histogram Representation - Gaussian Noise
50000
40000
30000
20000
10000
0
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75


Noisy Sinusoidal Wave
Amplitude
0.5
0.0
−0.5
−1.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Time (in Seconds)
Spectrum of Noisy signal
Magnitude (in dB)
−60
−80
−100
0
5000
10000
15000
20000
25000
Frequency (in Hertz)


Denoised Audio
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
Amplitude
0
20000
40000
60000
80000
100000
120000
140000
Number of Frames (Sampling Frequency * Time Duration)


Denoised Audio in Frequency Domain
A(f) in Volts
Frequency in Hertz
200
150
100
50
0
−50
−100
−150
0
20000
40000
60000
80000
100000
120000
140000