NATIONAL INSTITUTE OF TECHNOLOGY SURATHKAL

MANGALORE, KARNATAKA-575025

LAB ASSIGNMENT :-08

NAME :- CHIKKERI CHINMAYA

ROLL NO :- 211IT017

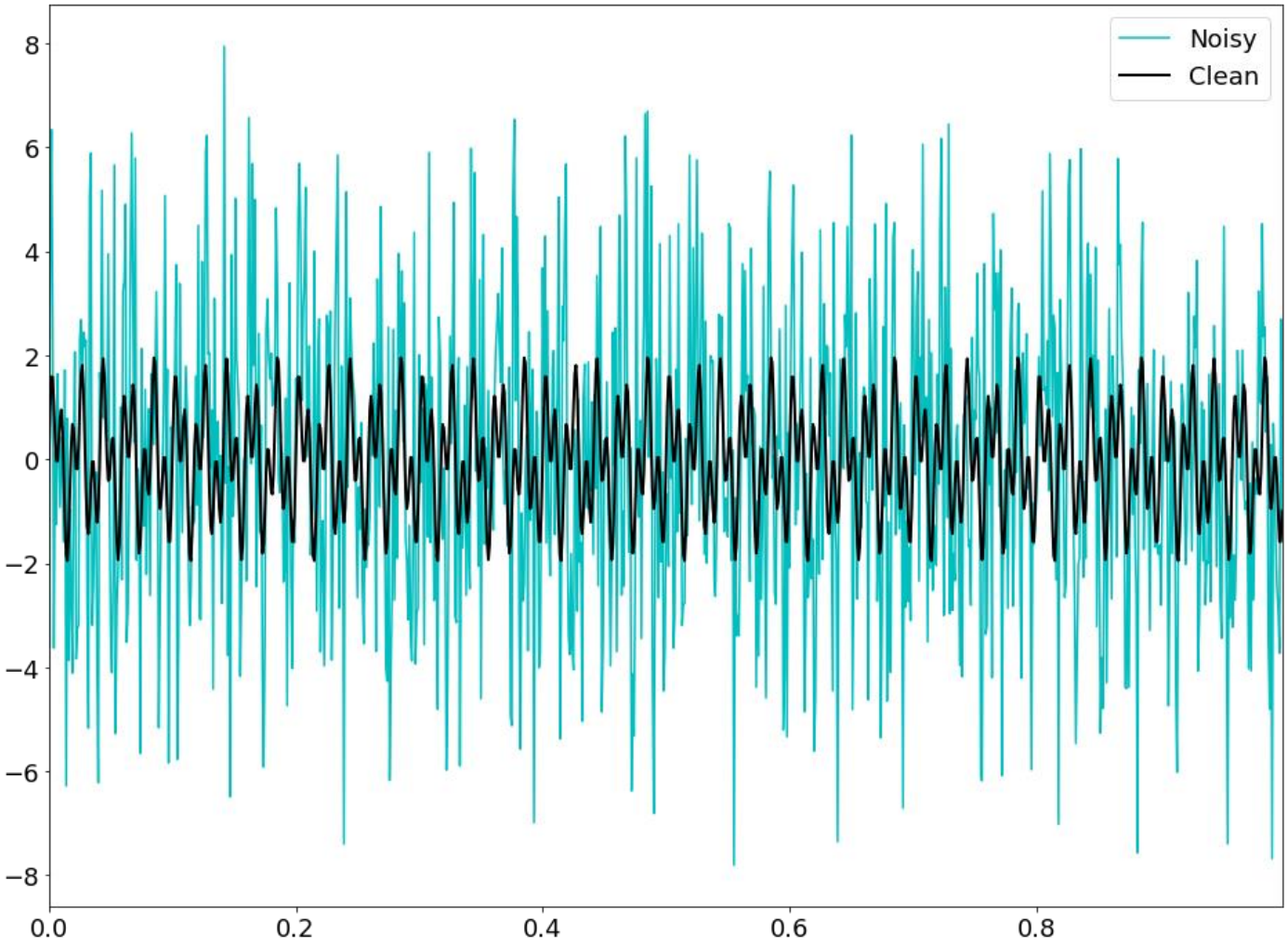
COURSE :- B.TECH (INFORMATION TECHNOLOGY)

SUBJECT :- IT204 (SIGNALS AND SYSTEM LAB)

SUBMITTED TO :-

REVANESH M SIR

QUESTION 01)



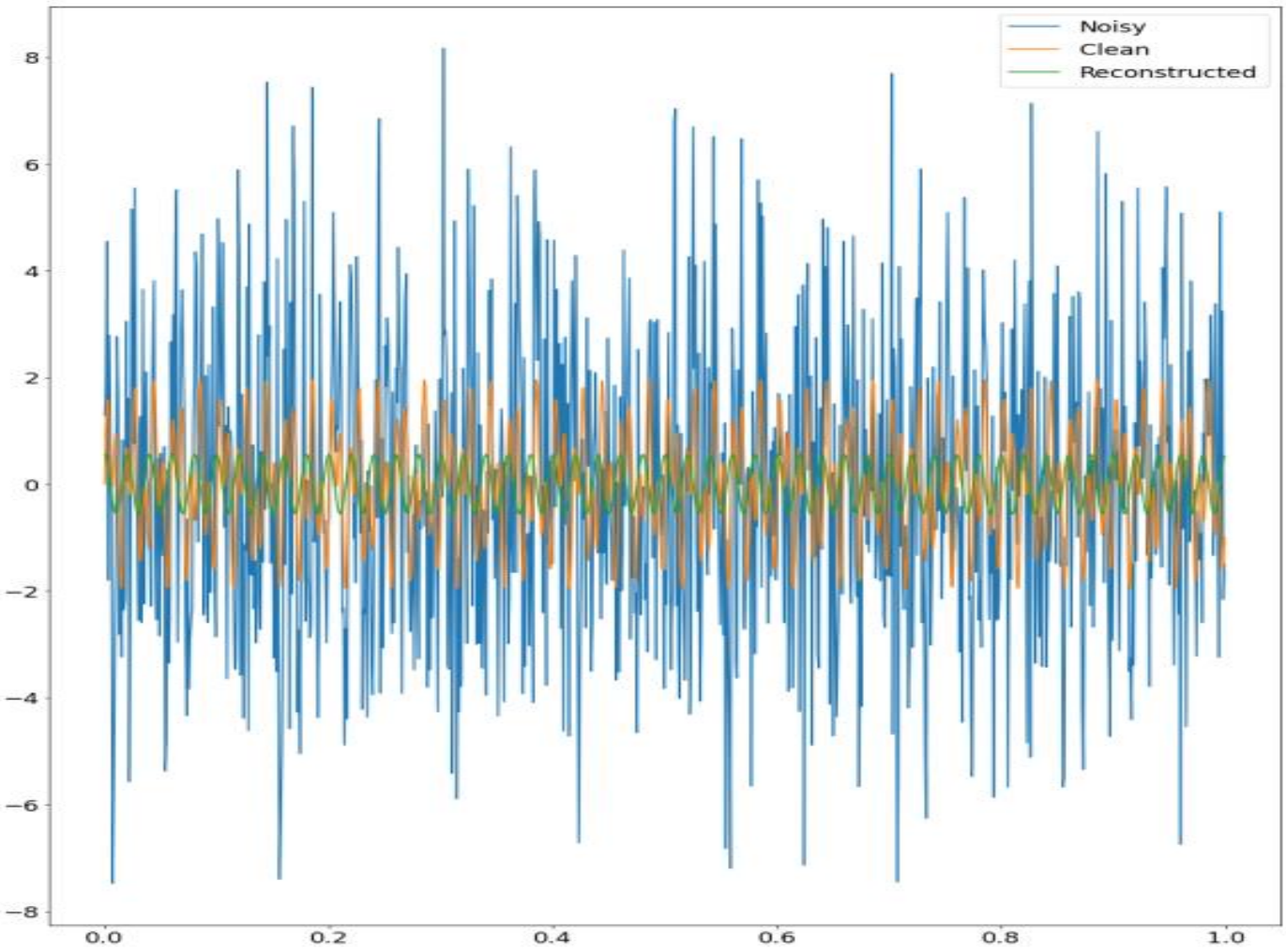
This code generates a noisy signal by adding some noise to a simple signal with two frequencies. The code creates an array t of time values from 0 to 1 with a time step of dt, which is 0.001. Then, it creates a signal f as the sum of two sinusoidal functions with frequencies 50 Hz and 120 Hz. This signal f is the "clean" signal, without noise.

Then, the code adds some noise to the clean signal f by adding a random normal distribution with mean 0 and standard deviation 2.5 to each element in f.

The resulting signal is the noisy signal. Finally, the code plots both the noisy and clean signals on the same plot for comparison.

QUESTION 02)

The code will produce a graph showing the original noisy signal in blue, the clean signal in orange, and the reconstructed signal in green. The reconstructed signal should resemble the original clean signal and be much closer to it than the noisy signal.




```python
#code
import numpy as np
import matplotlib.pyplot as plt

# Generate the noisy signal as before
dt = 0.001
t = np.arange(0,1,dt)
f = np.sin(2*np.pi*50*t) + np.sin(2*np.pi*120*t)
f_clean = f
f = f + 2.5*np.random.randn(len(t))

# Perform FFT on noisy signal
f_fft = np.fft.fft(f)

# Get magnitudes of frequency components
f_magnitudes = np.abs(f_fft)

# Find index of maximum magnitude
max_index = np.argmax(f_magnitudes)

# Set all magnitudes except the one at the max index to 0
f_magnitudes_filtered = np.copy(f_magnitudes)
f_magnitudes_filtered[:max_index] = 0
f_magnitudes_filtered[max_index+1:] = 0
```

```
# Use filtered magnitudes to reconstruct the signal
f_ifft = np.fft.ifft(f_magnitudes_filtered)

# Plot the results
plt.plot(t, f, label='Noisy')
plt.plot(t, f_clean, label='Clean')
plt.plot(t, f_ifft, label='Reconstructed')
plt.legend()
```

This code is using the Fast Fourier Transform (FFT) to extract the frequency components of a noisy signal, then filtering out all but the frequency with the highest magnitude, and using the inverse FFT (IFFT) to reconstruct the filtered signal.

Steps

1. Import the necessary libraries, such as numpy and matplotlib.pyplot.

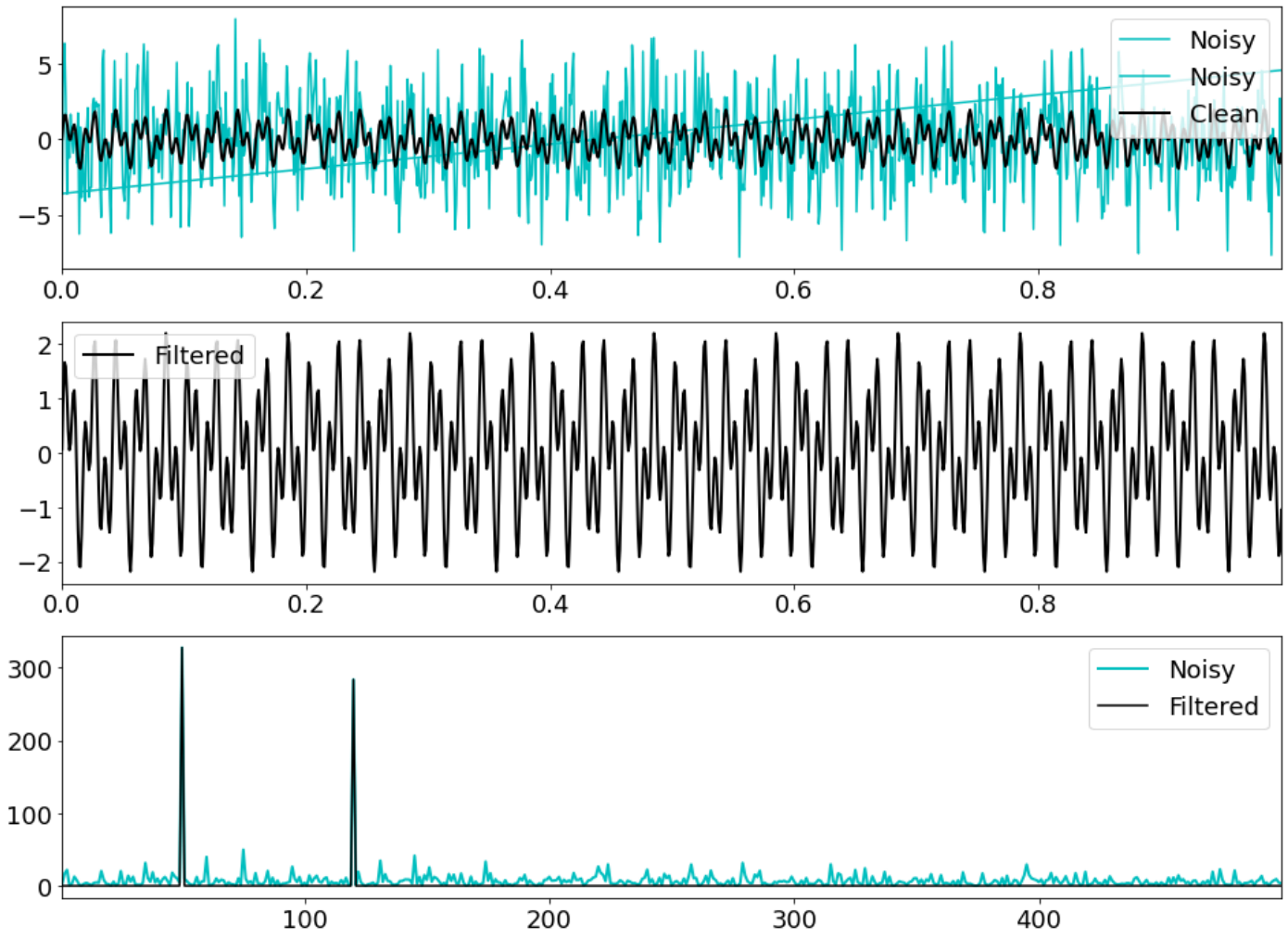
2. Generate the noisy signal as before. You can use the same code as in the previous example to do this.

3. Use the fft function from numpy to perform the FFT on the noisy signal. This will return the frequency components of the signal in the form of complex numbers.

4. Use the abs function from numpy to compute the magnitudes of the frequency components. This will give you an array of the same size as the original signal, with the magnitudes of the frequency components at each time index.

5. Find the index of the maximum magnitude in the array of magnitudes. This index corresponds to the frequency component with the highest amplitude in the original signal.

6. Use the ifft function from numpy to perform the inverse FFT (IFFT) on the array of frequency components, setting all the components except the one with the highest amplitude to 0. This will give you an approximation of the clean signal with only one frequency component. 7. Plot the resulting signal to visualize the result



Plotting the original noisy signal, the clean signal, and the filtered signal in separate subplots. creating 3 subplots on the same figure using **plt.subplots(3,1)**, where each subplot occupies one row of the figure. 3 different subplots , the first one for the time domain of original and filtered signal, second for only filtered signal and last one for the frequency domain of original and filtered signals.