BMI 500 Neural Time-Series Analysis Lab module

Alireza Rafiei - Fall 2022 - HW10

- **Part A:**

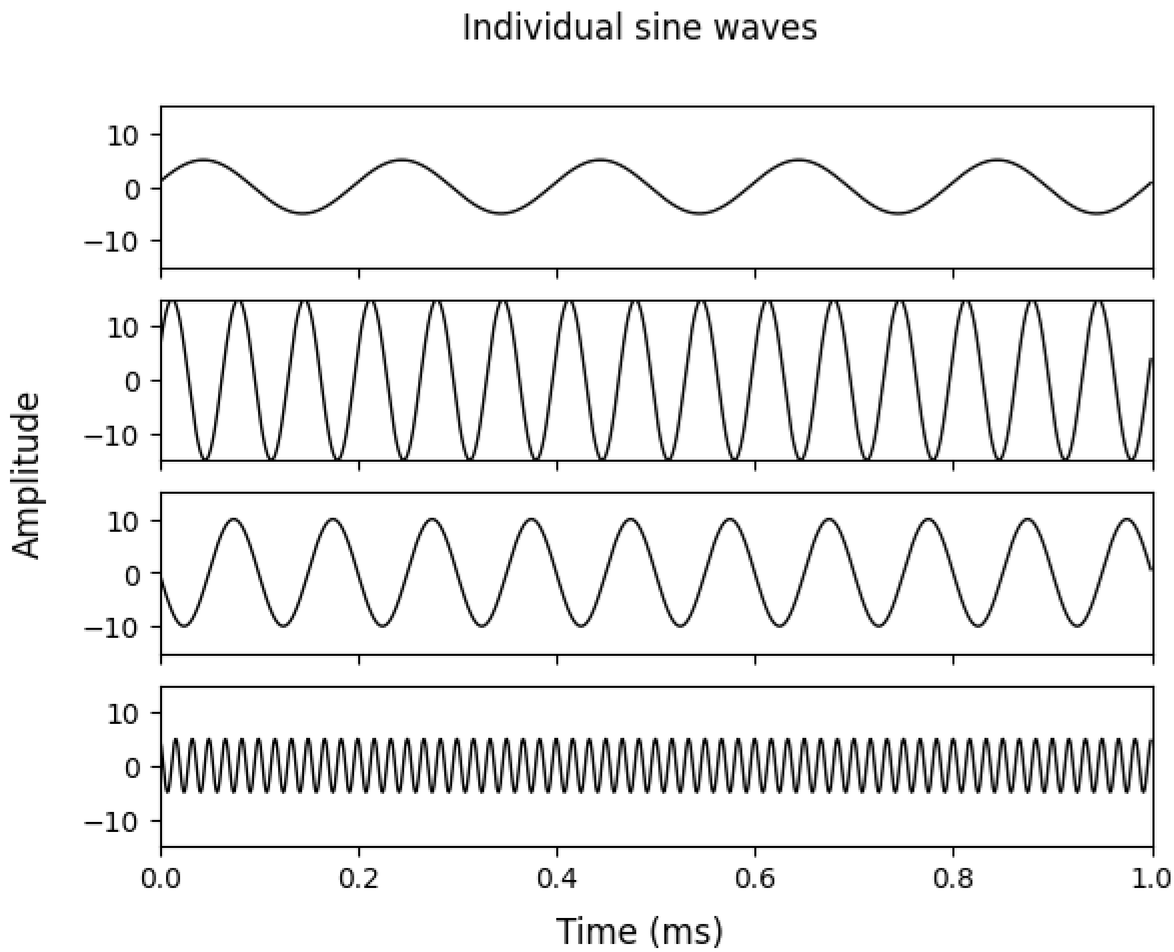
In [1]:
```
# Import required libraries

import numpy as np
import matplotlib.pyplot as plt
```

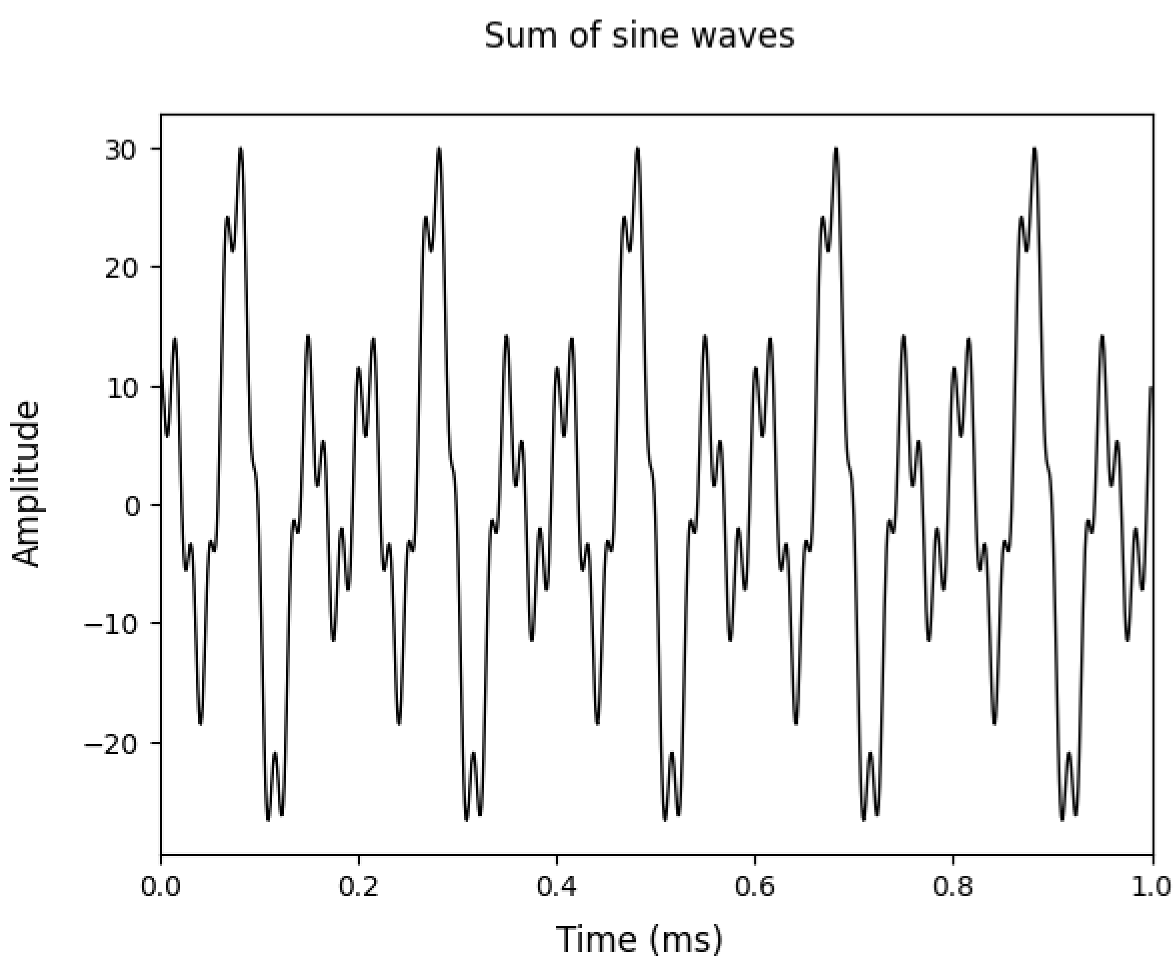
In [2]:
```
# Define four sine waves

fs = 1000;                                          # Sampling frequency
ts = 1/fs;                                          # Period
t = np.arange(0, 1, ts)
y1 = 5*np.sin(10*np.pi*t + np.pi/18)                # First sine wave
y2 = 15*np.sin(30*np.pi*t + np.pi/9)                # Second sine wave
y3 = 10*np.sin(20*np.pi*t + np.pi)                  # Third sine wave
y4 = 5*np.sin(120*np.pi*t + np.pi/2)                # Forth sine wave

y_all = np.array([y1,y2,y3,y4])
```

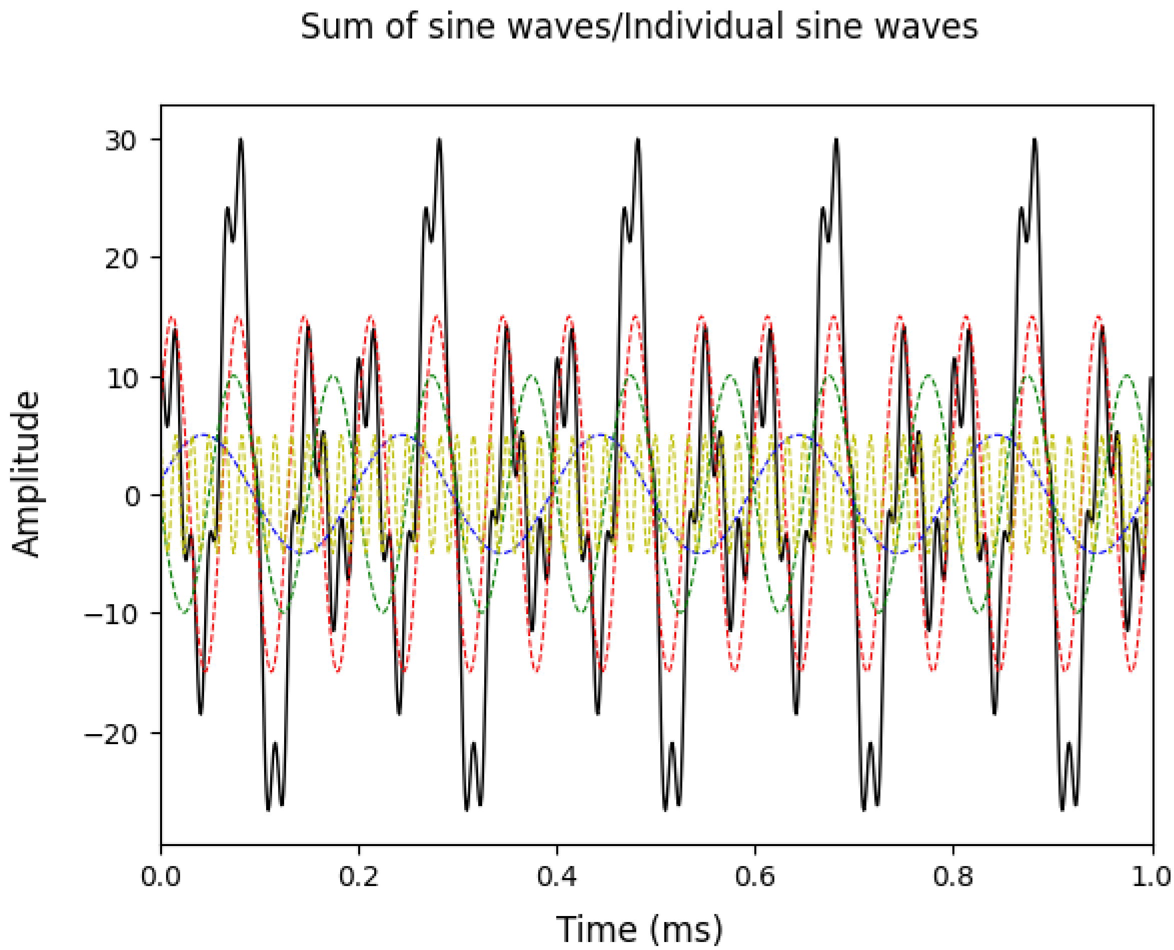
In [3]:
```
# Sum of sine waves
y = y1 + y2 + y3 + y4
```

In [4]:
```
# Visualization
plt.figure(1)
fig, ax = plt.subplots(4, 1, sharex='col', sharey='row')
fig.suptitle('Individual sine waves')
fig.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
for i in range(4):
    ax[i].plot(t,y_all[i], 'k',  linewidth = 1)
    ax[i].set_xlim(0, 1)
    ax[i].set_ylim(-15, 15)

fig2, ax2 = plt.subplots(1, 1)
ax2.plot(t,y,'k',  linewidth = 1)
fig2.suptitle('Sum of sine waves')
fig2.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig2.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
ax2.set_xlim(0, 1)

fig3, ax3 = plt.subplots(1, 1)
ax3.plot(t,y,'k',  linewidth = 1)
ax3.plot(t,y_all[0],'--b',  linewidth = 0.75)
ax3.plot(t,y_all[1],'--r',  linewidth = 0.75)
ax3.plot(t,y_all[2],'--g',  linewidth = 0.75)
ax3.plot(t,y_all[3],'--y',  linewidth = 0.75)
fig3.suptitle('Sum of sine waves/Individual sine waves')
fig3.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig3.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
ax3.set_xlim(0, 1)
```

```
Out[4]: (0.0, 1.0)

<Figure size 640x480 with 0 Axes>
```

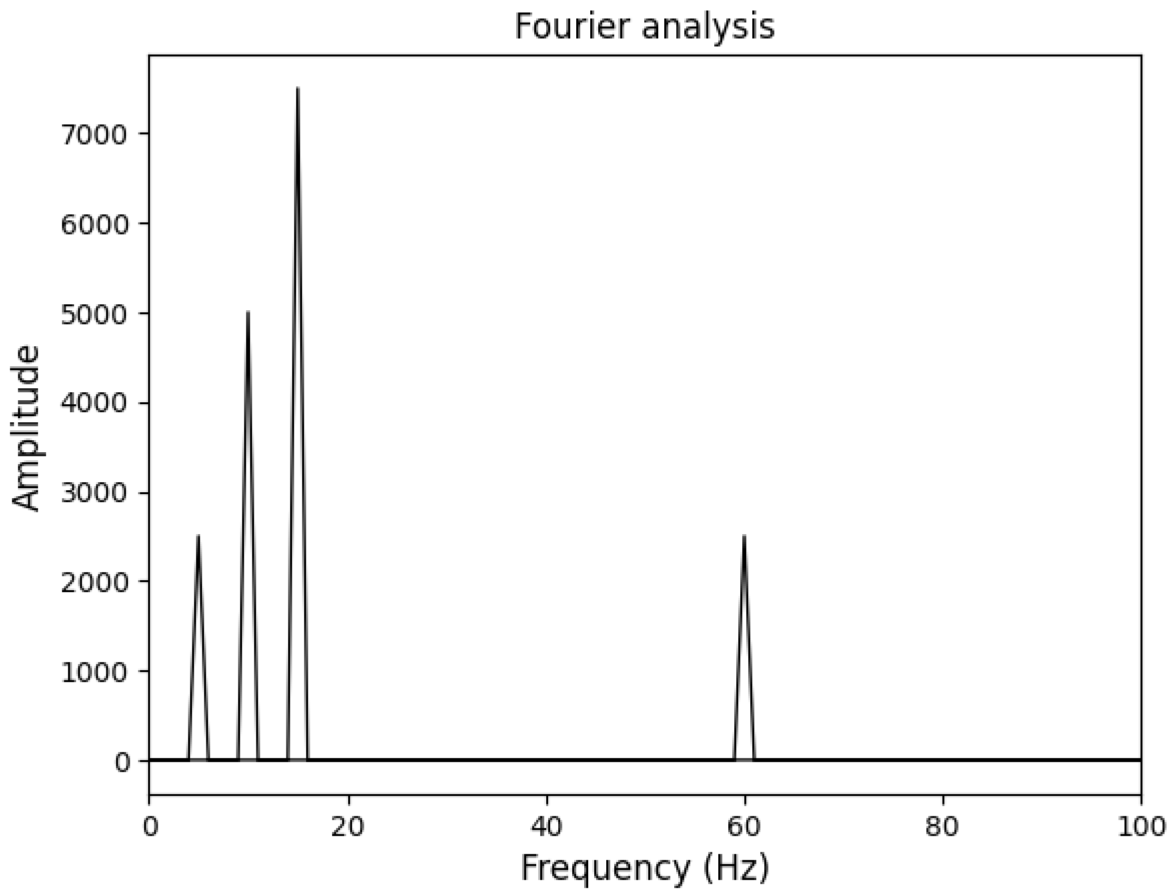






In [5]:

```
# Fourier analysis
from scipy.fft import fft, fftfreq

# Number of samples
N = fs * 1

yf = fft(y)
xf = fftfreq(N, ts)

plt.plot(xf, np.abs(yf), 'k', linewidth = 1)
plt.title('Fourier analysis')
plt.xlabel('Frequency (Hz)', fontsize = 12)
plt.ylabel('Amplitude', fontsize = 12)
plt.xlim(0,100)
plt.show()
```



First, four sine waves were generated with the frequencies of 5, 10, 15, 60 HZ. In Fourier analysis, the same frequencies can be observed, confirming that the code is correct.

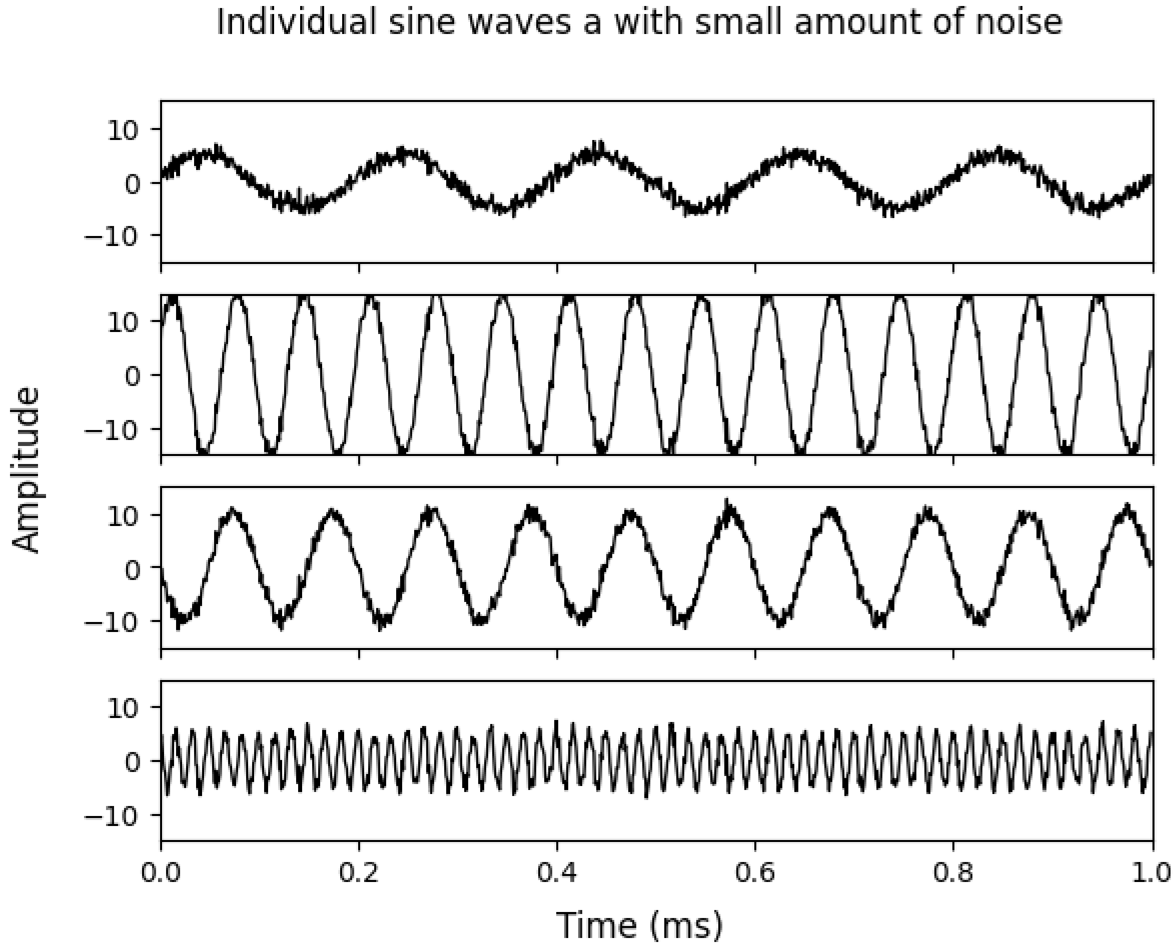
- **Part B:**

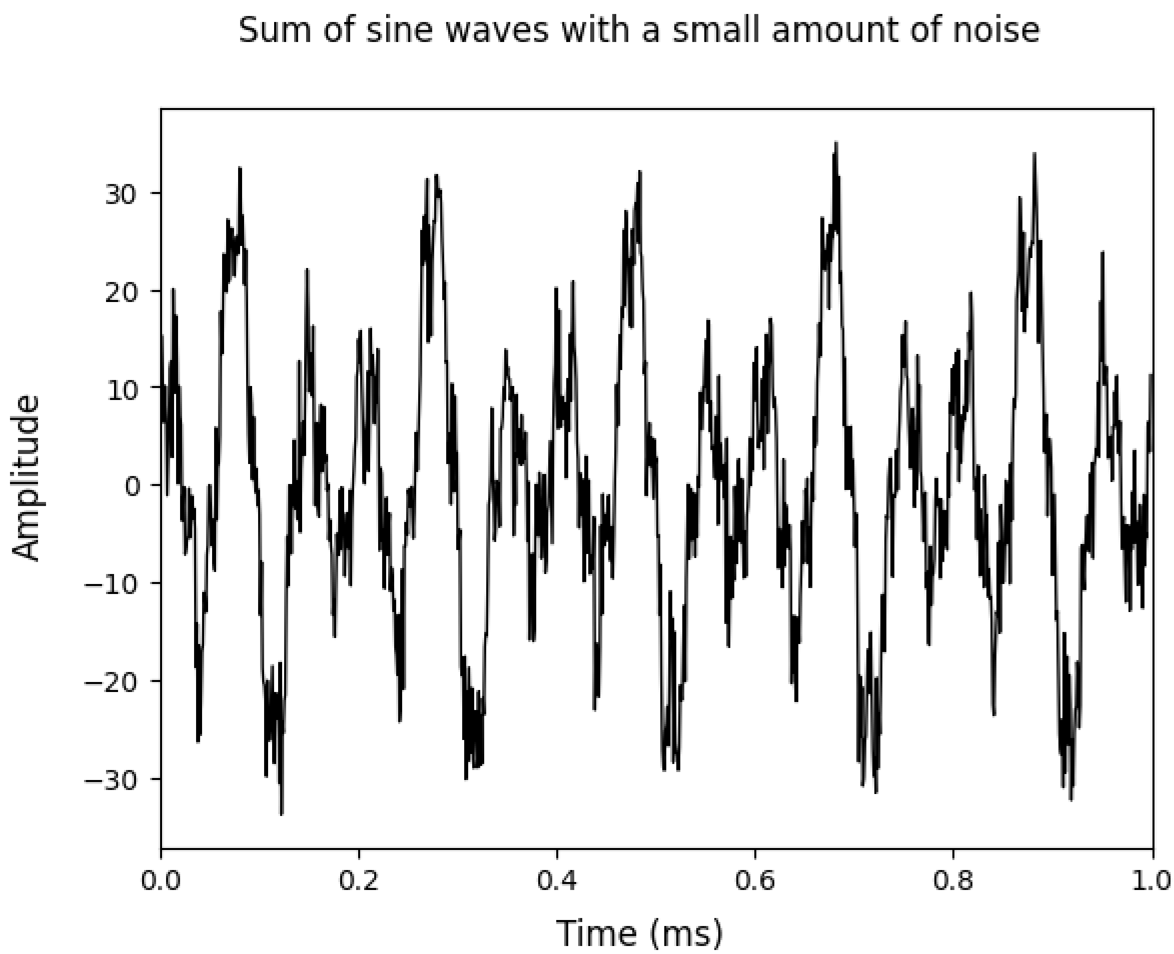
In [6]:
```
# Add a small and large amount of random noise

fs = 1000;
ts = 1/fs;
t = np.arange(0, 1, ts)
Small_noise = np.random.normal(0,1,len(t))                # Generate a small amount of random
Large_noise = np.random.normal(0,30,len(t))               # Generate a large amount of random
y1_sn = 5*np.sin(10*np.pi*t + np.pi/18) + Small_noise
y2_sn = 15*np.sin(30*np.pi*t + np.pi/9) + Small_noise
y3_sn = 10*np.sin(20*np.pi*t + np.pi) + Small_noise
y4_sn = 5*np.sin(120*np.pi*t + np.pi/2) + Small_noise

y1_ln = 5*np.sin(10*np.pi*t + np.pi/18) + Large_noise
y2_ln = 15*np.sin(30*np.pi*t + np.pi/9) + Large_noise
y3_ln = 10*np.sin(20*np.pi*t + np.pi) + Large_noise
y4_ln = 5*np.sin(120*np.pi*t + np.pi/2) + Large_noise

y_all_sn = np.array([y1_sn,y2_sn,y3_sn,y4_sn])
y_all_ln = np.array([y1_ln,y2_ln,y3_ln,y4_ln])

y_sn = y1_sn+y2_sn+y3_sn+y4_sn                            # Sum of sine waves with a small amo
y_ln = y1_ln+y2_ln+y3_ln+y4_ln                            # Sum of sine waves with a large amo
```

In [7]:
```
# Visualization
plt.figure(1)
fig, ax = plt.subplots(4, 1, sharex='col', sharey='row')
```

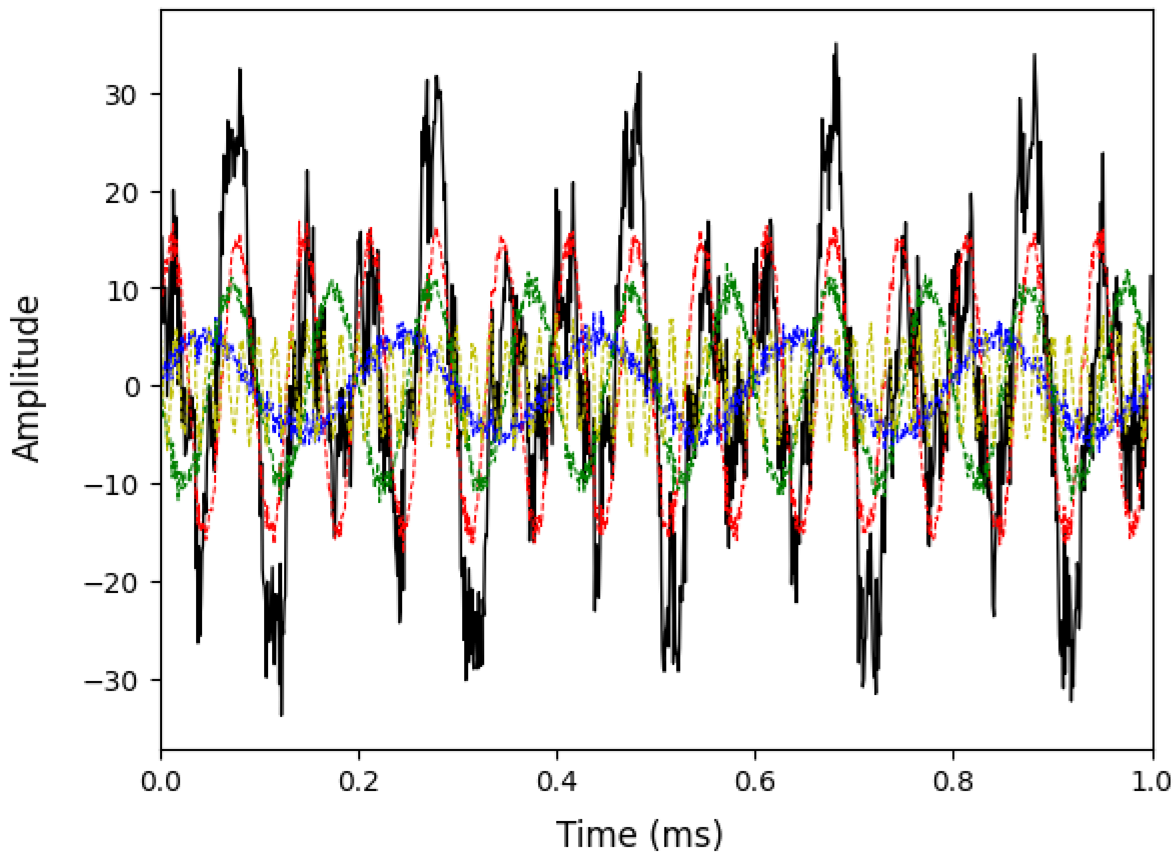
```python
fig.suptitle('Individual sine waves a with small amount of noise')
fig.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
for i in range(4):
    ax[i].plot(t,y_all_sn[i], 'k',  linewidth = 1)
    ax[i].set_xlim(0, 1)
    ax[i].set_ylim(-15, 15)



fig2, ax2 = plt.subplots(1, 1)
ax2.plot(t,y_sn,'k',  linewidth = 1)
fig2.suptitle('Sum of sine waves with a small amount of noise')
fig2.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig2.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
ax2.set_xlim(0, 1)

fig3, ax3 = plt.subplots(1, 1)
ax3.plot(t,y_sn,'k',  linewidth = 1)
ax3.plot(t,y_all_sn[0],'--b',  linewidth = 0.75)
ax3.plot(t,y_all_sn[1],'--r',  linewidth = 0.75)
ax3.plot(t,y_all_sn[2],'--g',  linewidth = 0.75)
ax3.plot(t,y_all_sn[3],'--y',  linewidth = 0.75)
fig3.suptitle('Sum of sine waves/Individual sine waves with a small amount of noise')
fig3.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig3.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
ax3.set_xlim(0, 1)

plt.figure(4)
fig, ax = plt.subplots(4, 1, sharex='col', sharey='row')
fig.suptitle('Individual sine waves with a large amount of noise')
fig.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
for i in range(4):
    ax[i].plot(t,y_all_ln[i], 'k',  linewidth = 1)
    ax[i].set_xlim(0, 1)
    #ax[i].set_ylim(-15, 15)



fig5, ax5 = plt.subplots(1, 1)
ax5.plot(t,y_ln,'k',  linewidth = 1)
fig5.suptitle('Sum of sine waves with a large amount of noise')
fig5.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig5.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
ax5.set_xlim(0, 1)

fig6, ax6 = plt.subplots(1, 1)
ax6.plot(t,y_sn,'k',  linewidth = 1)
ax6.plot(t,y_all_ln[0],'--b',  linewidth = 0.75)
ax6.plot(t,y_all_ln[1],'--r',  linewidth = 0.75)
ax6.plot(t,y_all_ln[2],'--g',  linewidth = 0.75)
ax6.plot(t,y_all_ln[3],'--y',  linewidth = 0.75)
fig6.suptitle('Sum of sine waves/Individual sine waves with a large amount of noise')
fig6.text(0.5, 0.01,'Time (ms)', ha='center', fontsize = 12)
fig6.text(0.01, 0.5,'Amplitude', va='center', rotation='vertical', fontsize = 12)
ax6.set_xlim(0, 1)
```

```
Out[7]: (0.0, 1.0)

<Figure size 640x480 with 0 Axes>
```

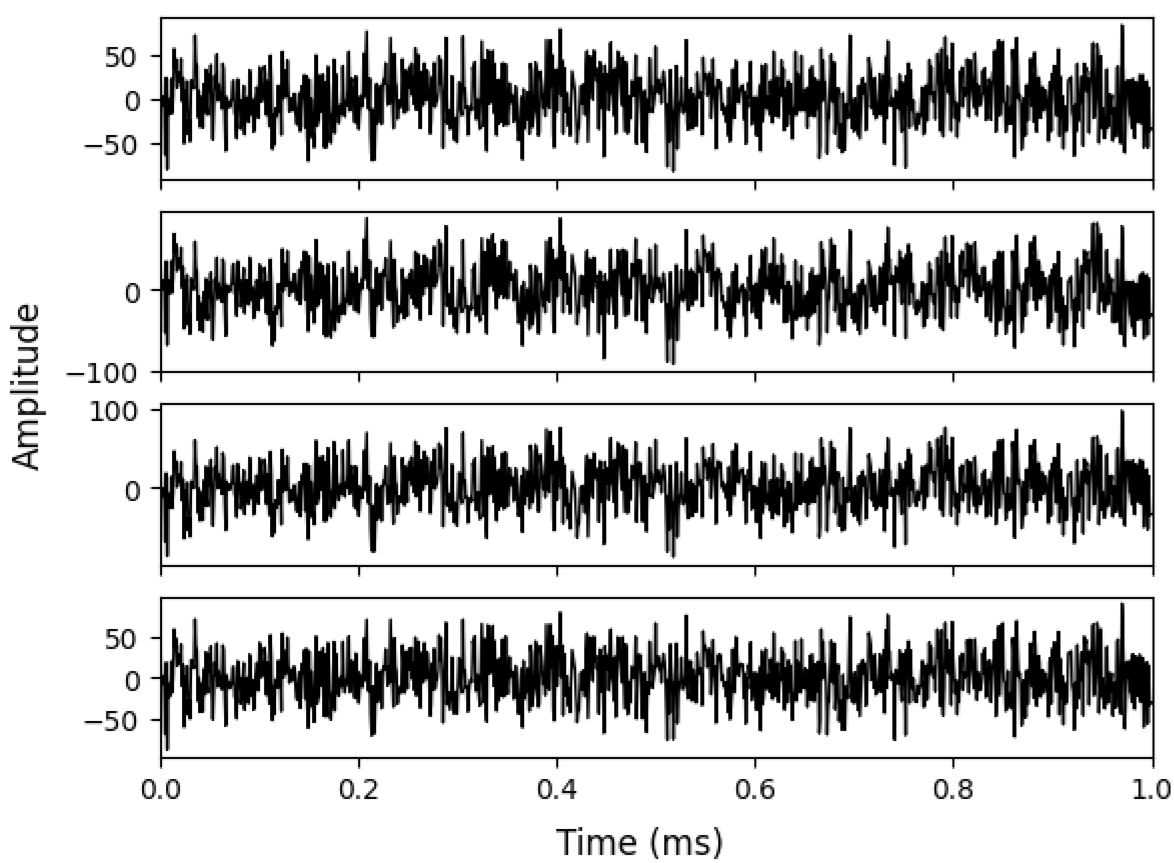

Individual sine waves a with small amount of noise
Amplitude
10
0
−10
0.0
0.2
0.4
0.6
0.8
1.0
Time (ms)


Sum of sine waves with a small amount of noise
Amplitude
30
20
10
0
−10
−20
−30
0.0
0.2
0.4
0.6
0.8
1.0
Time (ms)

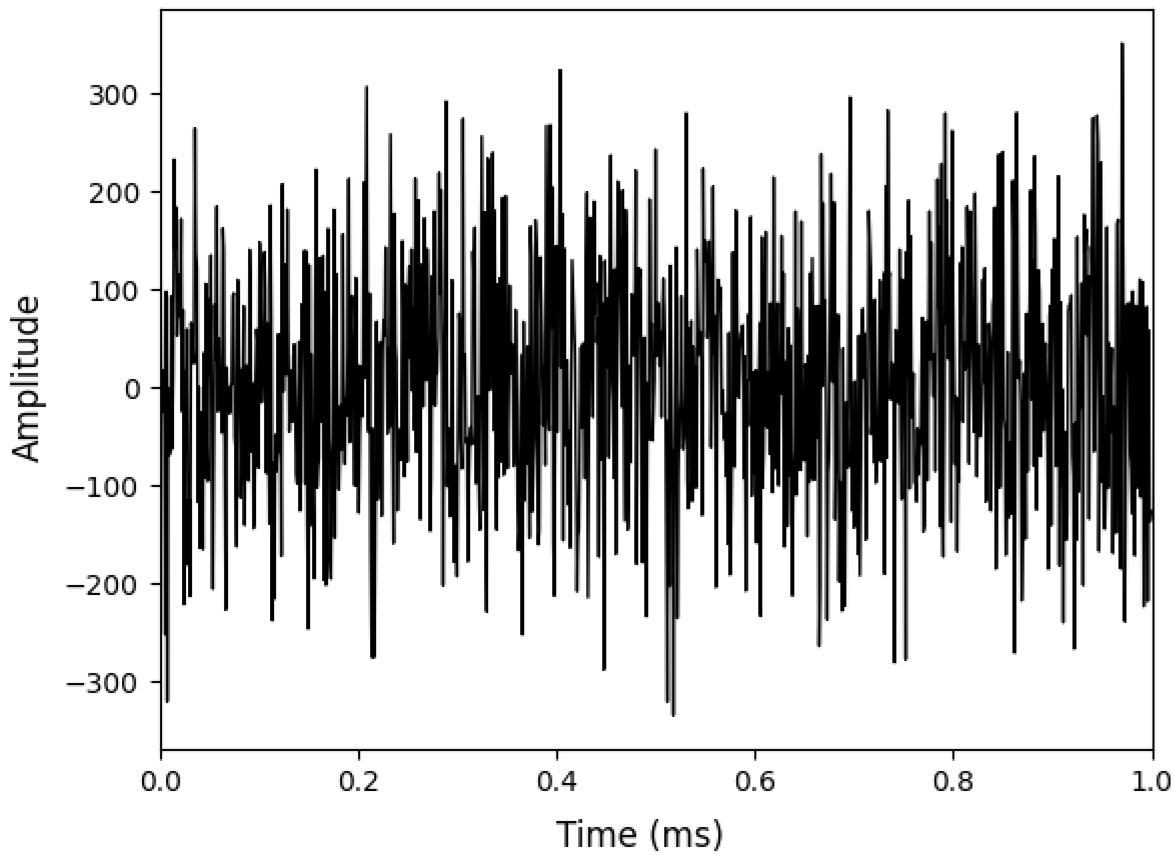
Sum of sine waves/Individual sine waves with a small amount of noise



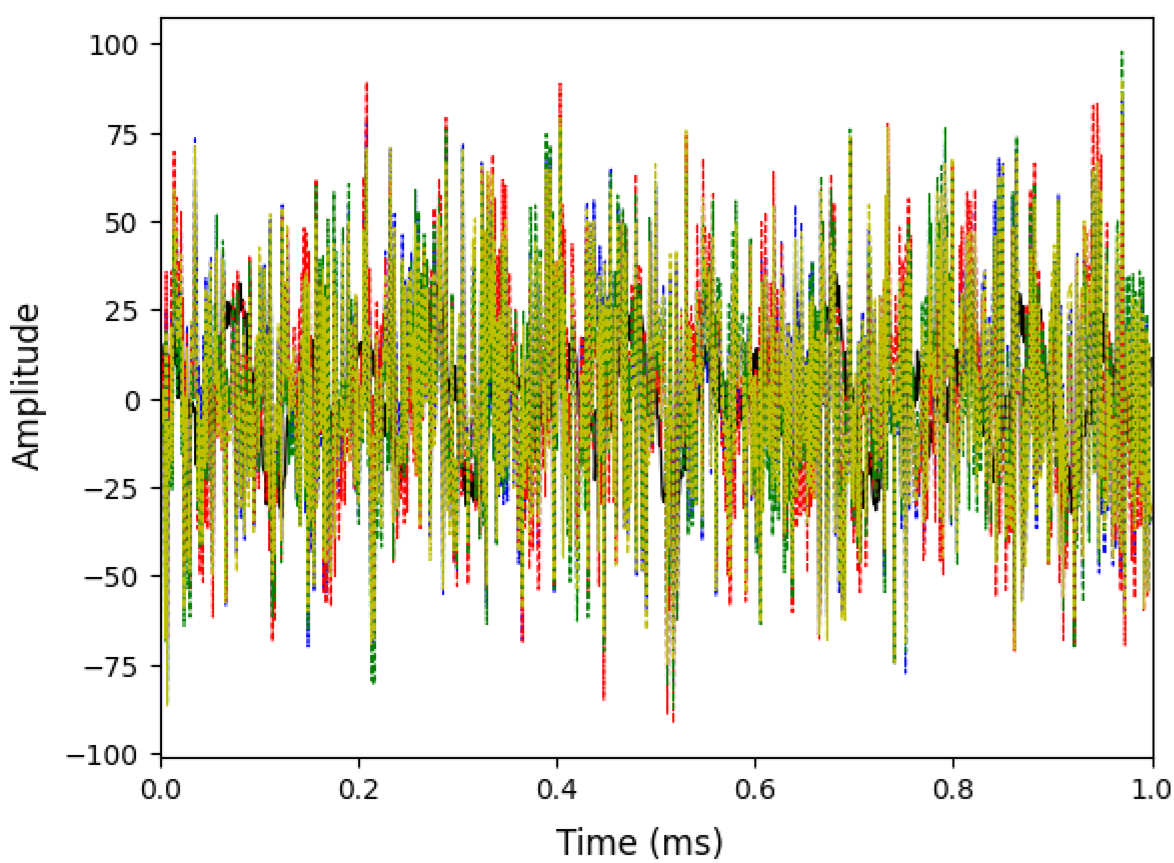
Individual sine waves with a large amount of noise



Sum of sine waves with a large amount of noise

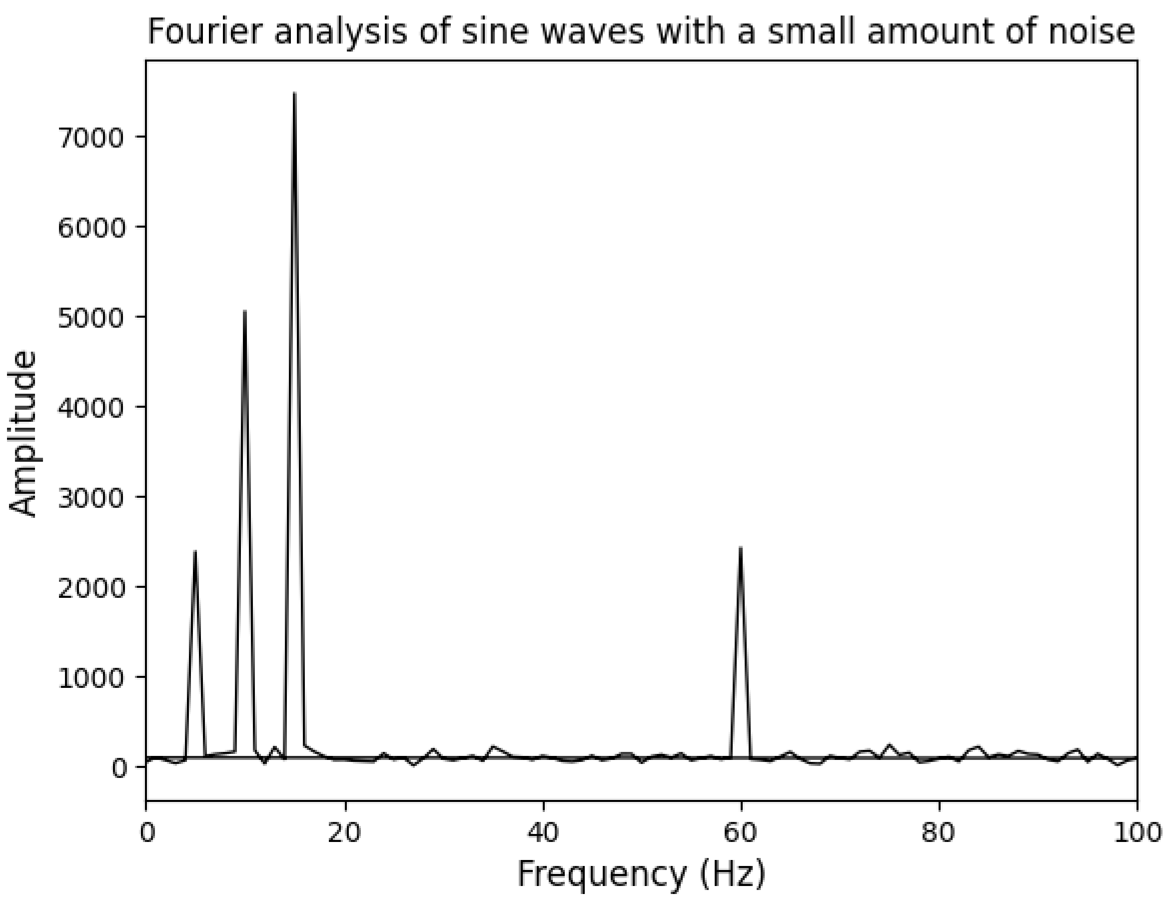


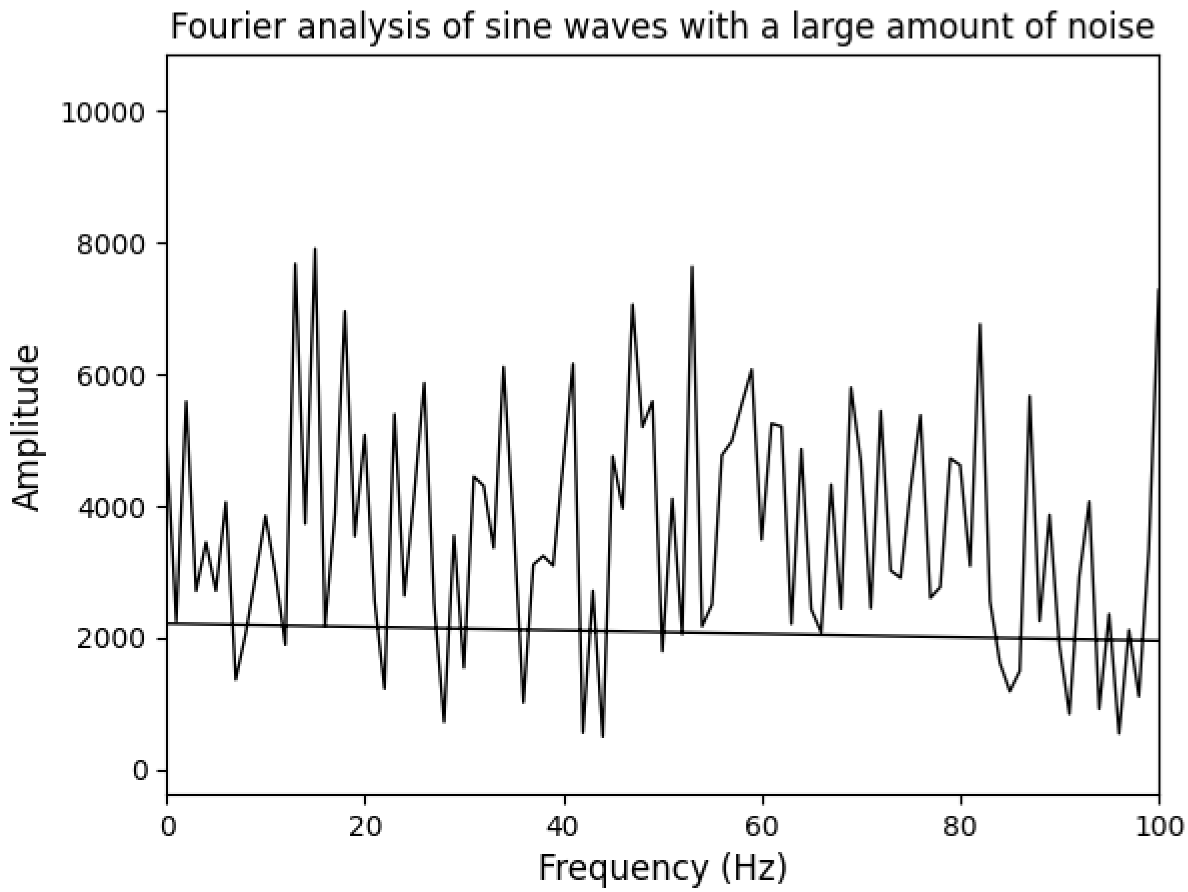
Sum of sine waves/Individual sine waves with a large amount of noise



In [8]:
```
# Fourier analysis

N = fs * 1
```

```
yf_sn = fft(y_sn)
xf_sn = fftfreq(N, ts)

plt.figure(1)
plt.plot(xf_sn, np.abs(yf_sn), 'k', linewidth = 1)
plt.title('Fourier analysis of sine waves with a small amount of noise')
plt.xlabel('Frequency (Hz)', fontsize = 12)
plt.ylabel('Amplitude', fontsize = 12)
plt.xlim(0,100)
plt.show()

yf_ln = fft(y_ln)
xf_ln = fftfreq(N, ts)

plt.figure(1)
plt.plot(xf_ln, np.abs(yf_ln), 'k', linewidth = 1)
plt.title('Fourier analysis of sine waves with a large amount of noise')
plt.xlabel('Frequency (Hz)', fontsize = 12)
plt.ylabel('Amplitude', fontsize = 12)
plt.xlim(0,100)
plt.show()
```



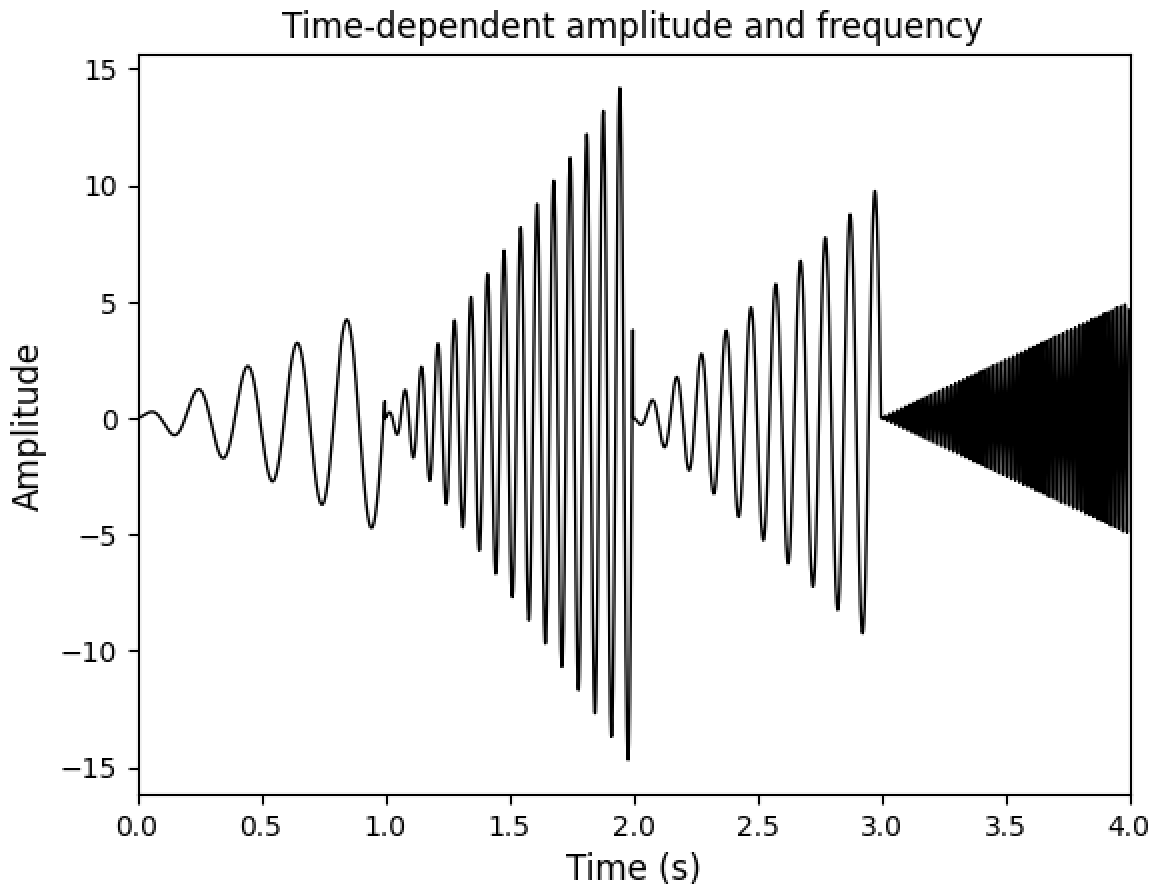


As can clearly be seen in the figures, the effect of a small amount of noise is negligible in the power spectrum. However, the effect of a large amount of noise is considerable. The sine waves with noise are easier to detect in the frequency domain. As shown in the Fourier analysis, the corresponding frequency components of the generated sine waves are easily visible (5, 10, 15, 60 Hz) in the frequency domain.

- **Part C:**

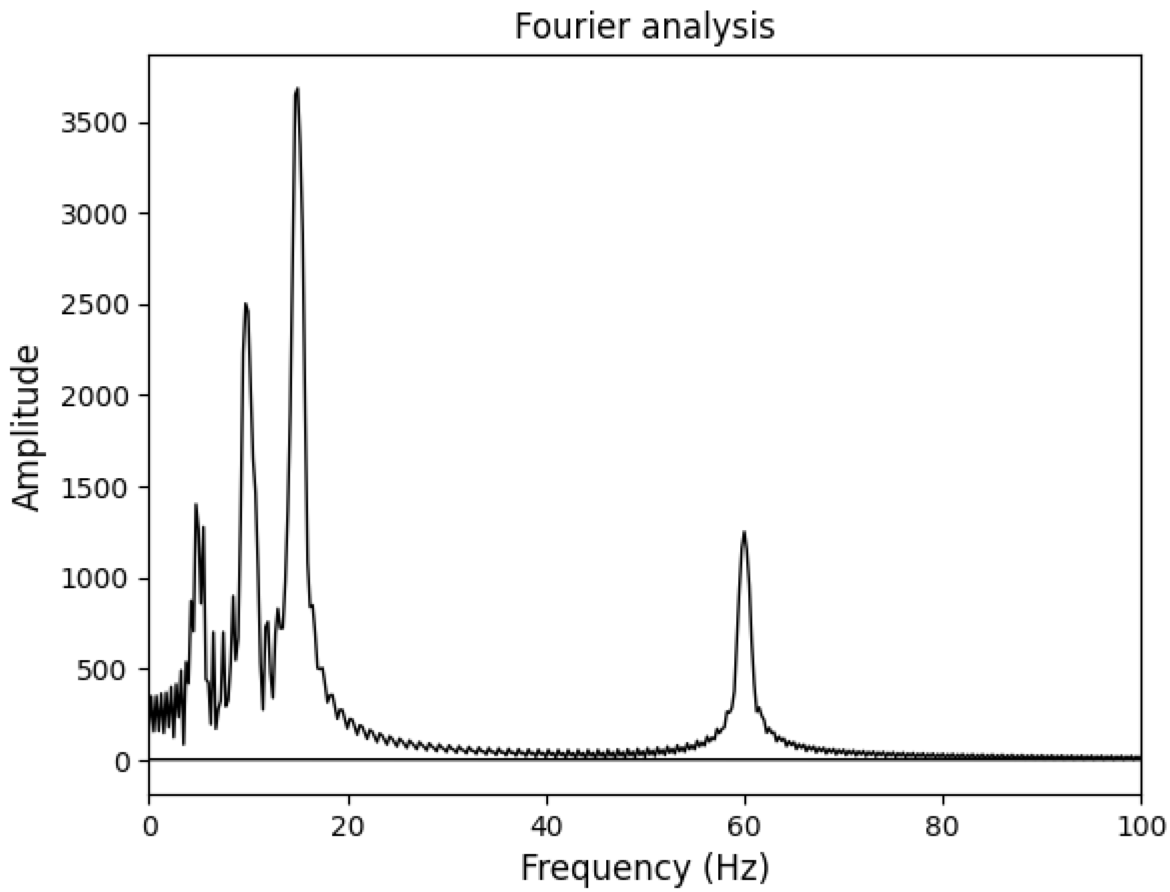
In [9]:

```
# Create a nonstationary time series using the sine waves from section A

t_nonsta = np.arange(0, 4, ts)
y_nonsta = np.array([t*y1, t*y2, t*y3, t*y4])
y_nonsta = y_nonsta.reshape((4000,))

plt.plot(t_nonsta, y_nonsta, 'k', linewidth = 1)
plt.title('Time-dependent amplitude and frequency')
plt.xlabel('Time (s)', fontsize = 12)
plt.ylabel('Amplitude', fontsize = 12)
plt.xlim(0,4)
plt.show()
```



In [10]:

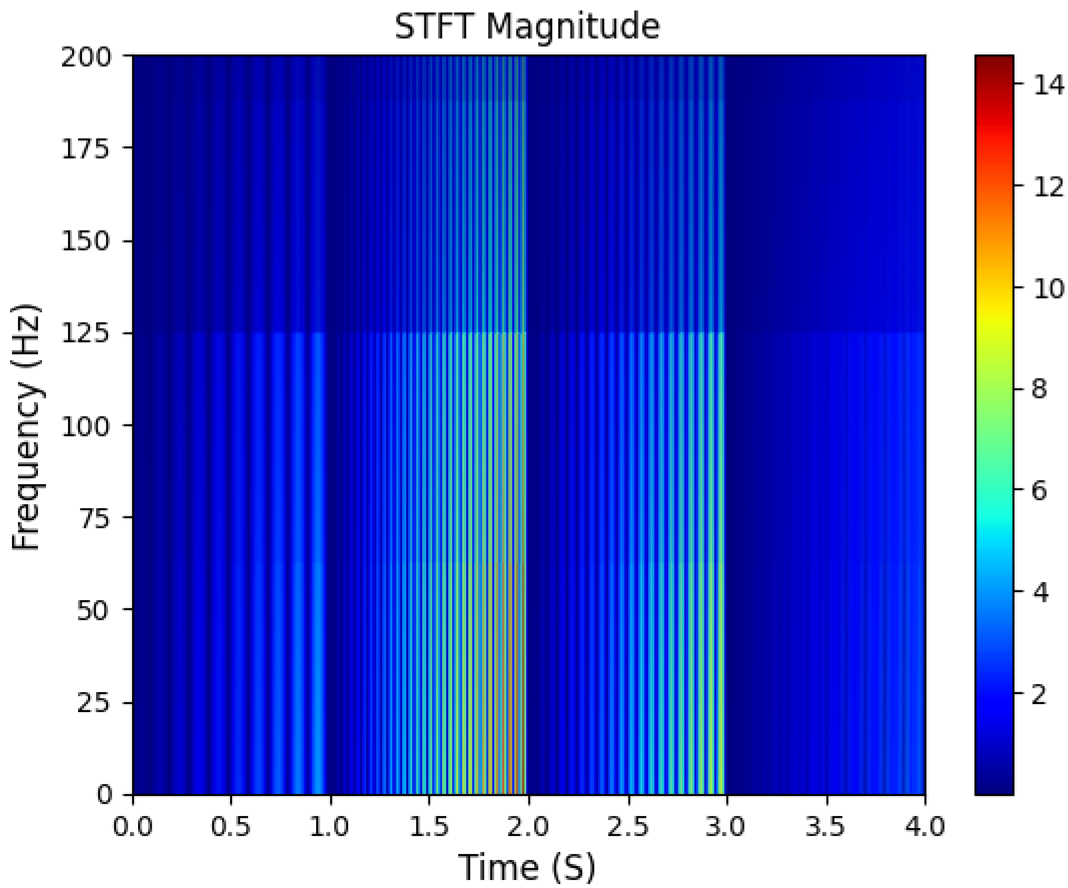
```
# Fourier analysis
from scipy.fft import fft, fftfreq

# Number of samples
N = fs * 4

yf_nonsta = fft(y_nonsta)
xf_nonsta = fftfreq(N, ts)

plt.plot(xf_nonsta, np.abs(yf_nonsta), 'k', linewidth = 1)
plt.title('Fourier analysis')
plt.xlabel('Frequency (Hz)', fontsize = 12)
plt.ylabel('Amplitude', fontsize = 12)
plt.xlim(0,100)
plt.show()
```



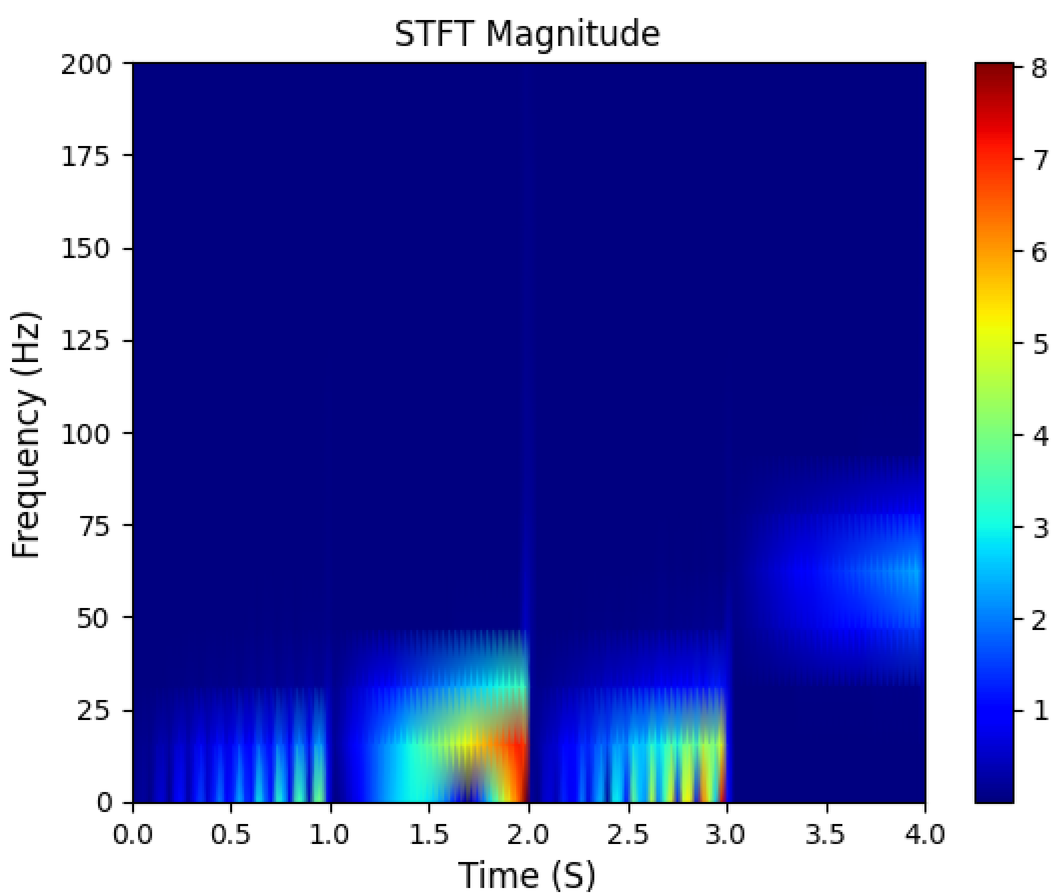
- **Part C:**

In [11]:
```
# STFT imaging
from scipy import signal
def STFT_plot(x, window_size):
    fs=1000
    f, t, Zxx = signal.stft(x, fs, nperseg=window_size)
    plt.pcolormesh(t, f, np.abs(Zxx), shading='gouraud', cmap='jet')
    plt.axis([t[0], t[-1], 0, 200])
    plt.title('STFT Magnitude')
    plt.ylabel('Frequency (Hz)', fontsize = 12)
    plt.xlabel('Time (S)', fontsize = 12)
    plt.colorbar()
```

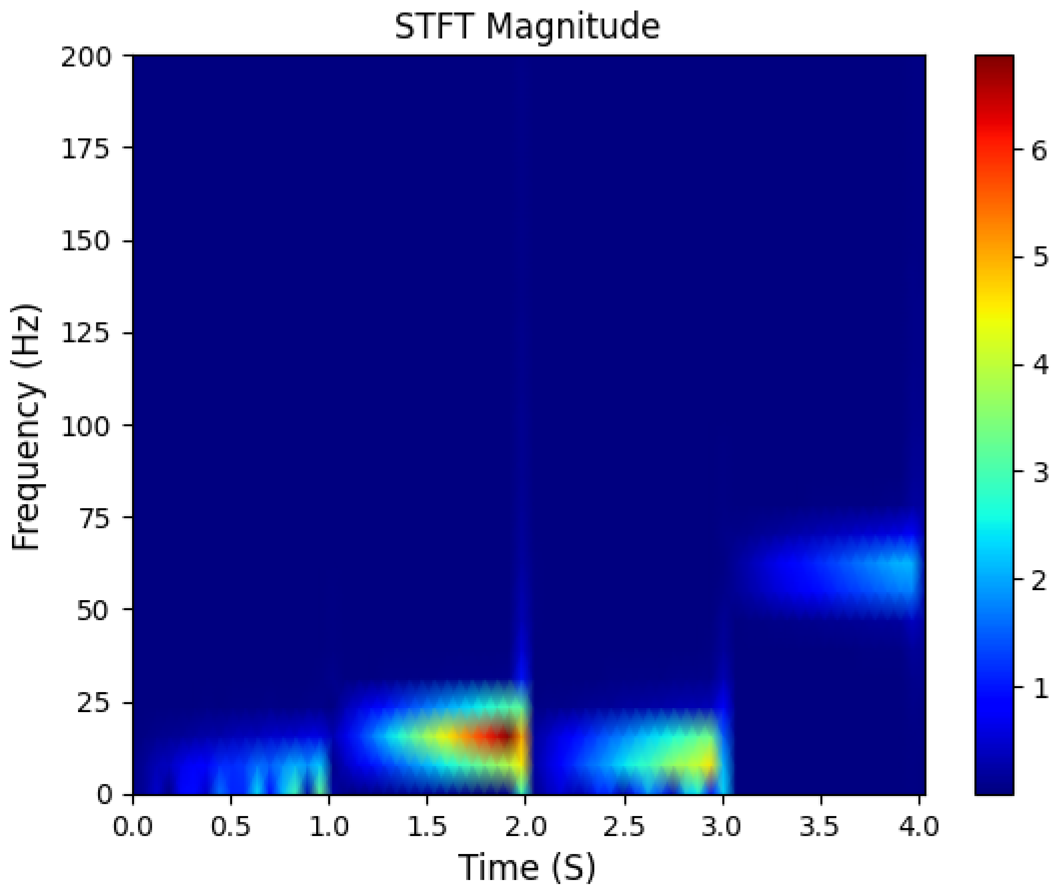
In [12]:
```
STFT_plot(y_nonsta, 8)
```



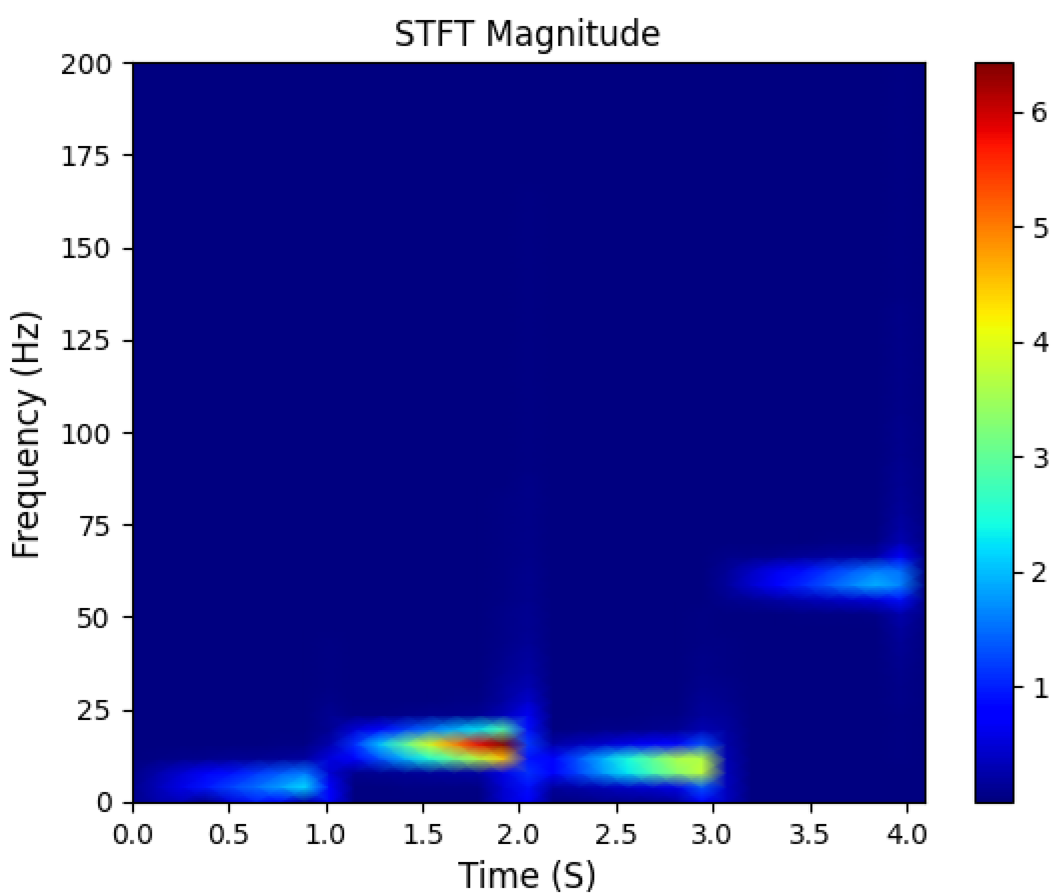
In [13]:
```
STFT_plot(y_nonsta, 64)
```



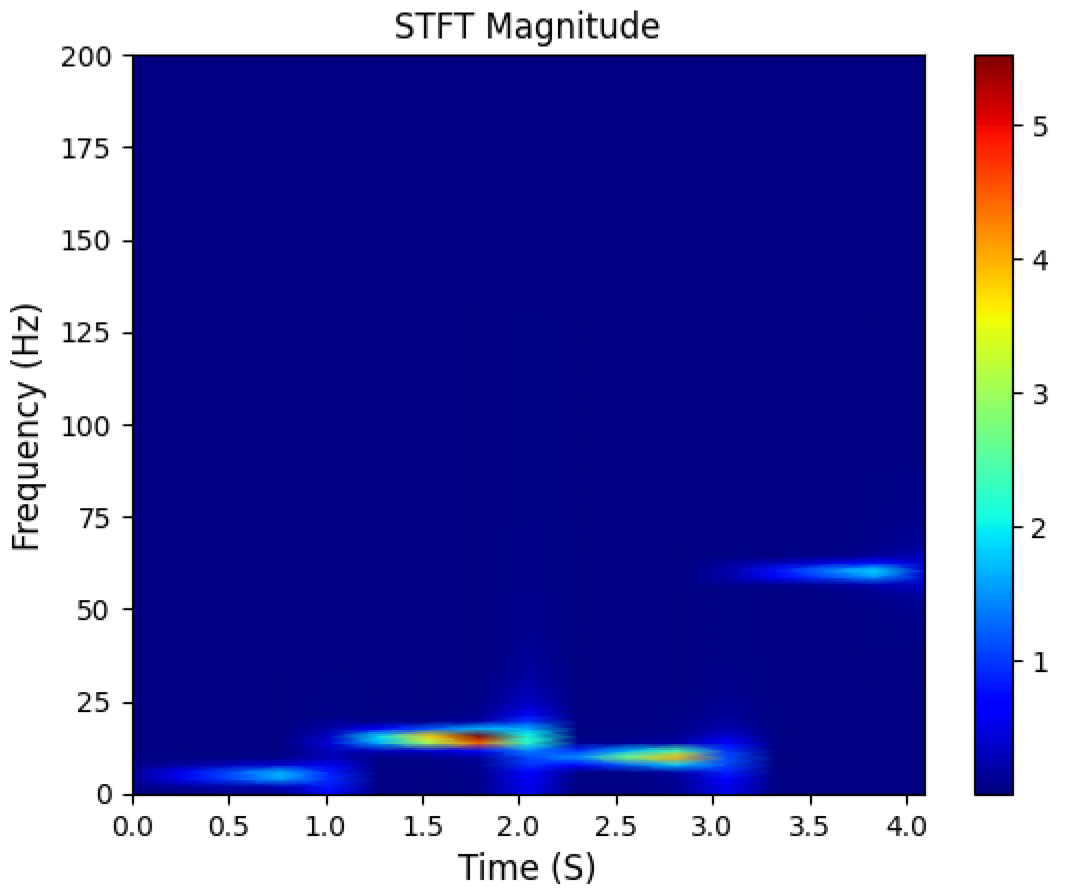
In [14]:
```
STFT_plot(y_nonsta, 128)
```



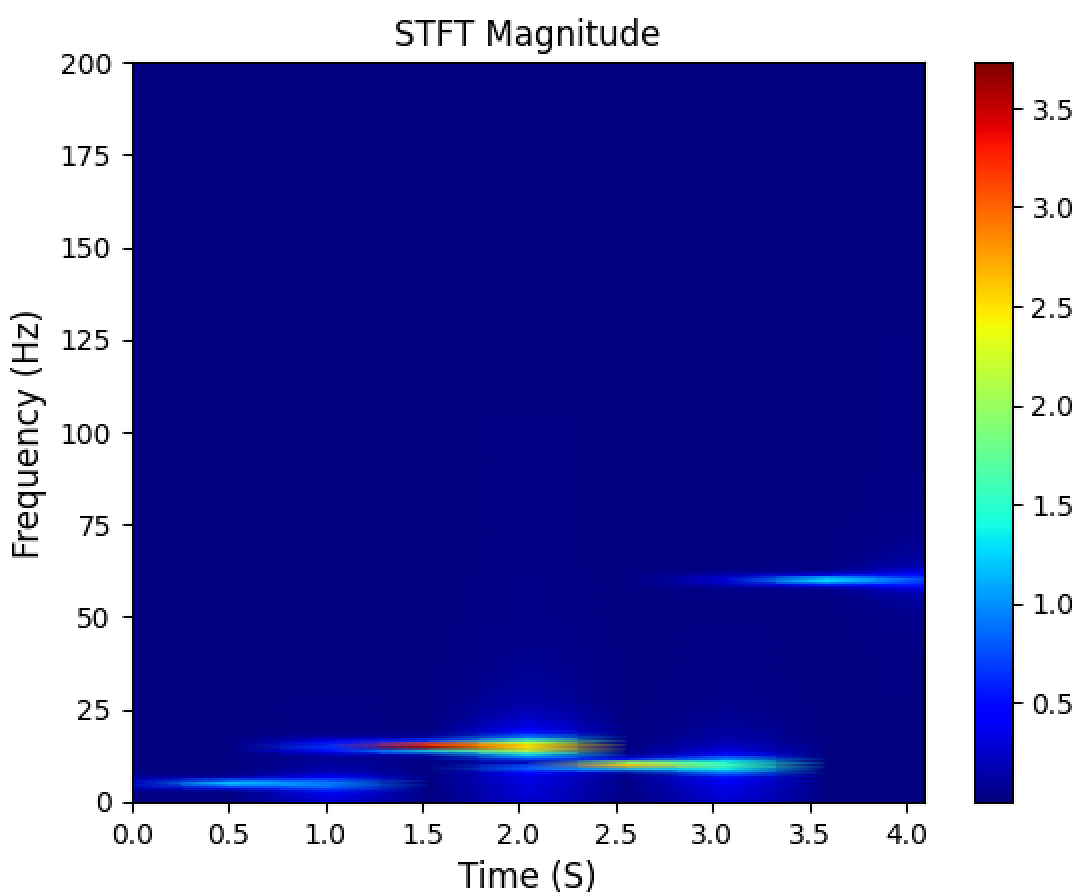
In [15]:
```
STFT_plot(y_nonsta, 256)
```



In [16]:
```
STFT_plot(y_nonsta, 512)
```



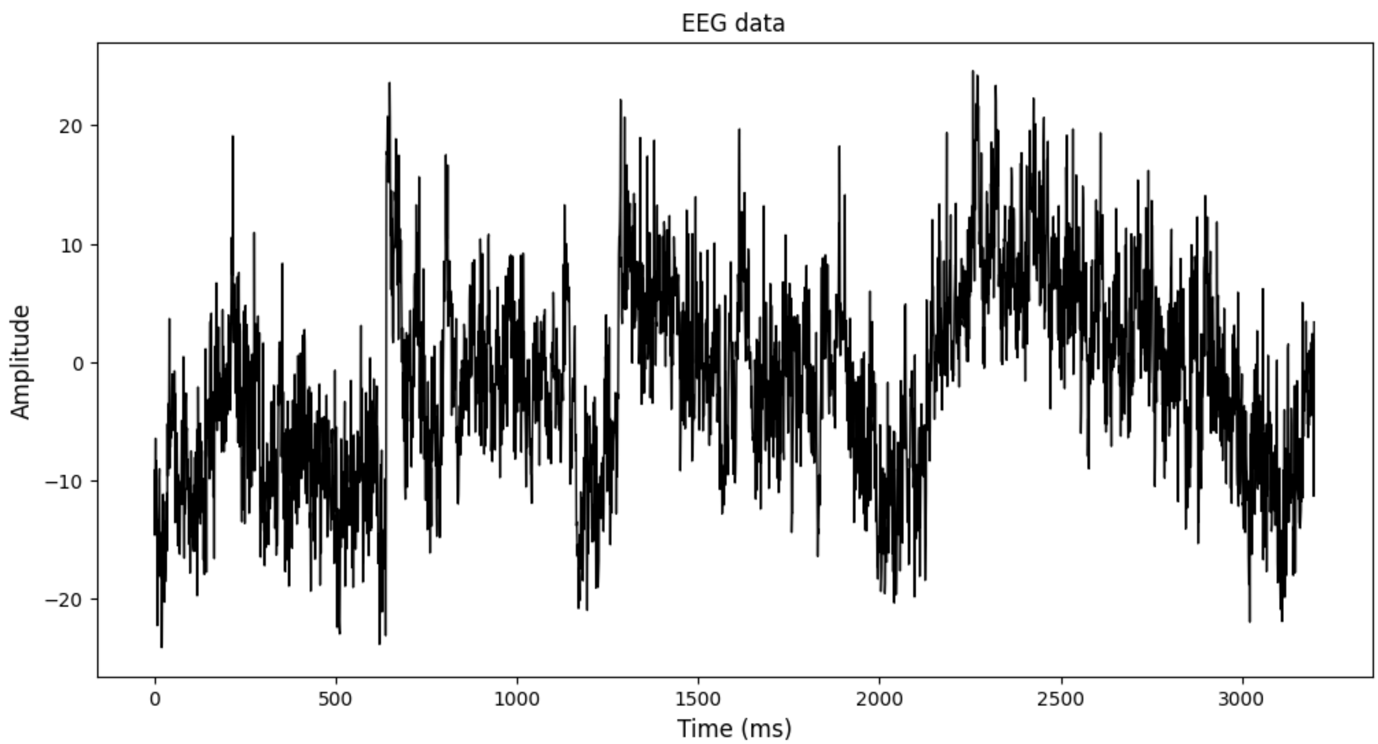

```
In [17]: STFT_plot(y_nonsta, 1024)
```



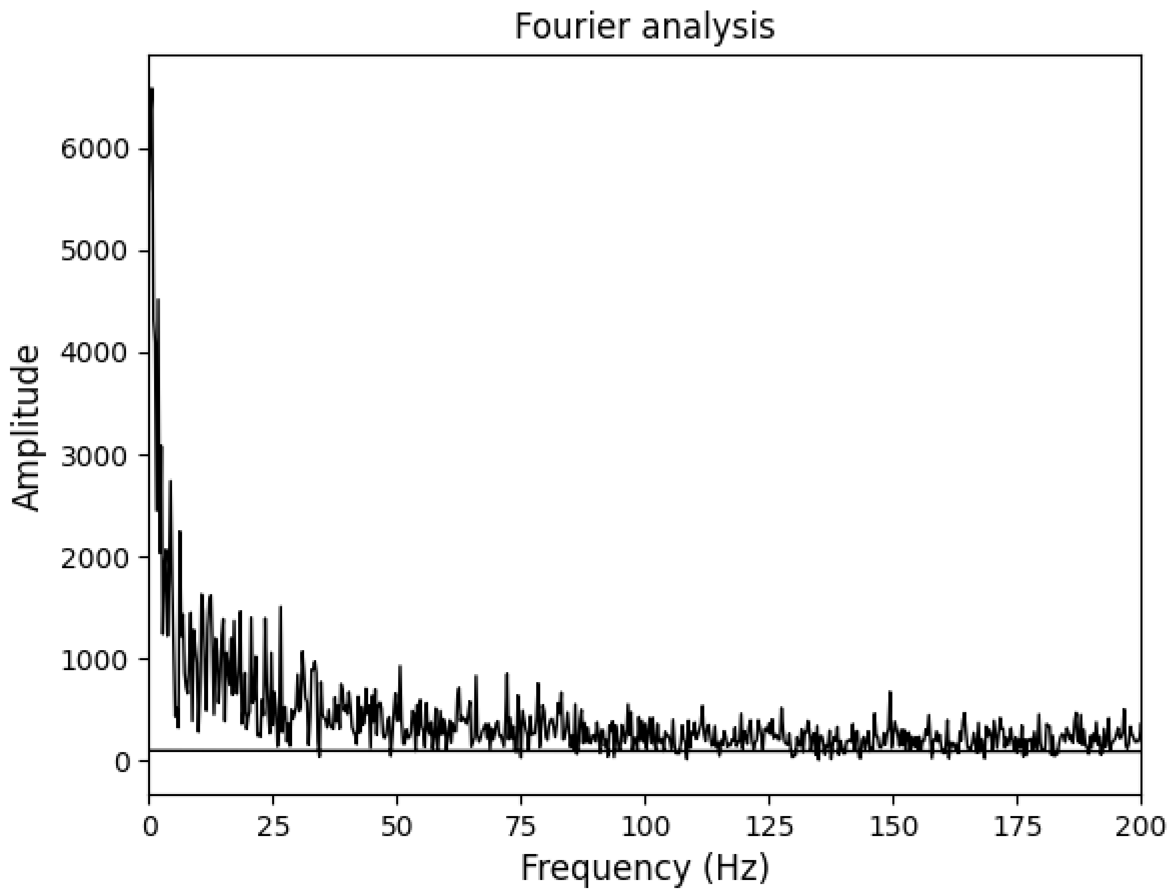
From the observed results, it is revealed that the smaller the window size, the easier the identification of the time component, while the resolution for the frequency components is low. The opposite occurs for larger windows, where the identification of the frequency components is easier, but the resolution for the time components is low. As a result, it seems there should be a trade-off to selecting the window size for different tasks.

- **Part D:**

In [18]:
```
# Load and visualization of the data
import scipy.io
data = scipy.io.loadmat('dataset.mat')['eeg']
data = data.flatten()

plt.figure(figsize=(12, 6))
plt.plot(data, 'k', linewidth = 1)
plt.title('EEG data')
plt.xlabel('Time (ms)', fontsize = 12)
plt.ylabel('Amplitude', fontsize = 12)
plt.show()
```

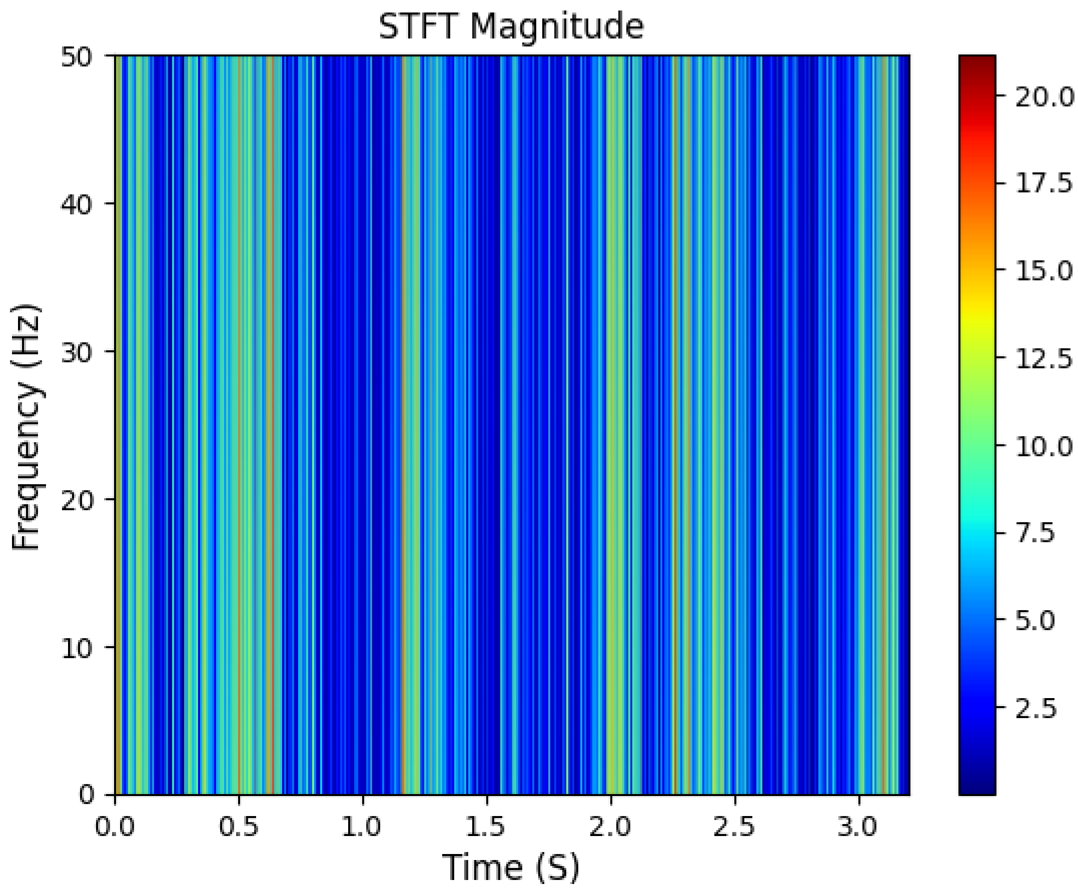


In [19]:
```
# Fourier analysis

N = int(fs * 3.2)

yf_data = fft(data)
xf_data = fftfreq(N, ts)

plt.figure(1)
plt.plot(xf_data, np.abs(yf_data), 'k', linewidth = 1)
plt.title('Fourier analysis')
plt.xlabel('Frequency (Hz)', fontsize = 12)
plt.ylabel('Amplitude', fontsize = 12)
plt.xlim(0,200)
plt.show()
```

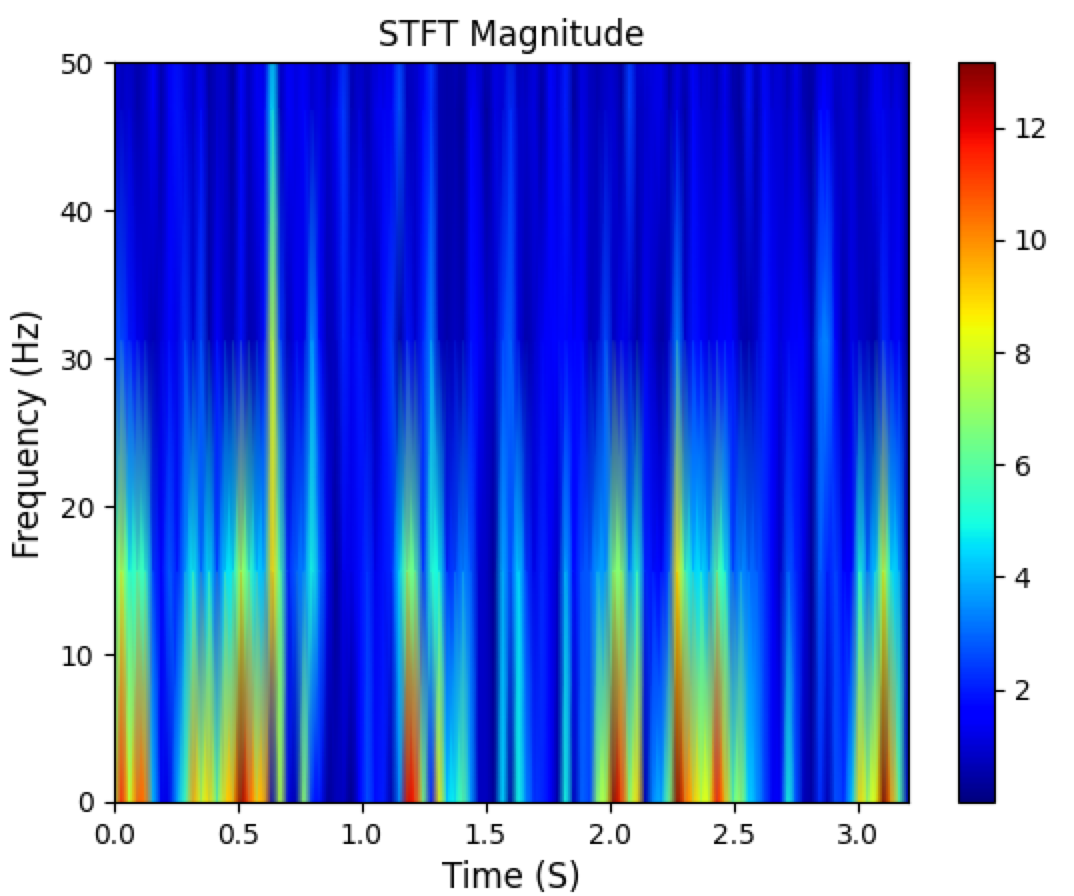


In [20]:
```
from scipy import signal
def STFT_plot_data(x, window_size):
    fs=1000
    f, t, Zxx = signal.stft(x, fs, nperseg=window_size)
    plt.pcolormesh(t, f, np.abs(Zxx), shading='gouraud', cmap='jet')
    plt.axis([t[0], t[-1], 0, 50])
    plt.title('STFT Magnitude')
    plt.ylabel('Frequency (Hz)', fontsize = 12)
    plt.xlabel('Time (S)', fontsize = 12)
    plt.colorbar()
```

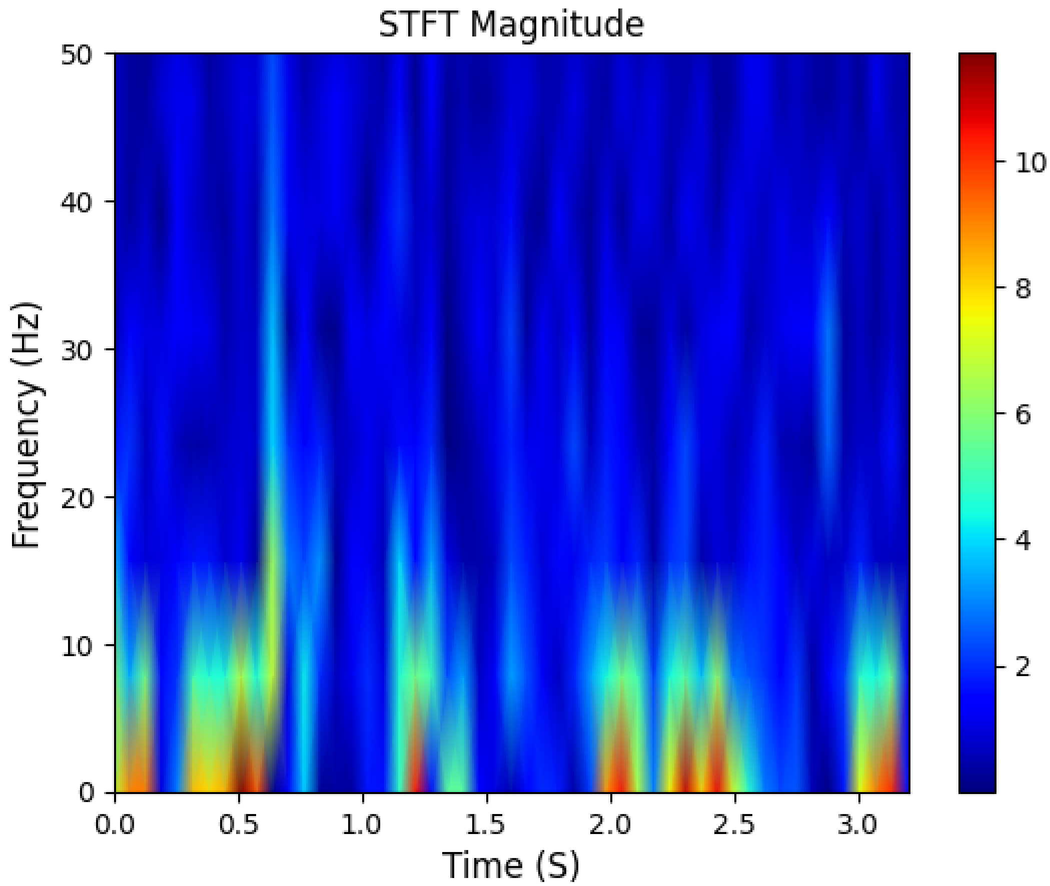
In [21]:
```
STFT_plot_data(data, 8)
```



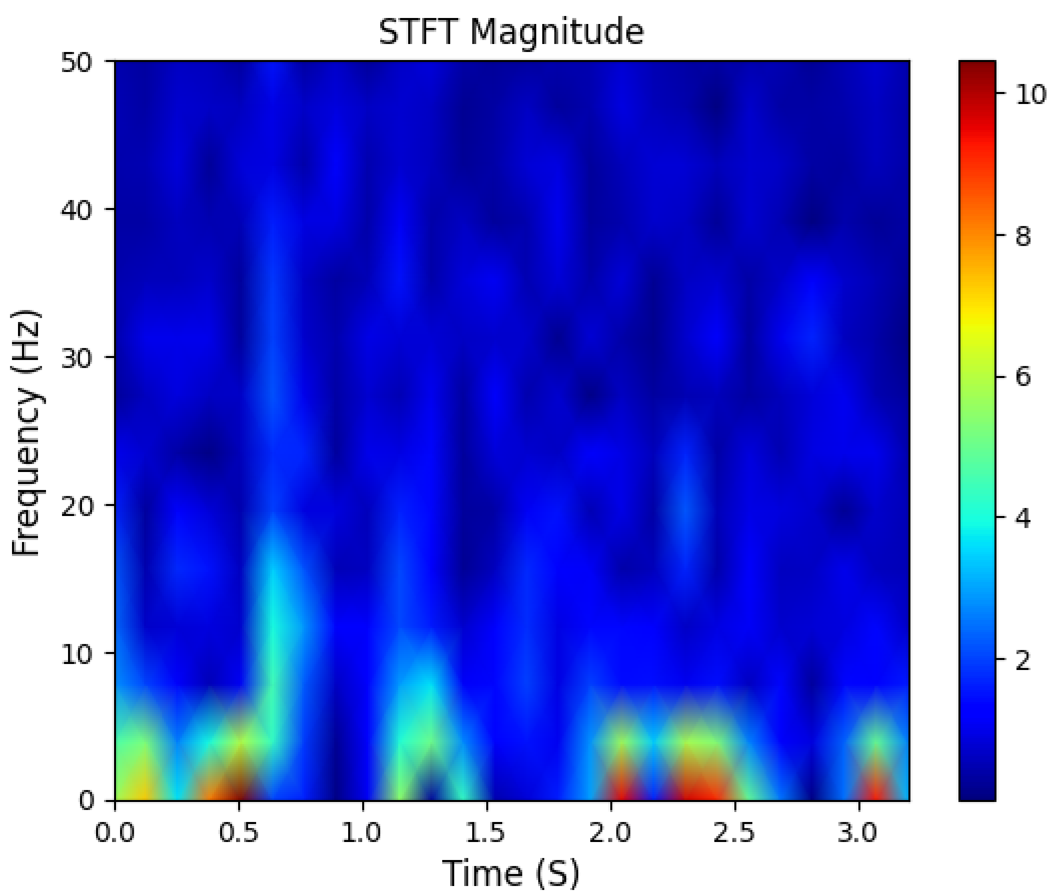
In [22]:
```
STFT_plot_data(data, 64)
```



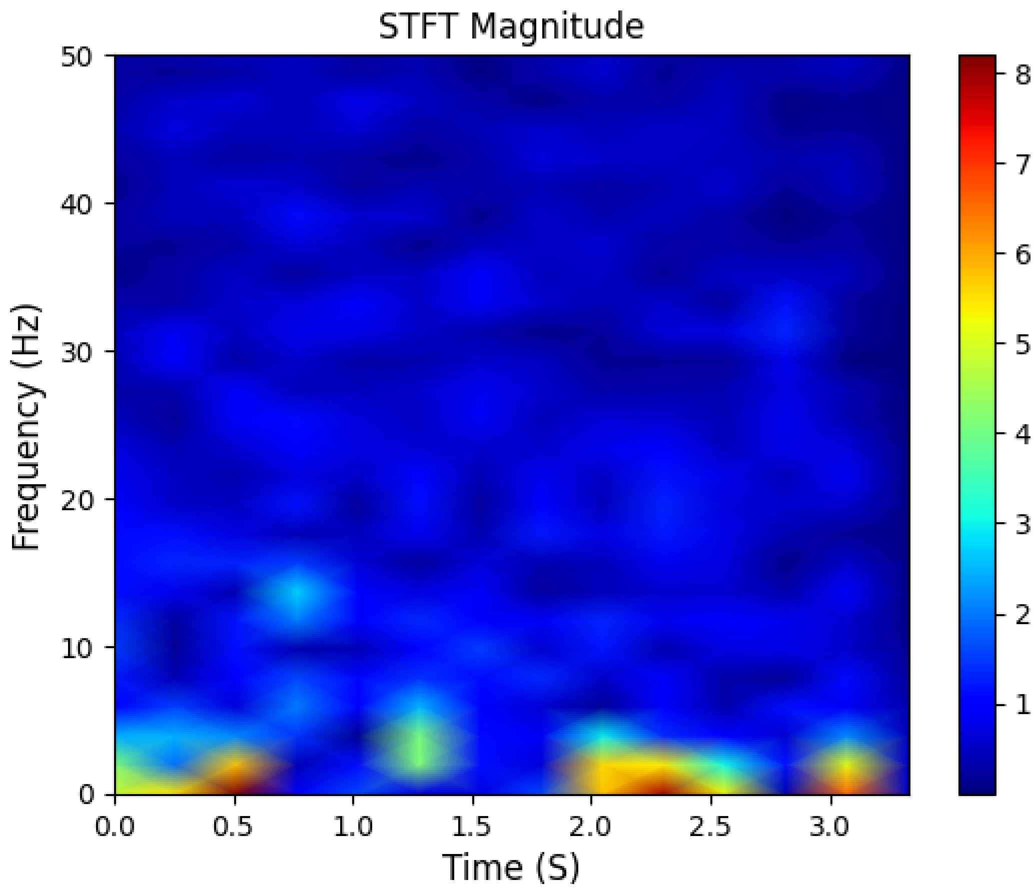
In [23]:
```
STFT_plot_data(data, 128)
```



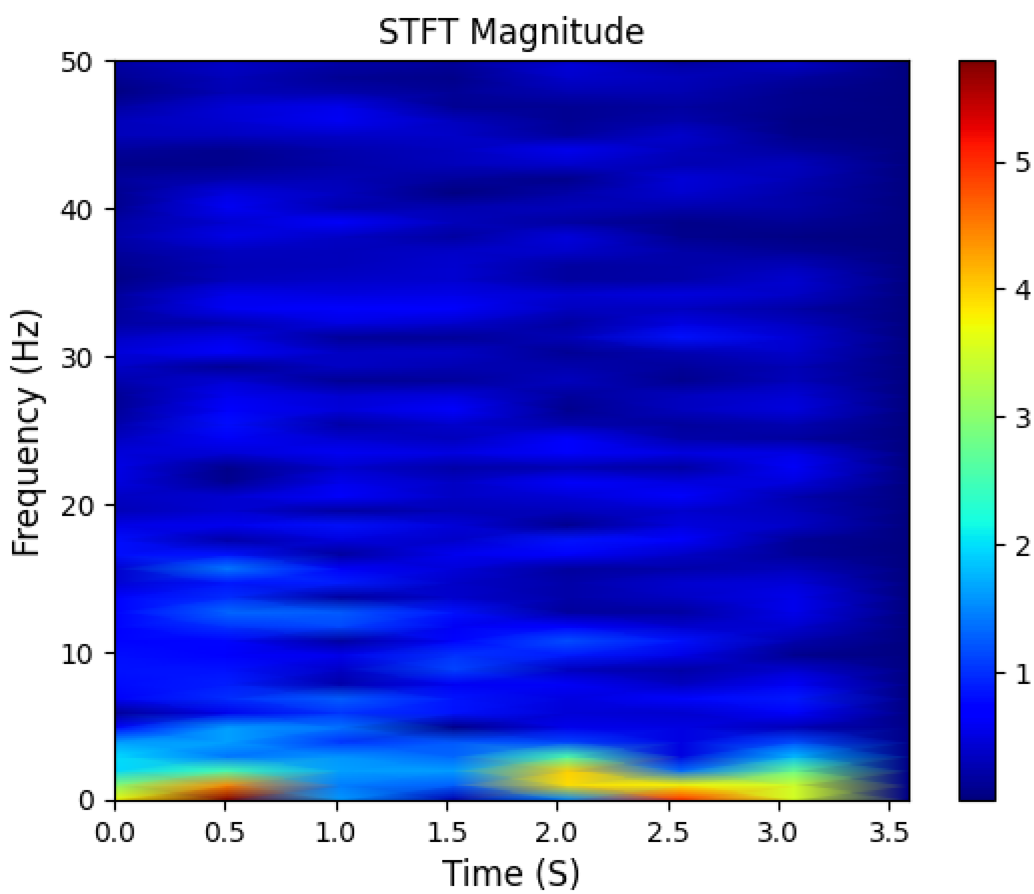
In [24]:
```
STFT_plot_data(data, 256)
```



In [25]:
```
STFT_plot_data(data, 512)
```



In [26]:
```
STFT_plot_data(data, 1024)
```



There is difficult to exactly say the number of trials after analysing the STFT plots. If we considered the EEG records was gathered during a resting-state (i.e, as low movement as possible), we could say there are 6 trials. If the EEG records was gathered during an activity (i.e, as high movement as possible), we could say there are 4 trials.