B. Deepak Kumar

AIE21028 ¶

LAB - 4

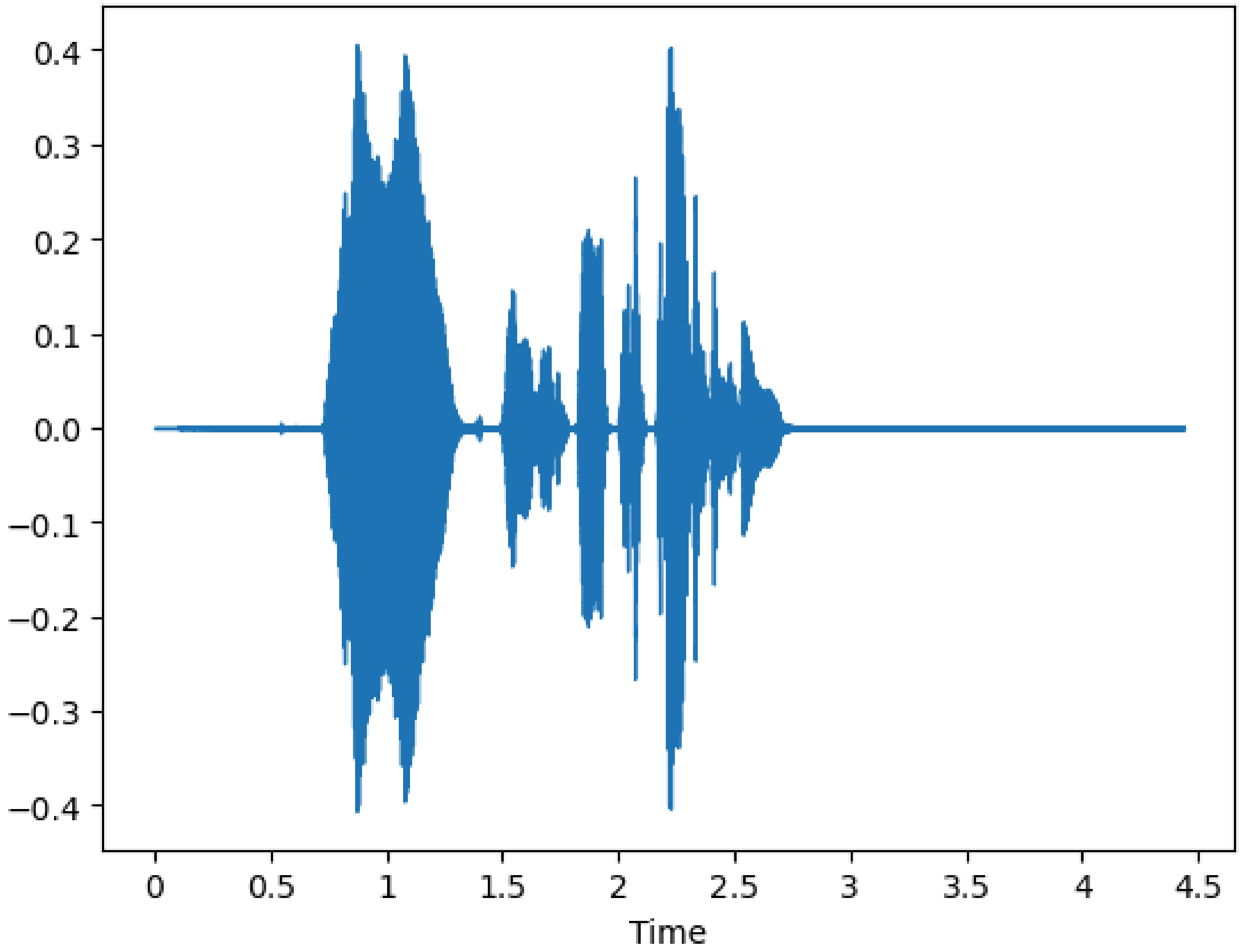
A1. Use numpy.fft.fft() to transform the speech signal to its spectral domain. Please plot the amplitude part of the spectral components and observe it.

In [82]:

```
import numpy as np
import librosa
import matplotlib.pyplot as plt
import IPython.display as ipd
import scipy.signal as signal
import scipy.io.wavfile as wavfile
from glob import glob
import seaborn as sns
from scipy.signal import spectrogram

```

In [22]:
```
y, sr = librosa.load('AISPS.wav')
librosa.display.waveshow(y)

```

Out[22]: <librosa.display.AdaptiveWaveplot at 0x1eb33958c40>


0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Time

In [23]:
```
a = glob('AISPS.wav')
ipd.Audio(a[0])
```

Out[23]:

0:04 / 0:04

In [27]:
```
# Use numpy.fft.fft() to transform the speech signal to its spectral domain
fft_result = np.fft.fft(y)
print("after fft:")
ipd.display(ipd.Audio(fft_result, rate=sr))
```

```
after fft:

C:\Users\saide\anaconda3\lib\site-packages\IPython\lib\display.py:159: ComplexWarnin
g: Casting complex values to real discards the imaginary part
  data = np.array(data, dtype=float)
```

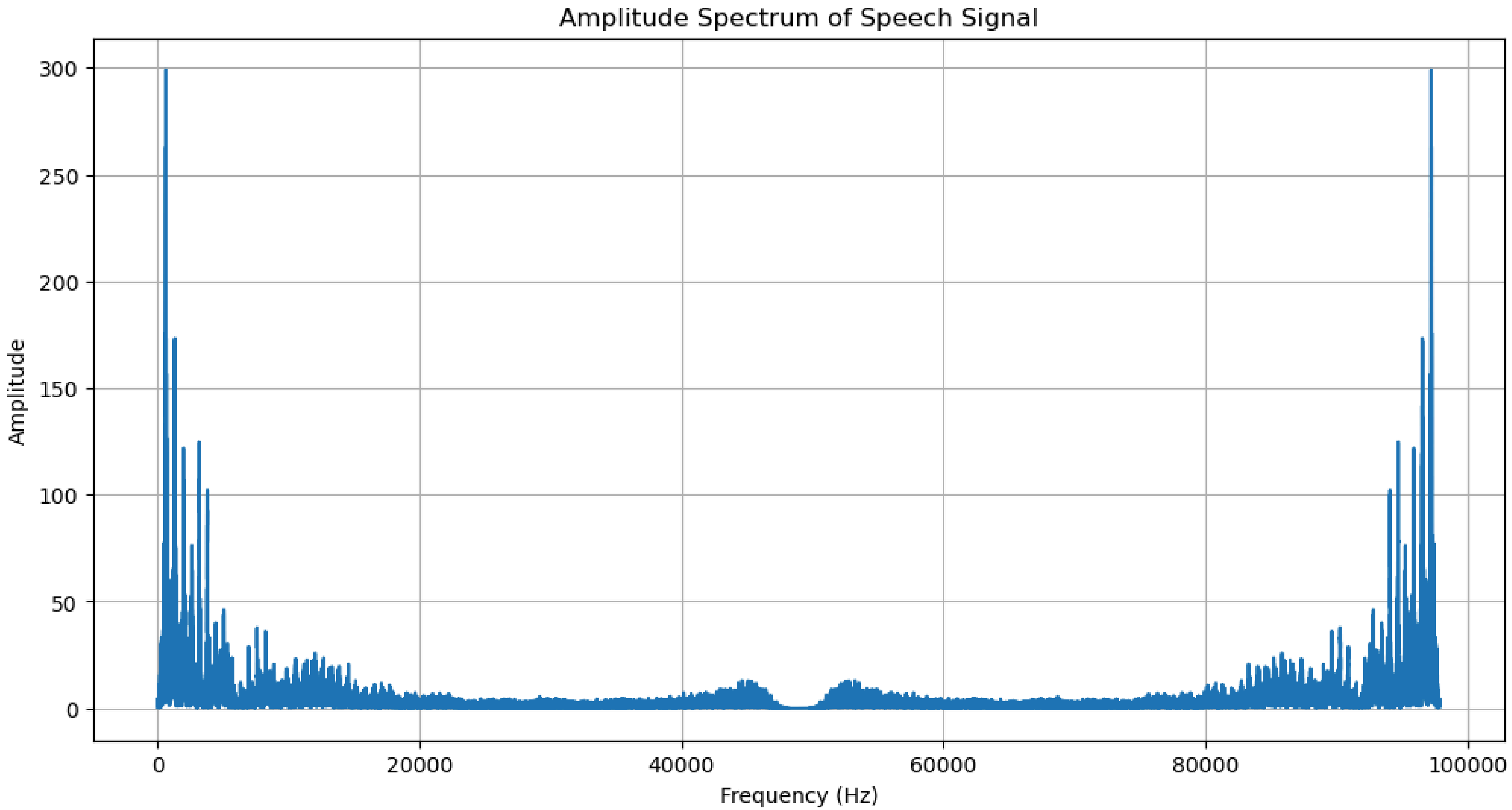
0:04 / 0:04

In [29]:
```
# Calculate the amplitude spectrum (absolute values of the complex numbers)
amplitude_spectrum = np.abs(fft_result)
print("amplitude spectrum")
ipd.display(ipd.Audio(amplitude_spectrum, rate=sr))
```

```
amplitude spectrum
```

0:04 / 0:04

In [8]:
```
plt.figure(figsize=(12, 6))
plt.plot(amplitude_spectrum)
plt.title('Amplitude Spectrum of Speech Signal')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
```



A2. Use numpy.fft.ifft() to inverse transform the frequency spectrum of the speech signal from frequency domain to time domain. Compare the generated time domain signal with the original signal.

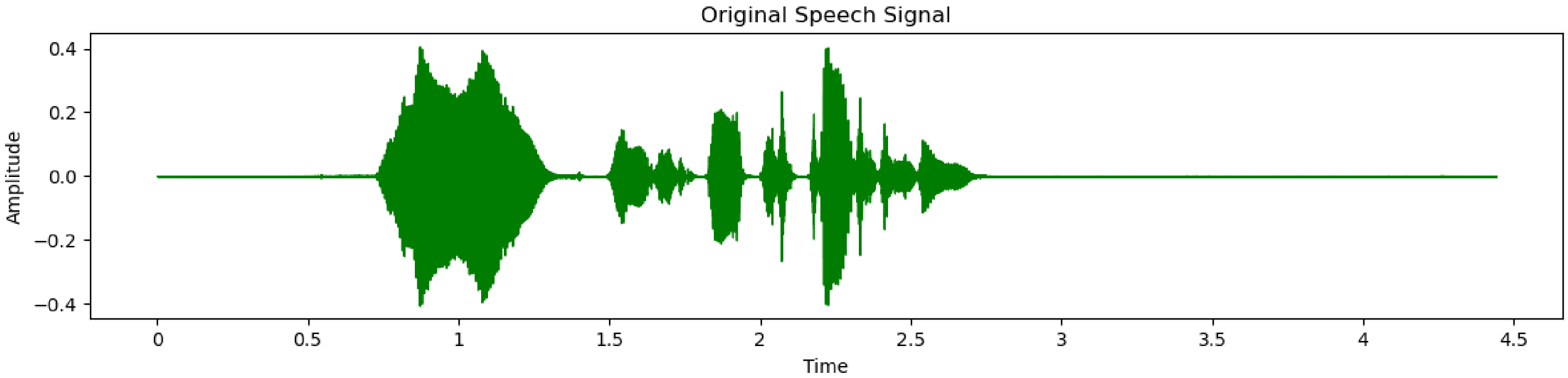
In [30]:

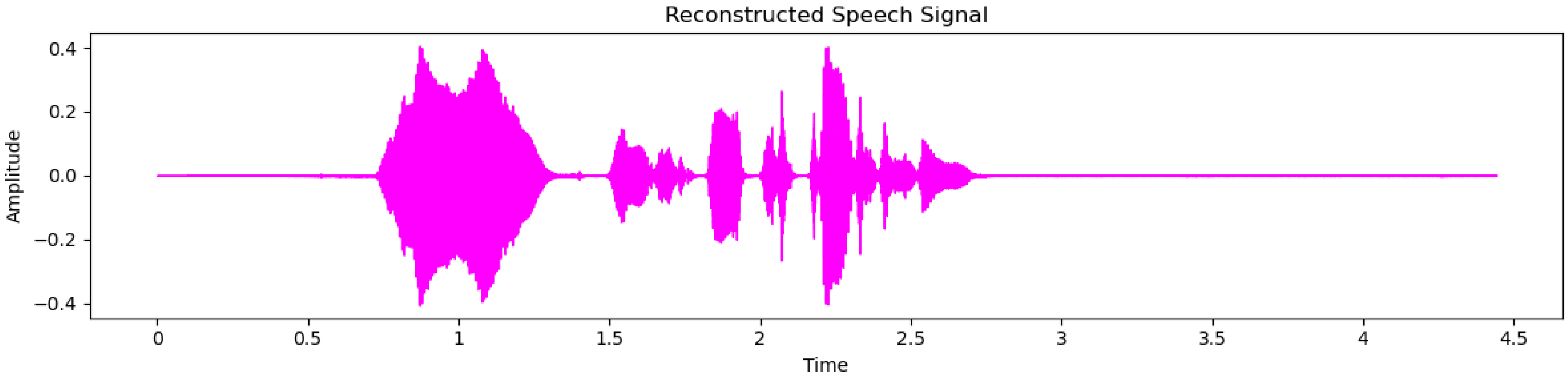
```python
# Use numpy.fft.ifft() to transform the speech signal from frequency domain to i
ifft_result = np.fft.ifft(fft_result)
print("after reconstruction")
ipd.display(ipd.Audio(ifft_result, rate=sr))
```

```
after reconstruction
```

0:04 / 0:04

In [19]:
```
# Plot the original and reconstructed signals for comparison
plt.figure(figsize=(12, 6))

# Plot the original signal
plt.subplot(2, 1, 1)
librosa.display.waveshow(y, sr=sr, color='green')
plt.title('Original Speech Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

# Plot the reconstructed signal
plt.subplot(2, 1, 2)
librosa.display.waveshow(np.real(ifft_result), sr=sr, color='magenta')  # Use np
plt.title('Reconstructed Speech Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()
```

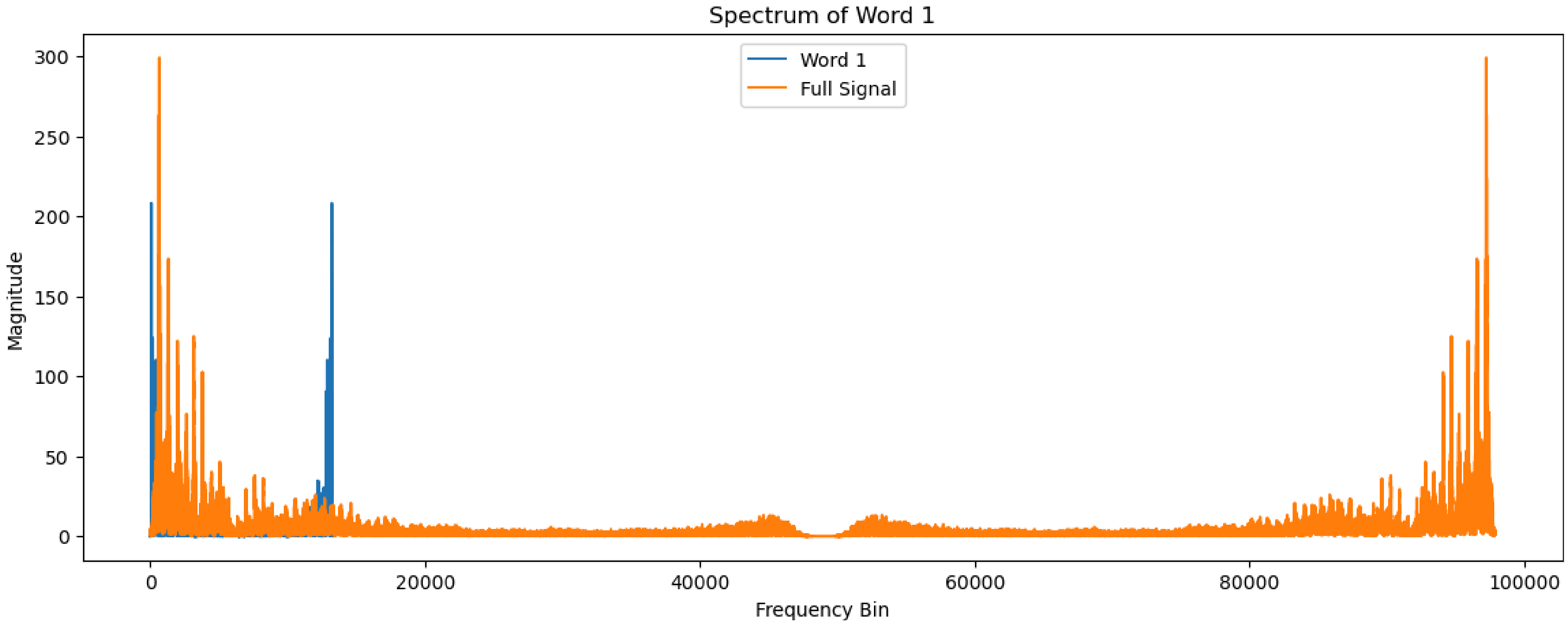

Original Speech Signal
Amplitude
0.4
0.2
0.0
−0.2
−0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Time

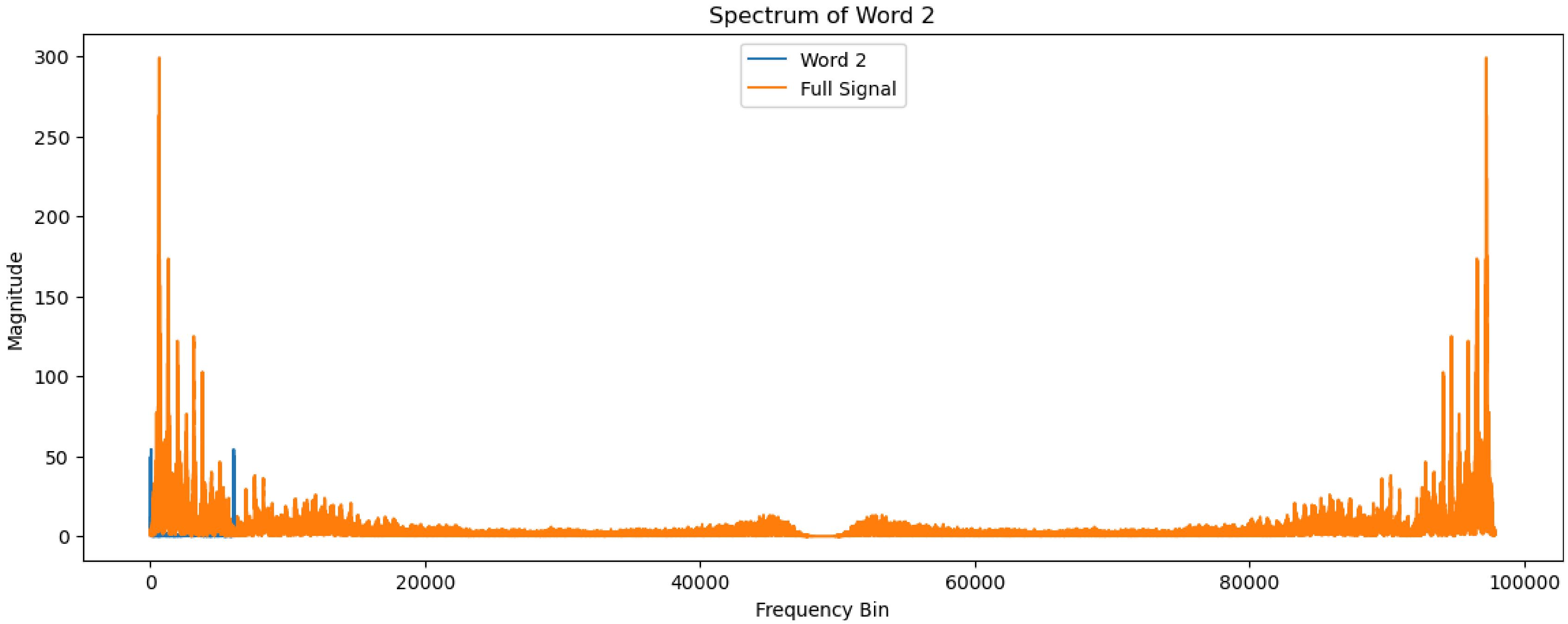

Reconstructed Speech Signal
Amplitude
0.4
0.2
0.0
−0.2
−0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Time

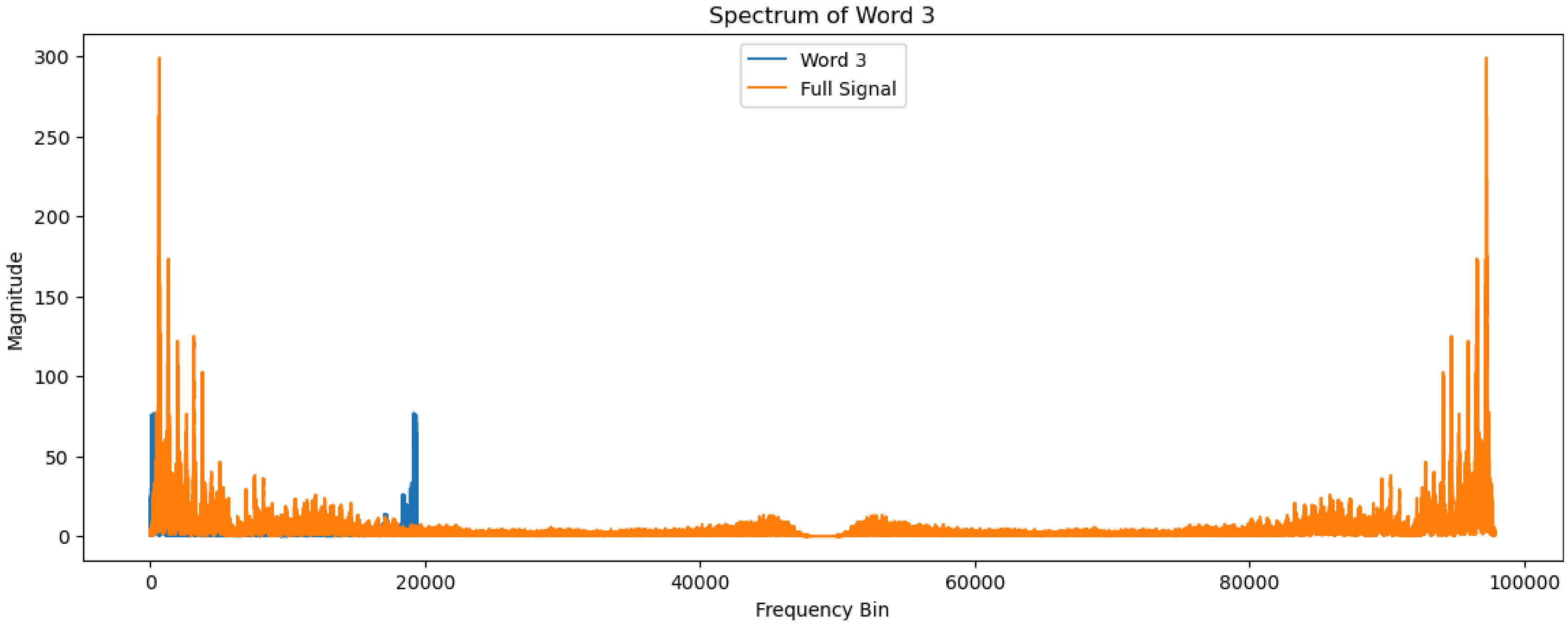
A3. Perform the spectral analysis of a word present in the recorded speech. Compare the spectrum with the spectrum of the full signal.

In [48]:

```
threshold = np.percentile(np.abs(y), 92)
segments = librosa.effects.split(y, top_db=-15 * np.log10(threshold))
for i, (start, end) in enumerate(segments):
    word = y[start:end]
    D_full = np.fft.fft(y)
    D_word = np.fft.fft(word)
    plt.figure(figsize=(14, 5))
    plt.plot(np.abs(D_word), label=f'Word {i+1}')
    plt.plot(np.abs(D_full), label='Full Signal')

    plt.title(f'Spectrum of Word {i+1}')
    plt.xlabel('Frequency Bin')
    plt.ylabel('Magnitude')
    plt.legend()
    plt.show()
```


Spectrum of Word 1
Word 1
Full Signal
Magnitude
0
50
100
150
200
250
300
0
20000
40000
60000
80000
100000
Frequency Bin


Spectrum of Word 2
Word 2
Full Signal
Magnitude
0
50
100
150
200
250
300
0
20000
40000
60000
80000
100000
Frequency Bin


Spectrum of Word 3
Word 3
Full Signal
Magnitude
0
50
100
150
200
250
300
0
20000
40000
60000
80000
100000
Frequency Bin

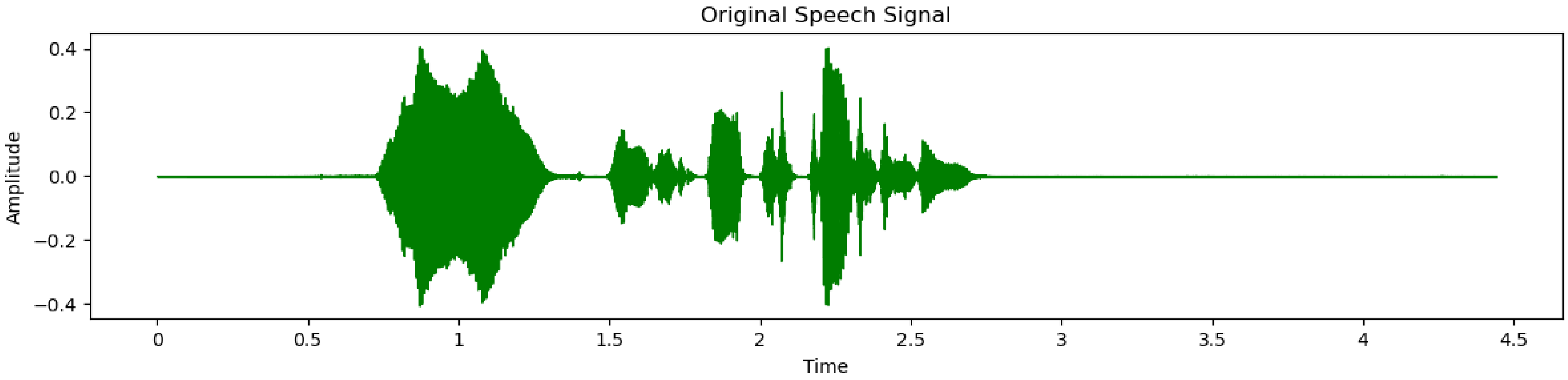
A4. Take a rectangular window of 20 mili-second sampled at 22.5 KHz. Using FFT, analyse the spectral components.

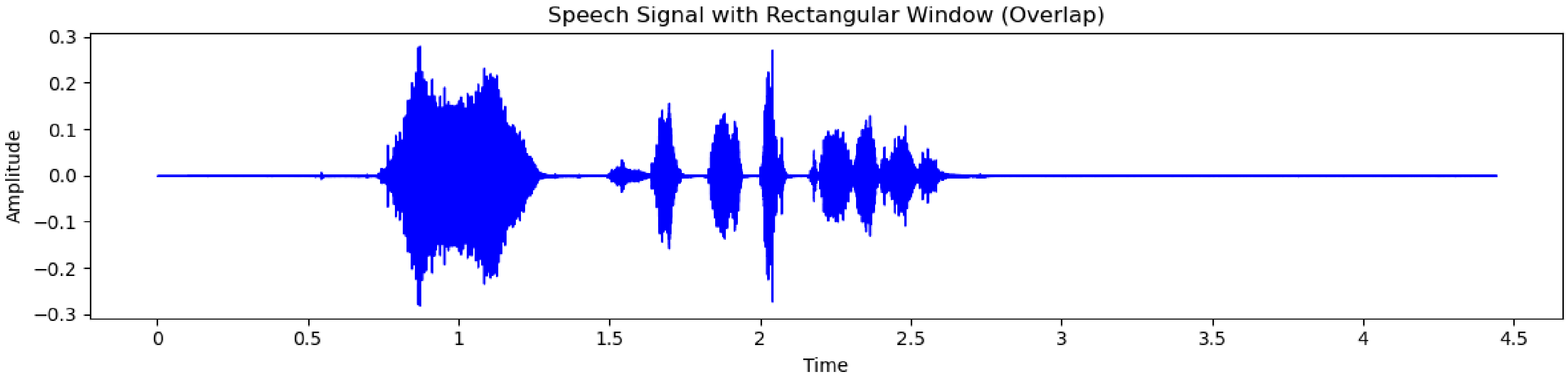
In [53]:

```
# Define the parameters for the rectangular window
window_size = int(0.02 * sr)  # 20 milliseconds window size
overlap = int(0.01 * sr)  # 10 milliseconds overlap

# Apply the window to the signal with overlap
y_windowed = librosa.effects.preemphasis(y)
y_frames = librosa.util.frame(y_windowed, frame_length=window_size, hop_length=o
```

In [57]:

```

# Display the original and windowed signals
plt.figure(figsize=(12, 6))

# Plot the original signal
plt.subplot(2, 1, 1)
librosa.display.waveshow(y, sr=sr, color='green')
plt.title('Original Speech Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')

# Plot the windowed signal (considering overlap)
plt.subplot(2, 1, 2)
librosa.display.waveshow(y_windowed, sr=sr, color='blue')
plt.title('Speech Signal with Rectangular Window (Overlap)')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()

print("rectangular window")
ipd.display(ipd.Audio(y_windowed, rate=sr))
```



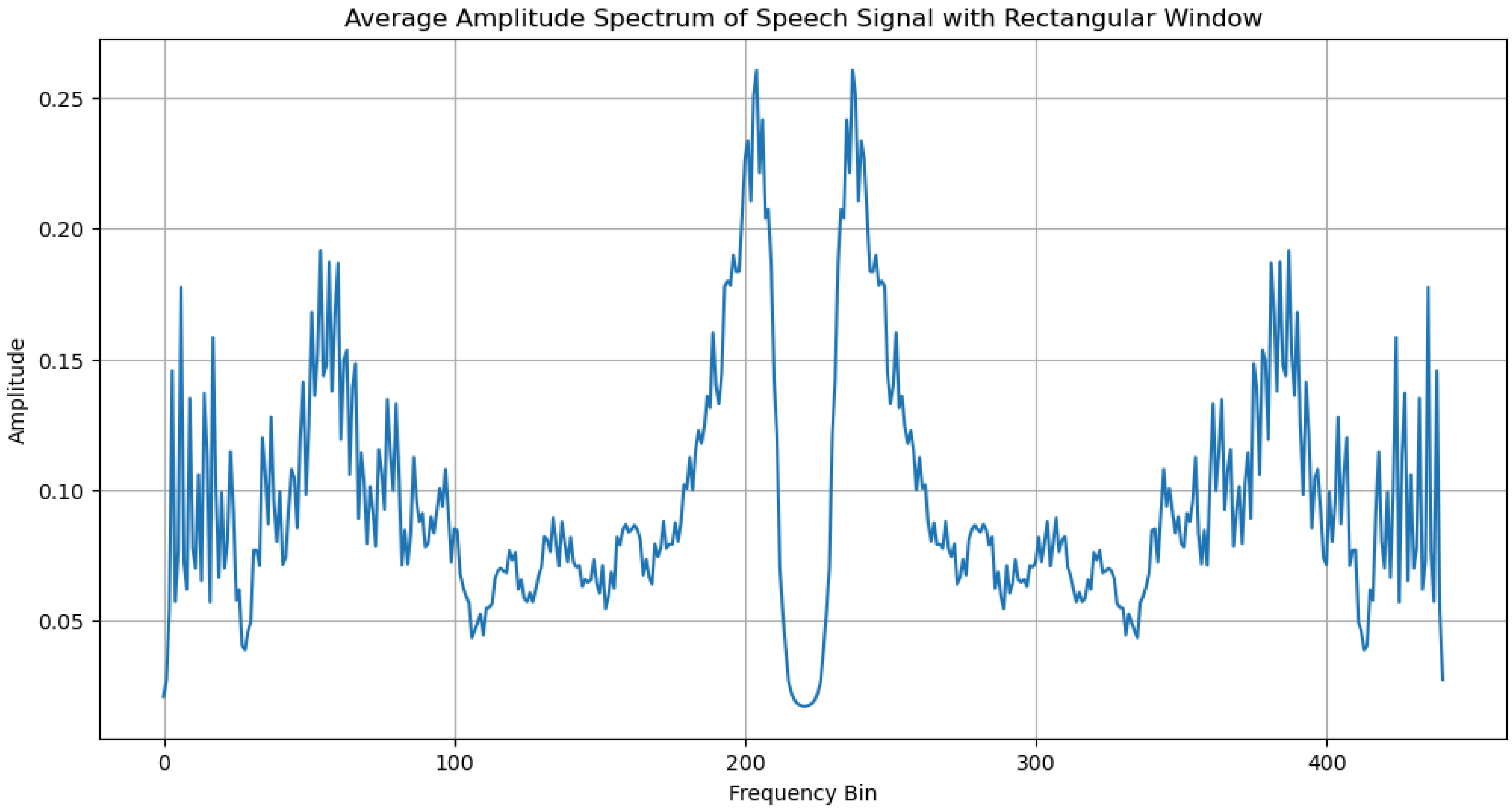



```
rectangular window
```

0:04 / 0:04

In [61]:

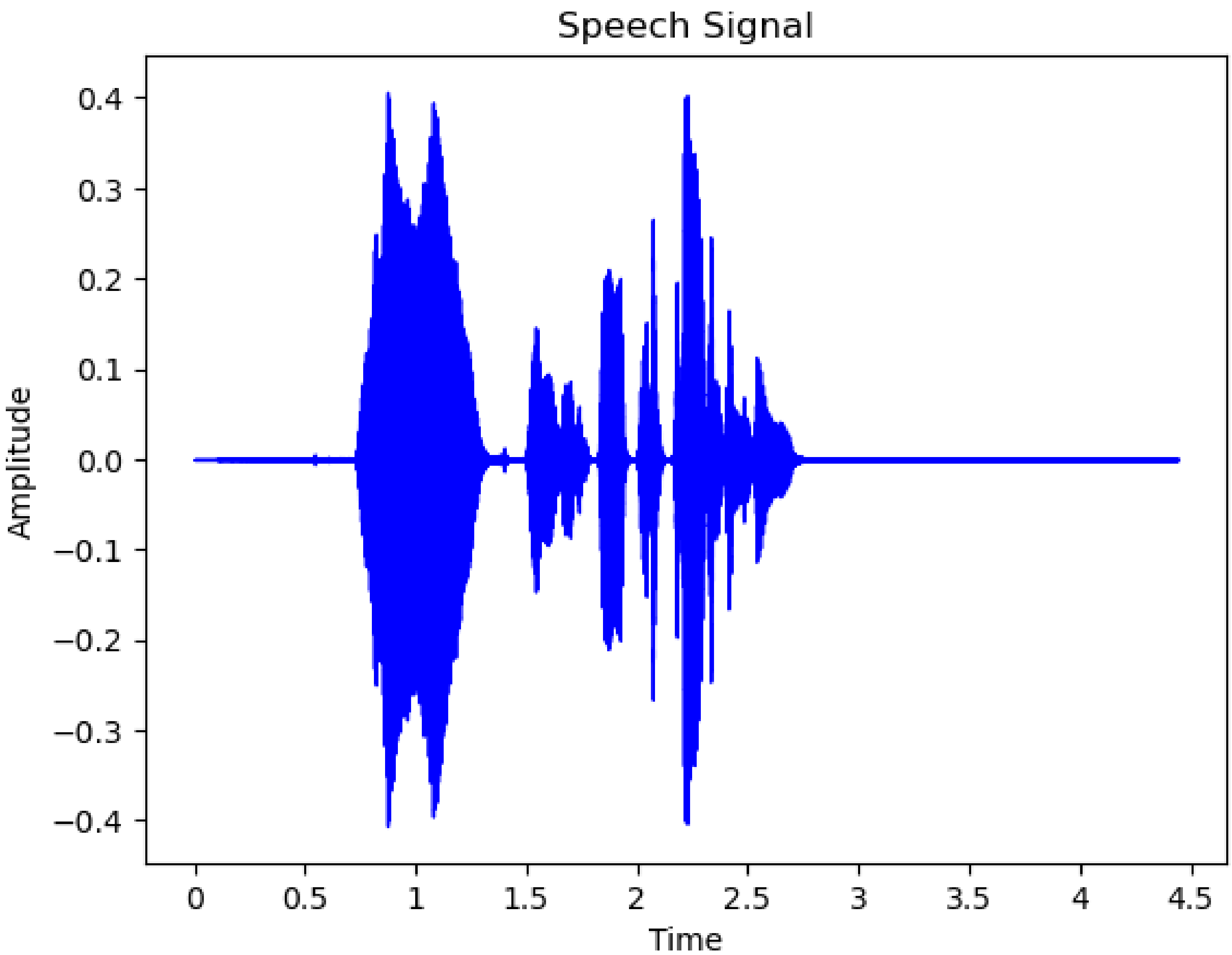
```python

# Apply FFT to each windowed segment
fft_results_windowed = np.fft.fft(y_frames, axis=0)

# Calculate the amplitude spectrum of the windowed signal
amplitude_spectrum_windowed = np.abs(fft_results_windowed)

# Display the amplitude spectrum
plt.figure(figsize=(12, 6))
plt.plot(np.mean(amplitude_spectrum_windowed, axis=1))  # Plot the average spectr
plt.title('Average Amplitude Spectrum of Speech Signal with Rectangular Window')
plt.xlabel('Frequency Bin')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
```

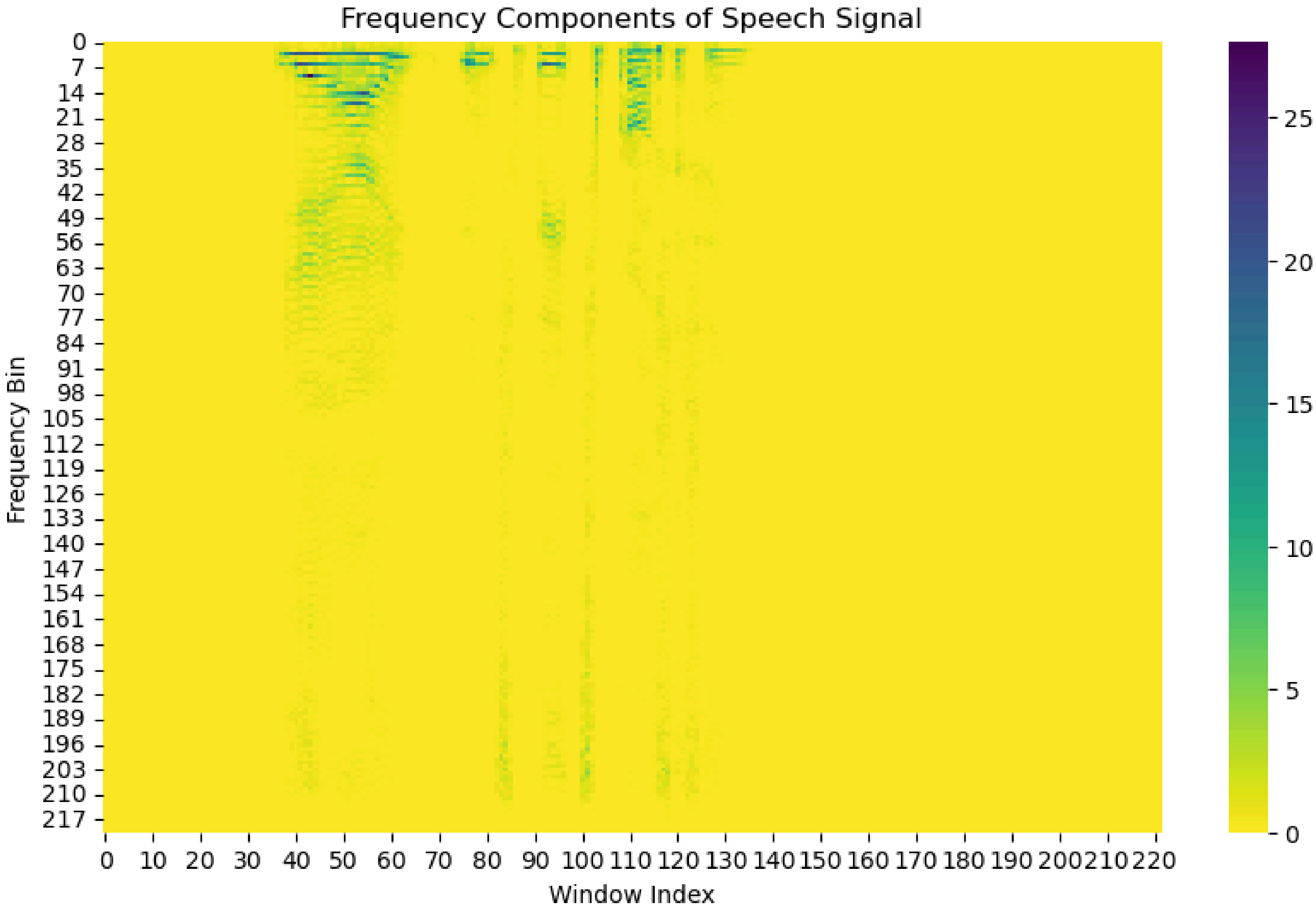

Average Amplitude Spectrum of Speech Signal with Rectangular Window
Amplitude
Frequency Bin
0.25
0.20
0.15
0.10
0.05
0
100
200
300
400

A5. Break your speech signal into window lengths of 20 mSec intervals. Evaluate the frequency components using numpy.fft.rfft(). Stack these frequency components as columns in a matrix. Use heatmap plot to display the matrix. You may use librosa.stft() or scipy.signal.stft() as well to achieve this.

In [66]:
```
frequencies, times, spectrogram = signal.stft(y, fs=sr, nperseg=window_size, nove
```

In [77]:
```
librosa.display.waveshow(y, sr=sr, color = 'blue')
plt.title('Speech Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.show()
window_length_sec = 0.02
window_length = int(window_length_sec * sr)
num_windows = len(y) // window_length
freq_matrix = np.zeros((num_windows, window_length // 2 + 1))
for i in range(num_windows):
    window = y[i * window_length: (i + 1) * window_length]
    fft_result = np.fft.rfft(window)
    freq_matrix[i, :] = np.abs(fft_result)

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(freq_matrix.T, cmap='viridis_r', xticklabels=10)
plt.title('Frequency Components of Speech Signal')
plt.xlabel('Window Index')
plt.ylabel('Frequency Bin')
plt.show()
```


Speech Signal
Amplitude
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Time

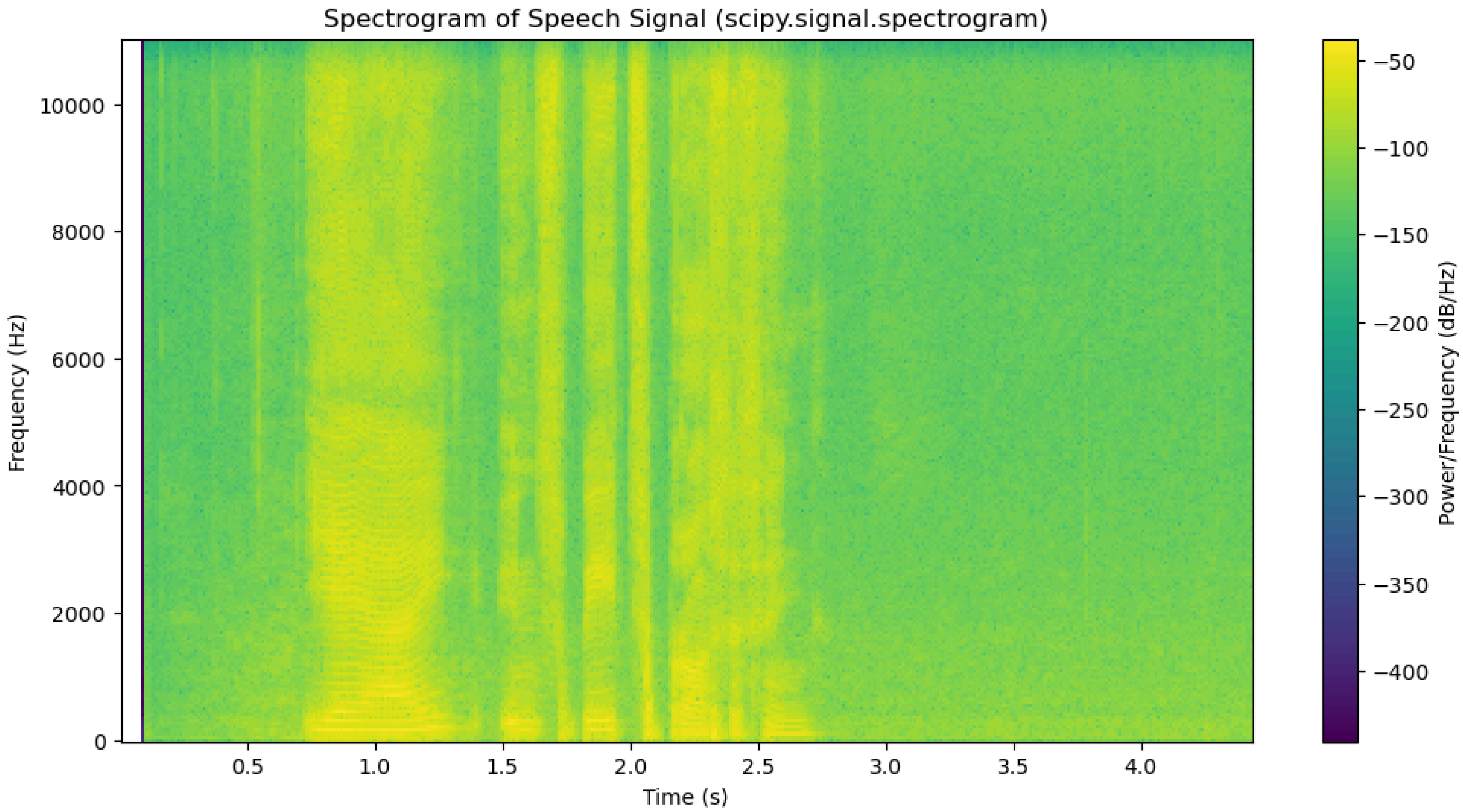

Frequency Components of Speech Signal
Frequency Bin
0
7
14
21
28
35
42
49
56
63
70
77
84
91
98
105
112
119
126
133
140
147
154
161
168
175
182
189
196
203
210
217
Window Index
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220
0
5
10
15
20
25

A6. Use scipy.signal.spectrogram() to plot the spectrogram of the speech signal at the same duration. Compare the plots.

In [83]:

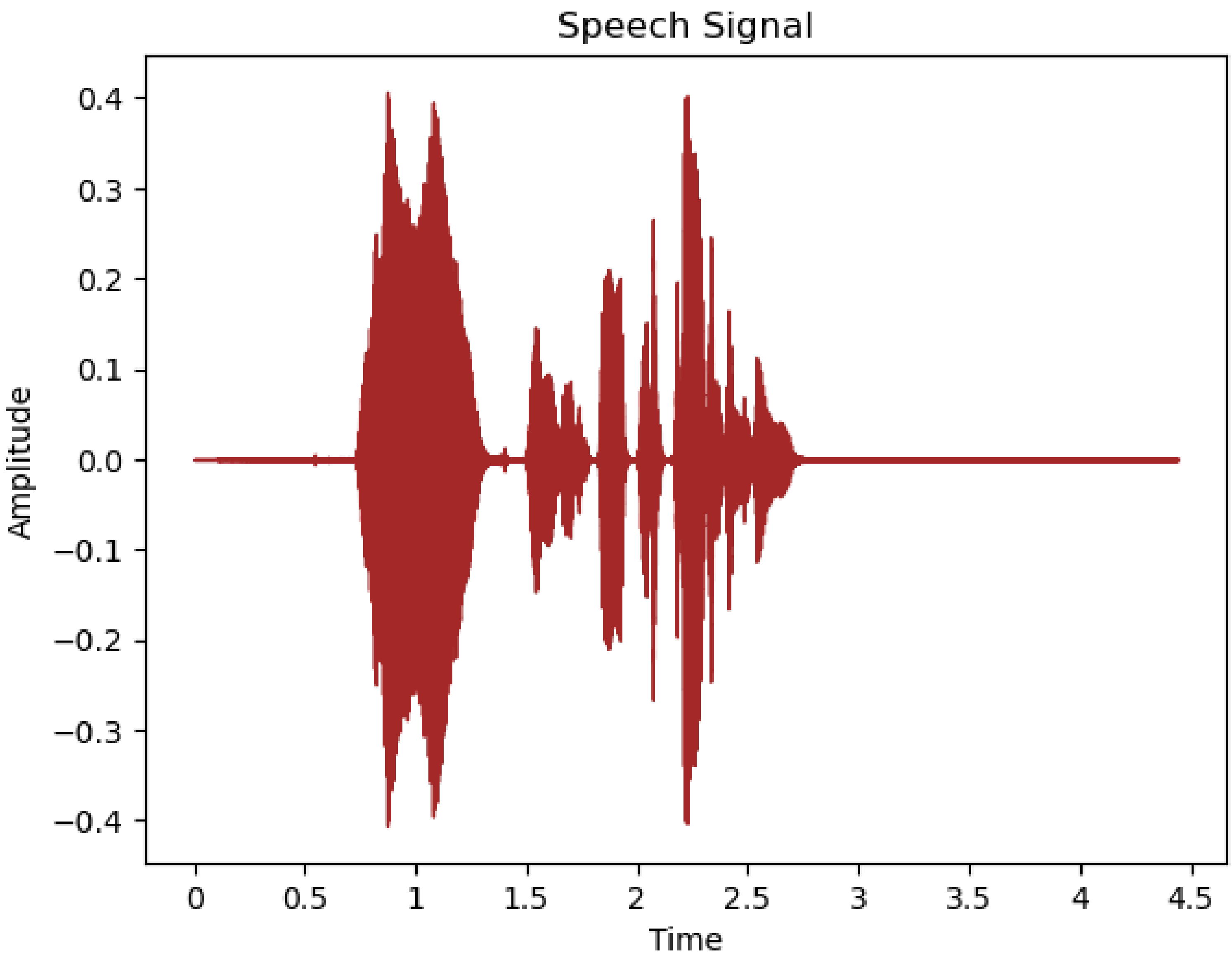
```
frequencies, times, Sxx = spectrogram(y, fs=sr, nperseg=window_size, noverlap=ove

# Display the spectrogram using matplotlib
plt.figure(figsize=(12, 6))
plt.pcolormesh(times, frequencies, 10 * np.log10(Sxx), shading='auto', cmap='vir

plt.title('Spectrogram of Speech Signal (scipy.signal.spectrogram)')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.colorbar(label='Power/Frequency (dB/Hz)')
plt.show()
```

```
C:\Users\saide\AppData\Local\Temp\ipykernel_28520\3548733323.py:5: RuntimeWarning: d
ivide by zero encountered in log10
  plt.pcolormesh(times, frequencies, 10 * np.log10(Sxx), shading='auto', cmap='virid
is')
```

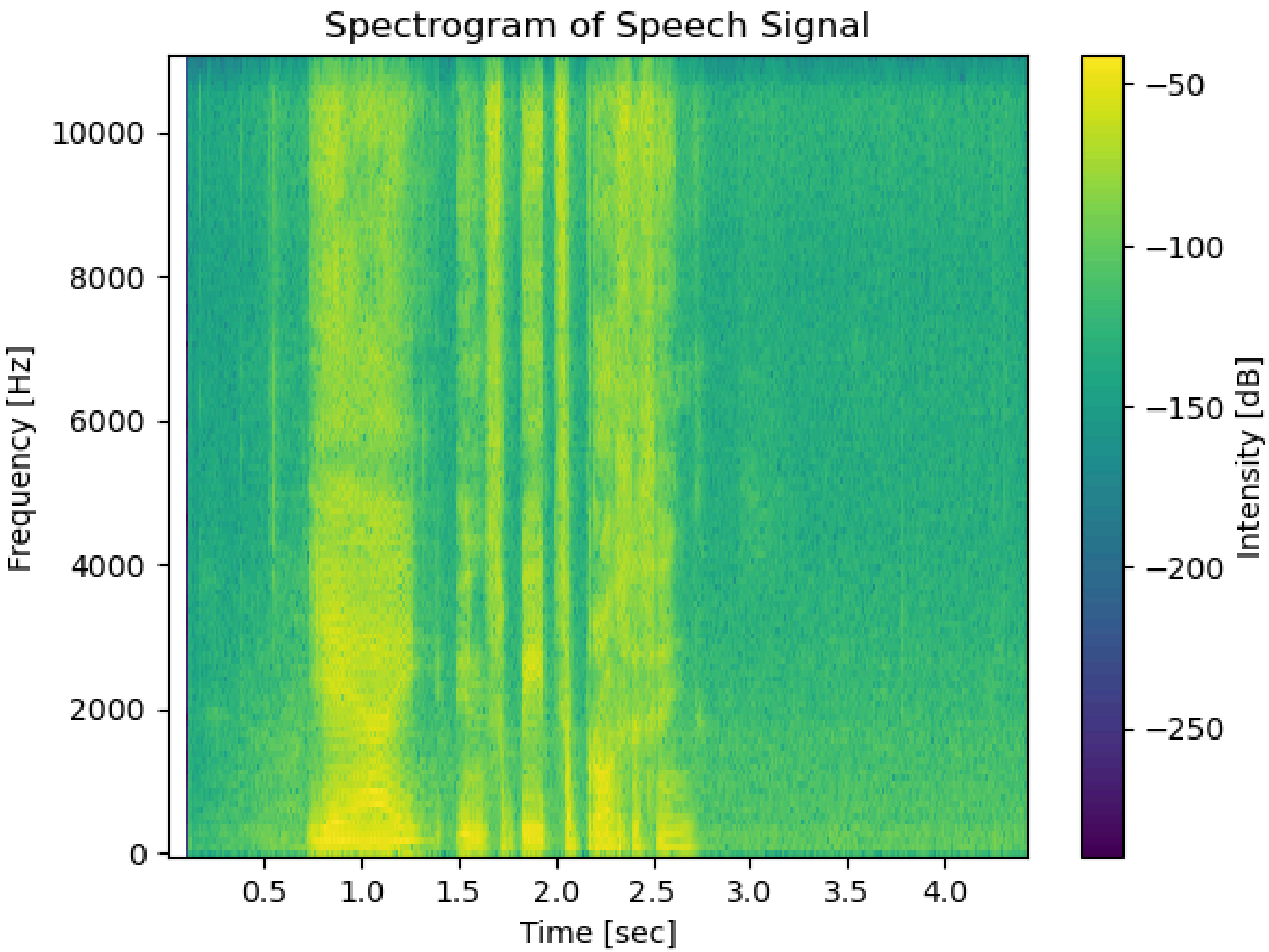

Spectrogram of Speech Signal (scipy.signal.spectrogram)
Frequency (Hz)
0
2000
4000
6000
8000
10000
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Time (s)
−50
−100
−150
−200
−250
−300
−350
−400
Power/Frequency (dB/Hz)

In [80]:

```
from scipy.signal import spectrogram

librosa.display.waveshow(y, color = 'brown')
plt.title('Speech Signal')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.show()

f, t, Sxx = spectrogram(y, sr)
plt.pcolormesh(t, f, 10 * np.log10(Sxx))
plt.ylabel('Frequency [Hz]')
plt.xlabel('Time [sec]')
plt.title('Spectrogram of Speech Signal')
plt.colorbar(label='Intensity [dB]')
plt.show()
```




```
C:\Users\saide\AppData\Local\Temp\ipykernel_28520\590411667.py:10: RuntimeWarning: d
ivide by zero encountered in log10
  plt.pcolormesh(t, f, 10 * np.log10(Sxx))
```


Spectrogram of Speech Signal
Frequency [Hz]
0
2000
4000
6000
8000
10000
Time [sec]
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Intensity [dB]
−50
−100
−150
−200
−250

In [ ]:

```
1
```