Task3: Signal Processing II

Name: Noreen Gao

Degree: BA

ID: 20786725

```
# import libraries
import matplotlib.pyplot as plt
import numpy as np
import scipy.io
```

Problem 1: Convolution (10 points)

$$f(t) = \begin{Bmatrix} A, & a > |t| \\ 0, & otherwise \end{Bmatrix} \qquad g(t) = \begin{Bmatrix} B, & b > |t| \\ 0, & otherwise \end{Bmatrix}$$

where A = 10, a= 3, and B = 7, b = 3

(a) Compute an analytic y(t) which is the convolution of f(t) and g(t):

$$y(t) = f(t) * g(t)$$

1(a) $f(t) = \begin{cases} 10, & |t|<3 \\ 0, & \text{otherwise} \end{cases}$ $g(t) = \begin{cases} 7, & |t|<3 \\ 0, & \text{otherwise} \end{cases}$

$g(t-\tau) = \begin{cases} 7, & -6 < t-\tau < 6 \\ 0, & \text{otherwise} \end{cases}$

$$\begin{aligned} y(t) &= f(t) * g(t) \\ &= \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \\ &= \int_{-6}^{6} f(\tau)\, g(t-\tau)\, d\tau \\ &= f(\tau) \int_{-6}^{6} g(t-\tau)\, d\tau - \int_{-6}^{6} f'(\tau) \left(\int_{-6}^{6} g(t-\tau) \right) d\tau \quad (f'(\tau) \to 0) \\ &= 10 \int_{-6}^{6} g(t-\tau)\, d\tau \end{aligned}$$

$f(\tau) = 10$ where the function is nonzero
$f'(\tau) = 0$

Case 1: when $-6 < t \le 0$

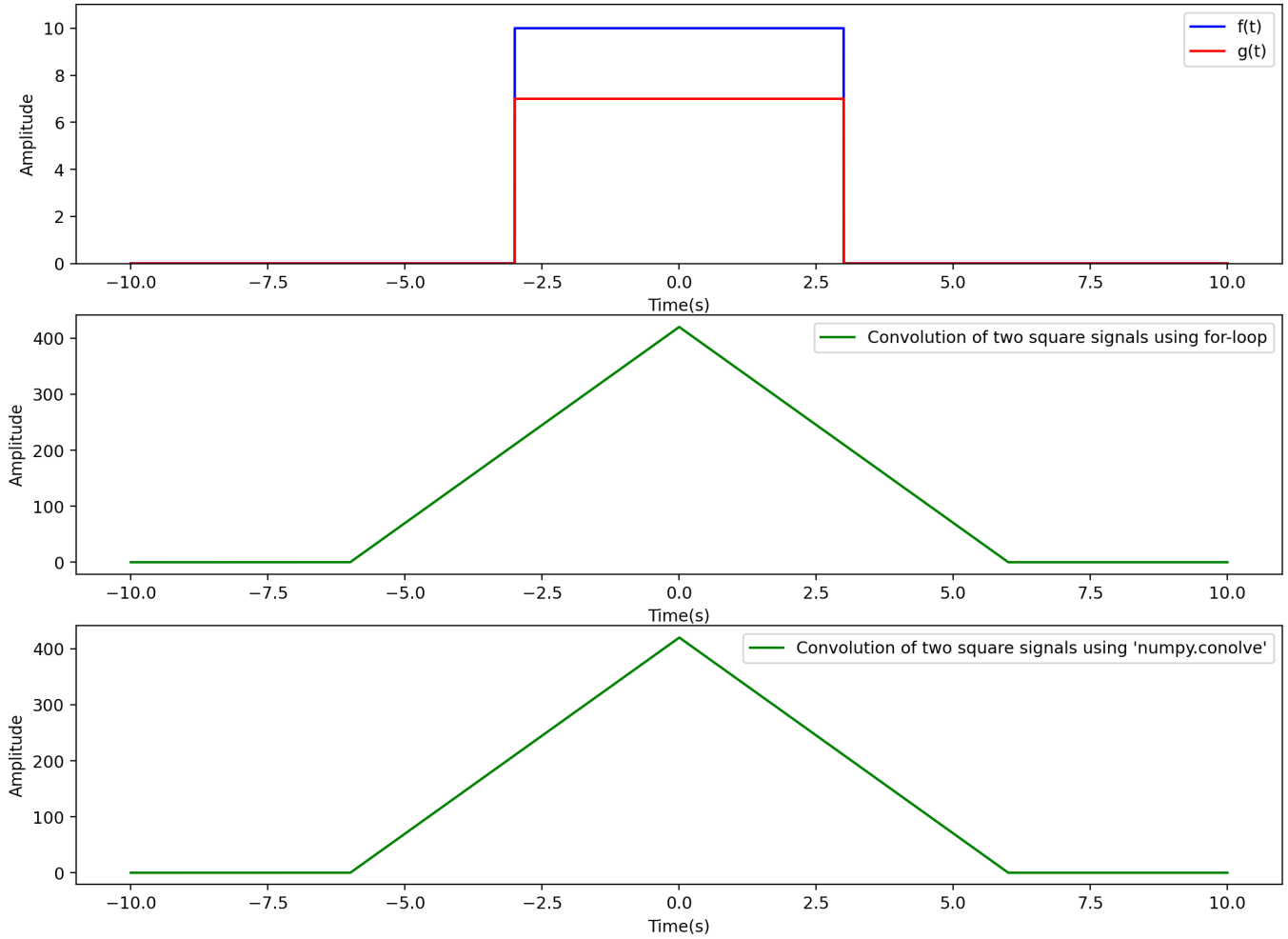
$$\begin{aligned} y(t) &= 10 \int_{-3-t}^{3} 7\, d\tau = 70 \left[\tau\right]_{-3-t}^{3} \\ &= 70[\,3+3+t\,] \\ &= 420 + 70t \end{aligned}$$

Case 2: when $0 < t < 6$:

$$\begin{aligned} y(t) &= 10 \int_{-3}^{3-t} 7\, d\tau = 70 \left[\tau\right]_{-3}^{3-t} \\ &= 70[\,3-t+3\,] \\ &= 420 - 70t \end{aligned}$$

$$\therefore\ y(t) = \begin{cases} 420 + 70t, & -6 < t \le 0 \\ 420 - 70t, & 0 < t < 6 \\ 0, & \text{otherwise} \end{cases}$$

(b) Write a code to numerically compute y(t) and plot y(t). Please use for-loop and do not use conv.




```
fig, ax = plt.subplots(3)
fig.set_dpi(200)
fig.set_size_inches(13, 10, forward=True)

A = 10
a = 3
B = 7
b = 3
Fsa = 1000
t = np.arange(-10,10,1/Fsa)

# plot f(t) and g(t)
f = lambda t: A * np.heaviside(a-np.abs(t),A)
g = lambda t: B * np.heaviside(a-np.abs(t),B)
ax[0].plot(t, f(t), 'b', label = "f(t)")
ax[0].plot(t, g(t), 'r', label = "g(t)")
ax[0].set_ylim([0, 11])
ax[0].legend(loc = 'upper right')
ax[0].set_xlabel("Time(s)")
ax[0].set_ylabel("Amplitude")

# convolution using for-loop numerically
y = []
for tau in t:
    y_interand =  np.multiply(f(t),g(t-tau)) # element-wise multiplication
```

```
    y_sig = np.trapz(y_interand,t)
    y.append(y_sig)
ax[1].plot(t, y, 'g', label = "Convolution of two square signals using
for-loop")
ax[1].legend(loc = 'upper right')
ax[1].set_xlabel("Time(s)")
ax[1].set_ylabel("Amplitude")

# convolution using 'np.convolve' numerically
y_conv = np.convolve(f(t),g(t),'same')/Fsa
ax[2].plot(t, y_conv, 'g', label = "Convolution of two square signals
using 'numpy.conolve'")
ax[2].legend(loc = 'upper right')
ax[2].set_xlabel("Time(s)")
ax[2].set_ylabel("Amplitude")
plt.rcParams.update({'font.size': 8})
```

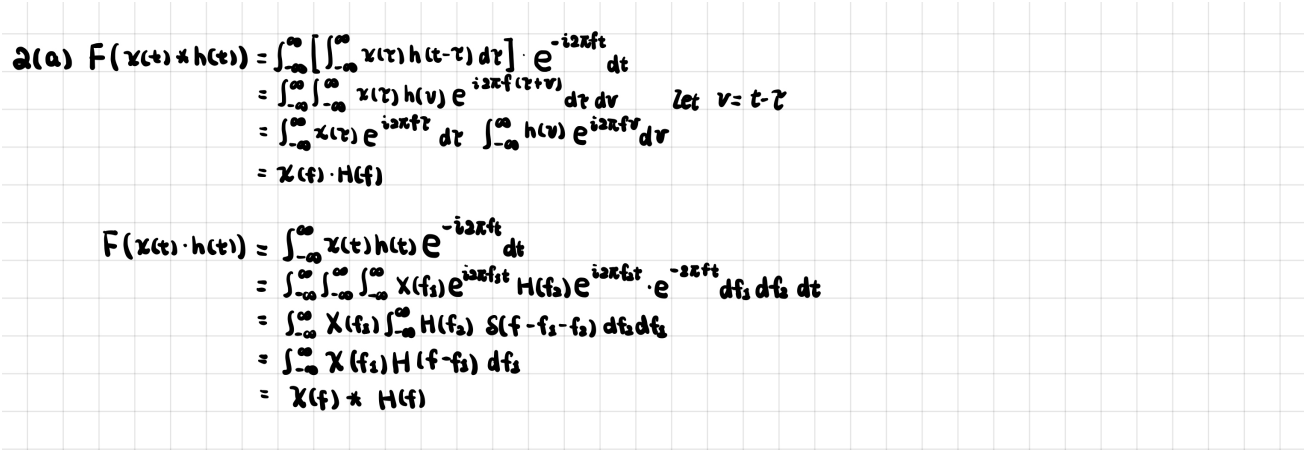
(c) Write a code to numerically compute y(t) and plot y(t). Please use conv.

Please see the solution in 1(b).

Problem 2: Convolution Theorem (10 points)

(a) Proof the convolution theorem and explain the meaning of these relationships in your words.

$$F\{x(t) * h(t)\} = X(f) \cdot H(f)$$
$$F\{x(t) \cdot h(t)\} = X(f) * H(f)$$

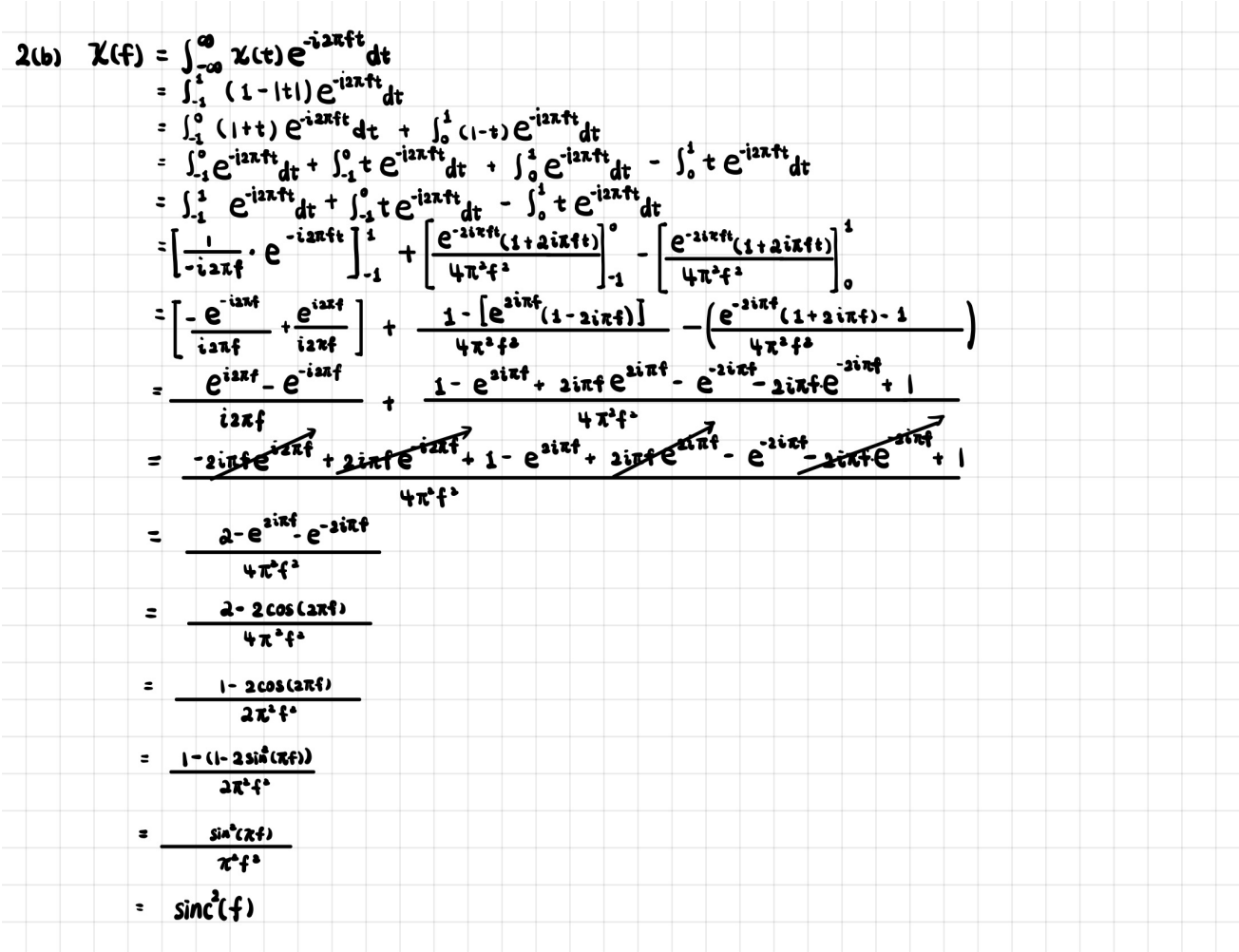


Fourier transform of the convolution of two time signals is the product of their Fourier transforms. Fourier transform of the product of two time signals is the convolution of their Fourier transforms.

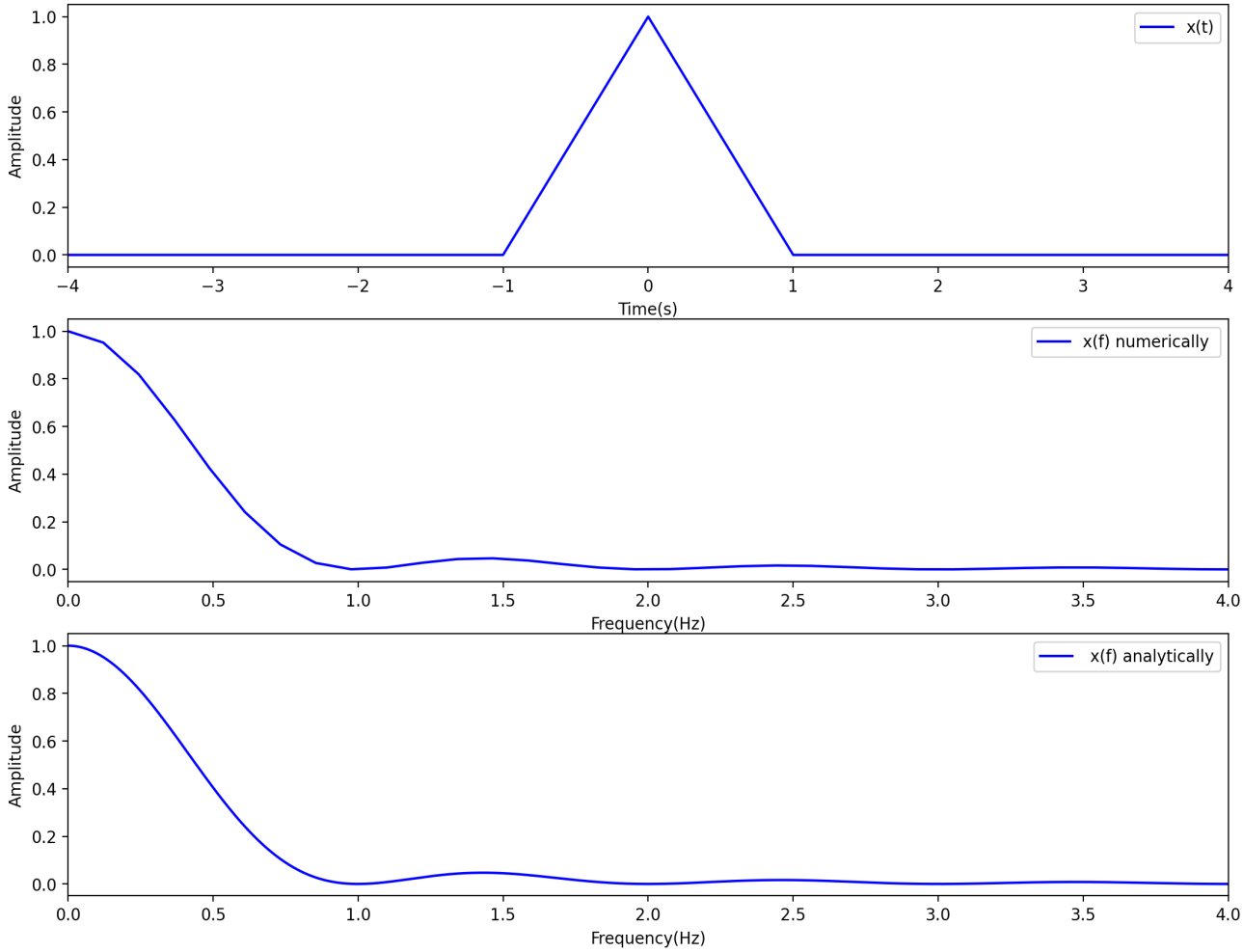
(b) Compute a Fourier transform of the triangular function in both analytic and numeric ways (Note that this function is not periodic):

$$x(t) = \begin{cases} 1 - |t| & |t| < 1 \\ 0 & |t| > 1 \end{cases}$$

Analytical Way



Numerical Way

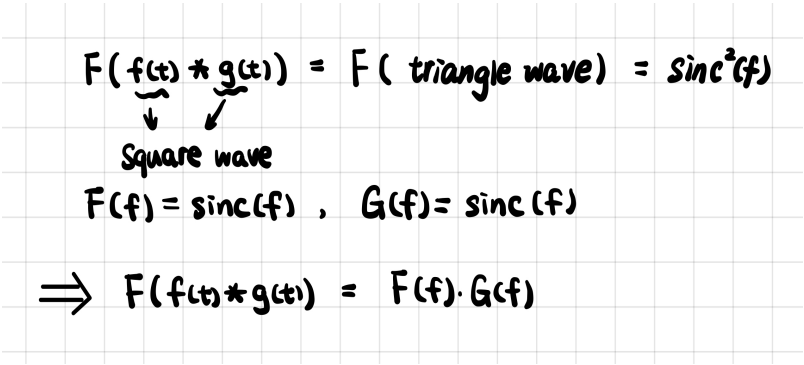



```
fig, ax = plt.subplots(3)
fig.set_dpi(200)
fig.set_size_inches(13, 10, forward=True)

fs = 1000 # sampling rate
t = np.arange(-4,4,1/fs)
dt = 1/fs

# plot of x(t)
x = lambda t: (1-np.abs(t)) * np.heaviside(1-np.abs(t),1-np.abs(t))
ax[0].plot(t, x(t), 'b', label = "x(t)")
ax[0].set_xlim([-4,4])
ax[0].legend(loc = 'upper right')
ax[0].set_xlabel("Time(s)")
ax[0].set_ylabel("Amplitude")

# plot of x(f) numerically
L = len(t)
n = int(2**np.ceil(np.log2(np.abs(L)))) # 'nextpow2' equivalent in python
to improve fft accuracy
x_k = np.fft.fft(x(t),n, axis = -1,norm=None)
f = 1/(n*dt)*np.arange(0,n/2,1) # 'np.arange()' does not include the end
number
# python indexing: end index not incuded --> indexing till n/2-1
ax[1].plot(f,1/fs*np.abs(x_k[0:int(n/2)]), 'b-', label = "x(f) numerically
```

```
")
ax[1].set_xlim([0,4])
ax[1].legend(loc = 'upper right')
ax[1].set_xlabel("Frequency(Hz)")
ax[1].set_ylabel("Amplitude")

# plot of x(f) analytically
f = np.arange(-4,4,1/1000)
x_f = lambda f : np.sinc(f) ** 2
ax[2].plot(f, x_f(f), 'b', label = " x(f) analytically")
ax[2].set_xlim([0,4])
ax[2].legend(loc = 'upper right')
ax[2].set_xlabel("Frequency(Hz)")
ax[2].set_ylabel("Amplitude")
```

(c) Please explain the result in (b) using your answers for Problem 1.

From problem 1, we can find that the convolution of two square waves is a triangular function. In 2(b), we can find that the Fourier transform of a triangular function is sinc^2(f). Also, we know that the fourier transform of a square wave is sinc(f); therefore, this proves that the Fourier transform of the convolution of two time signals is the product of their Fourier transforms.



Problem 3: Discrete Fourier Transform 1 (20 points)

(a) Please explain the difference between the following three notations in the lecture slide:

X(f) is the Fourier transform of a continuous time signal into the frequency domain. Xs(f) is the Fourier transform of a discrete time signal into the frequency domain. X(k) is Xs(f) evaluated at f = k/(N*Δ) Hz (k integer).

(b) What do these two graphs explain in the lecture slides?

The left graph: Xf is the orginal frequency signal, which is not periodic; however, sampling the time signal and applying the Fourier transform will make the signal periodic, like Xsf. The sampling rate is 100 Hz, so the pattern is repeated every fs (100 Hz). No aliasing occurs.

The right graph: The sampling rate is 10 Hz, so the nyquist frequency is 5 Hz. The frequency range is 0 - 10 Hz, which can be higher than the nyquist frequency, so aliasing occurs

(c) What is the meaning of the following relationship in the lecture slide? Please explain it.

Xs(f) is a periodic signal with a period of 1/Δ or fs: Xs(f+r/Δ) = Xs(f) This is becasue Xs(f) is the Fourier transform of the sampled time signal with a sampling rate fs or 1/delta.

(d) What is the meaning of the following relationship in the lecture slide? What issues are introduced by Fourier transform a discrete sequence? Please answer this question using this graph.

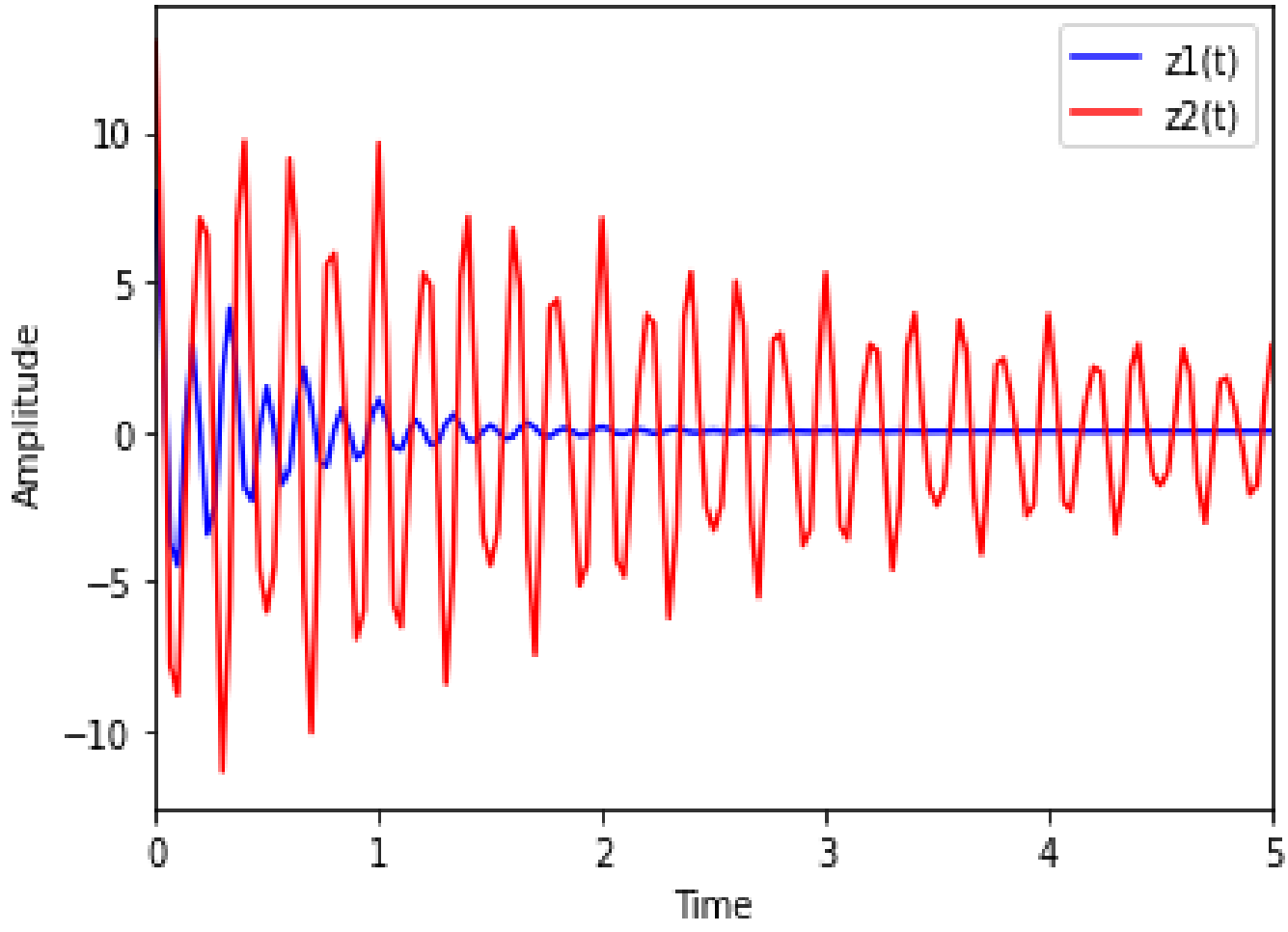
The Fourier Transform of a discrete sampled signal is the sum of the shifted Fourier Transform of a original continuous signal (X(f)) multiplied by (1/Δ). Aliasing can occur when the nyquist frequncy is lower than sampling frequency, resulting in an overlapping region between the adjacent shifted frequency signals.

Problem 4: Discrete Fourier Transform 2 (20 points) - Use FFT

$y_1(t) = e^{-a|t|}(b \cdot cos2\pi f_1 t + c \cdot cos2\pi f_2 t)$ where a = 2, b = 2, c= 6, f_1 = 3, and f_2 = 6

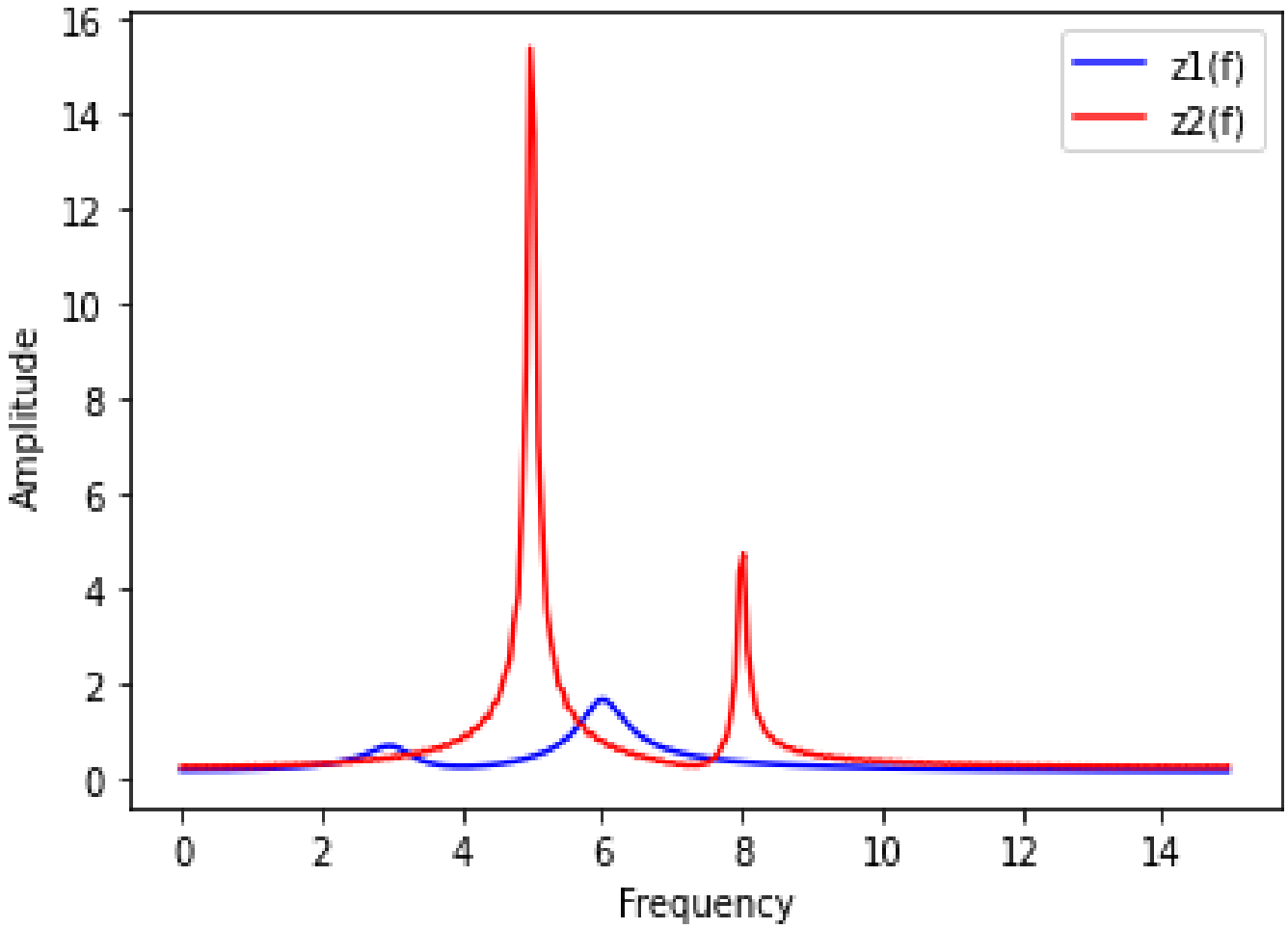
$y_2(t) = e^{-a|t|}(b \cdot cos2\pi f_1 t + c \cdot cos2\pi f_2 t)$ where a = 0.3, b = 10, c= 3, f_1 = 5, and f_2 = 8

(a) z1 and z2 are discrete signals, which are obtained by digitizing (sampling) y1(t) and y2(t) with a sampling rate of 30 Hz and collecting them for 10 seconds, respectively. Please plot z1 and z2 in the time domain (include a proper time axis).



```
time = 10  # sampling time for 10 sec
fs = 30 # sampling rate (Hz)
t = np.arange(0,time,1/fs) # start end step

z1 = lambda t: np.exp(-2*np.abs(t)) * ( 2*np.cos(2*np.pi*3*t) +
6*np.cos(2*np.pi*6*t) )
z2 = lambda t: np.exp(-0.3*np.abs(t)) * ( 10*np.cos(2*np.pi*5*t) +
3*np.cos(2*np.pi*8*t) )

plt.plot(t,z1(t), 'b-', label = "z1(t)")
plt.plot(t,z2(t), 'r-', label = "z2(t)")
plt.xlim([0, 5])
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.legend()
```

(b) Perform the discrete Fourier transform of z1 and z2, and plot your graphs in the frequency domain (include a proper frequency axis). Plot only positive frequency signals with absolute values.




```
time = 10  # sampling time for 10 sec
fs = 30 # sampling rate (Hz)
t = np.arange(0,time,1/fs) # start end step
```

```
dt = 1/fs
L = len(t)
n = np.int(2**np.ceil(np.log2(np.abs(L)))) # 'nextpow2' equivalent in
python to improve fft accuracy

z1 = lambda t: np.exp(-2*np.abs(t)) * ( 2*np.cos(2*np.pi*3*t) +
6*np.cos(2*np.pi*6*t) )
z2 = lambda t: np.exp(-0.3*np.abs(t)) * ( 10*np.cos(2*np.pi*5*t) +
3*np.cos(2*np.pi*8*t) )

# 'np.fft.fft' function
z1_k = np.fft.fft(z1(t),n, axis = -1,norm=None)
z2_k = np.fft.fft(z2(t),n, axis = -1,norm=None)
f = 1/(n*dt)*np.arange(0,n/2,1) # 'np.arange()' does not include the end
number

# python indexing: end index not incuded --> indexing till n/2-1
plt.plot(f,1/fs*np.abs(z1_k[0:int(n/2)]), 'b-', label = "z1(f)")
plt.plot(f,1/fs*np.abs(z2_k[0:int(n/2)]), 'r-', label = "z2(f)")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.legend()
```

(c) Please compare the shape of the frequency curves of z1 and z2. Which frequency curve is thinner (more narrow)? For example, compare the frequency curve at f1 in both graphs. Which one is thinner? Please explain your answer. What makes the difference?

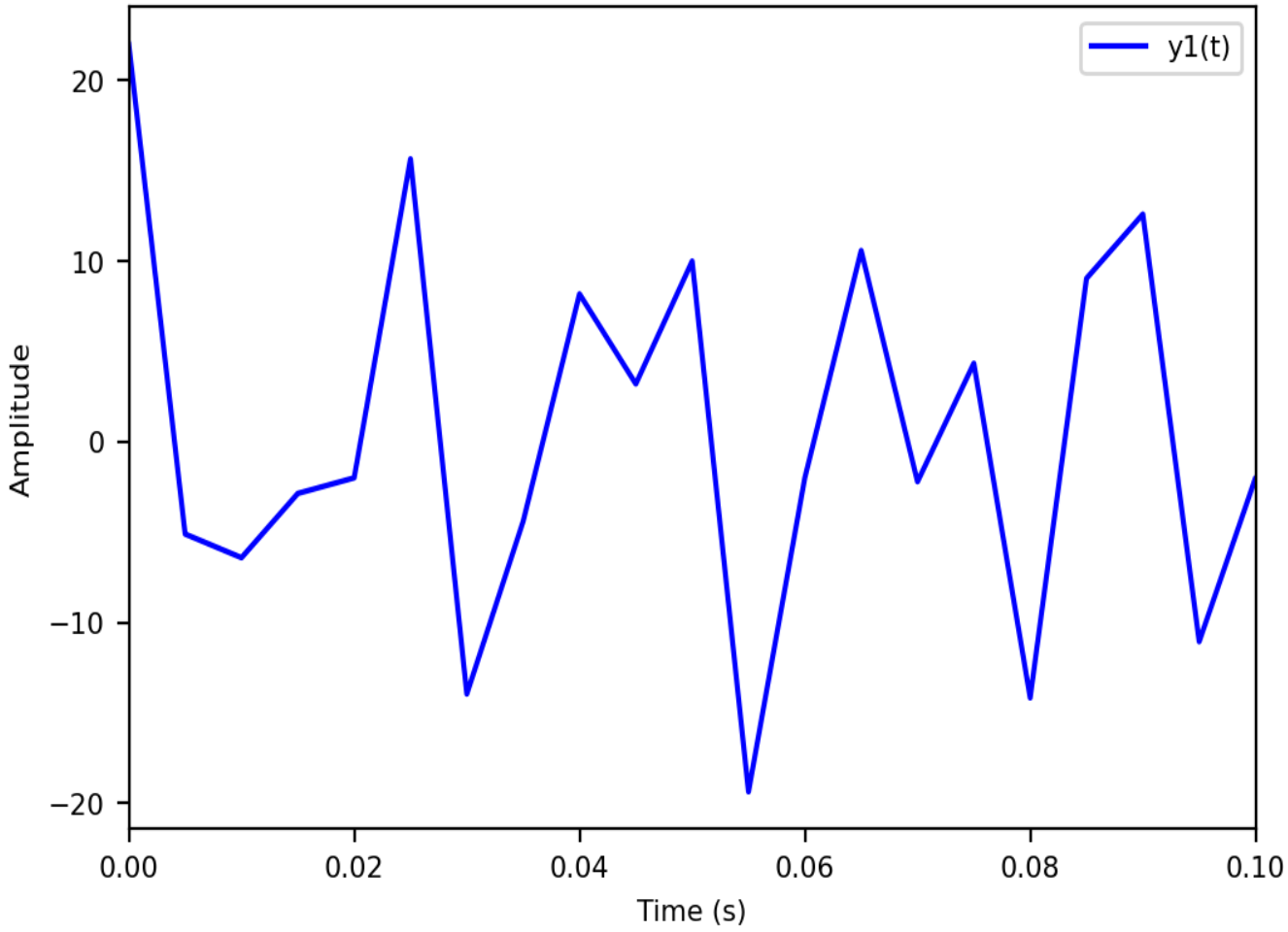
z2(f) is narrower becasue z2(t) is wider in the time domain because it has a smaller a (0.3).For example, at f1, z2(f1) is greater than z1(f1). This due to the inverse spreading property of the fourier transform.

Problem 5: Discrete Fourier Transform 3 (30 points) - Use FFT

$$y(t) = A1cos2\pi(25)t + A2cos2\pi(45)t + A3cos2\pi(80)t$$

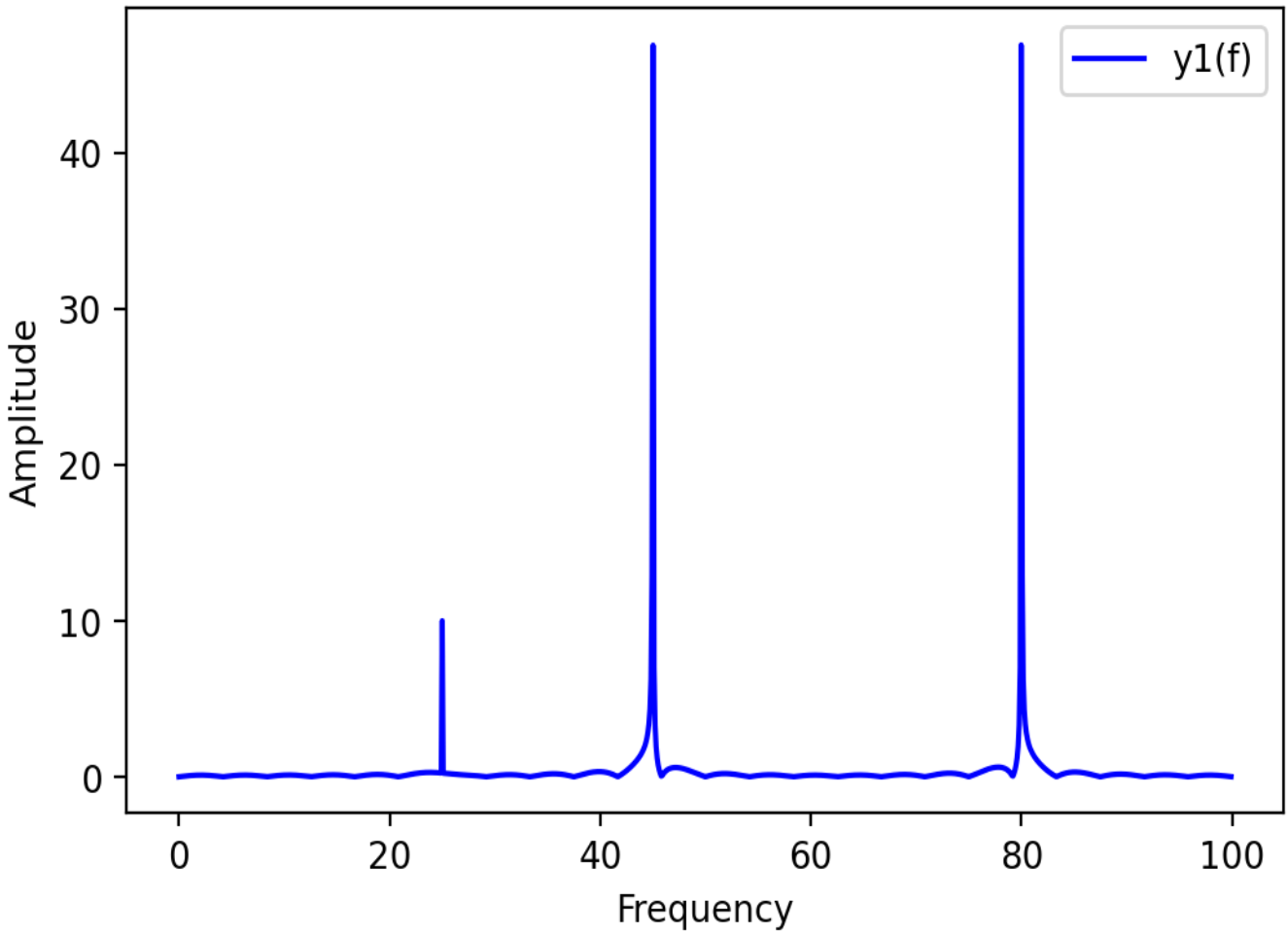
where A1 = 2, A2 = 10, and A3 = 10.

(a) y1 is a discrete signal, which is obtained by digitizing y(t) with a sampling rate of 200 Hz for 10 seconds. Please plot y1 in the time domain (include a proper time axis). Please connect the sampled points.




```
fig, ax = plt.subplots()
fig.set_dpi(200)

A1 =2
A2 = 10
A3 = 10
fs = 200 # sampling rate
t = np.arange(0,10,1/fs) # 10 seconds
y1 = lambda t: A1*np.cos(2*np.pi*25*t) + A2*np.cos(2*np.pi*45*t) +
A3*np.cos(2*np.pi*80*t)
ax.plot(t, y1(t), 'b', label = "y1(t)")
plt.xlim([0,0.1])
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.legend(loc = 'upper right')
```

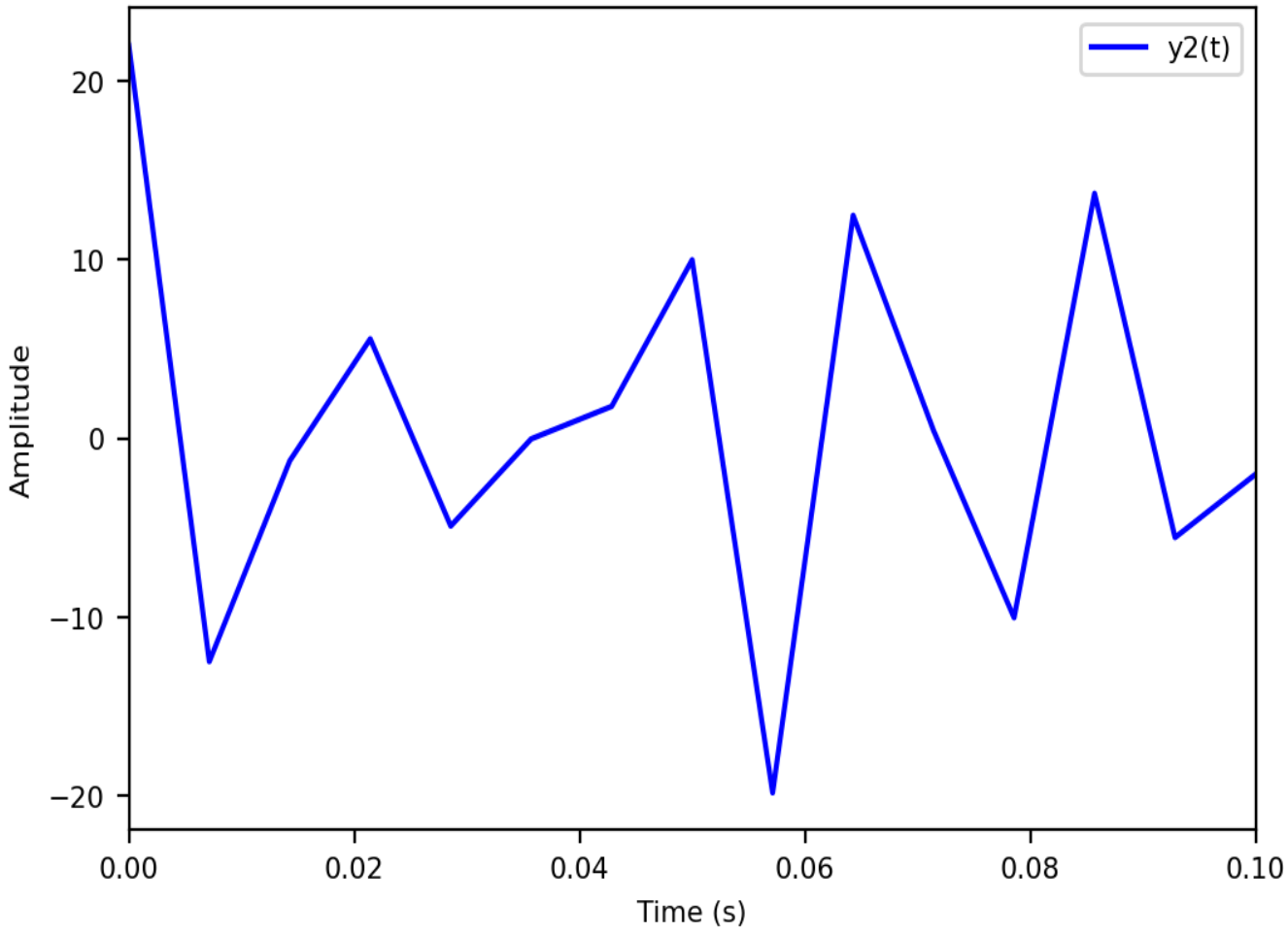
(b) Perform the discrete Fourier transform of y1 and plot your graph in the frequency domain (include a proper frequency axis). Plot only a positive frequency signal. Can we measure all three frequencies?




```
fig, ax = plt.subplots()
fig.set_dpi(200)

A1 =2
A2 = 10
A3 = 10
fs = 200 # sampling rate
t = np.arange(0,10,1/fs) # 10 seconds
dt = 1/fs
L = len(t)
n = np.int(2**np.ceil(np.log2(np.abs(L)))) # 'nextpow2' equivalent in
python to improve fft accuracy

# 'np.fft.fft' function
y1 = lambda t: A1*np.cos(2*np.pi*25*t) + A2*np.cos(2*np.pi*45*t) +
A3*np.cos(2*np.pi*80*t)
y1_k = np.fft.fft(y1(t),n, axis = -1,norm=None)
f = 1/(n*dt)*np.arange(0,n/2,1) # 'np.arange()' does not include the end
number

# python indexing: end index not incuded --> indexing till n/2-1
plt.plot(f,1/fs*np.abs(y1_k[0:int(n/2)]), 'b-', label = "y1(f)")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.legend(loc = 'upper right')
```

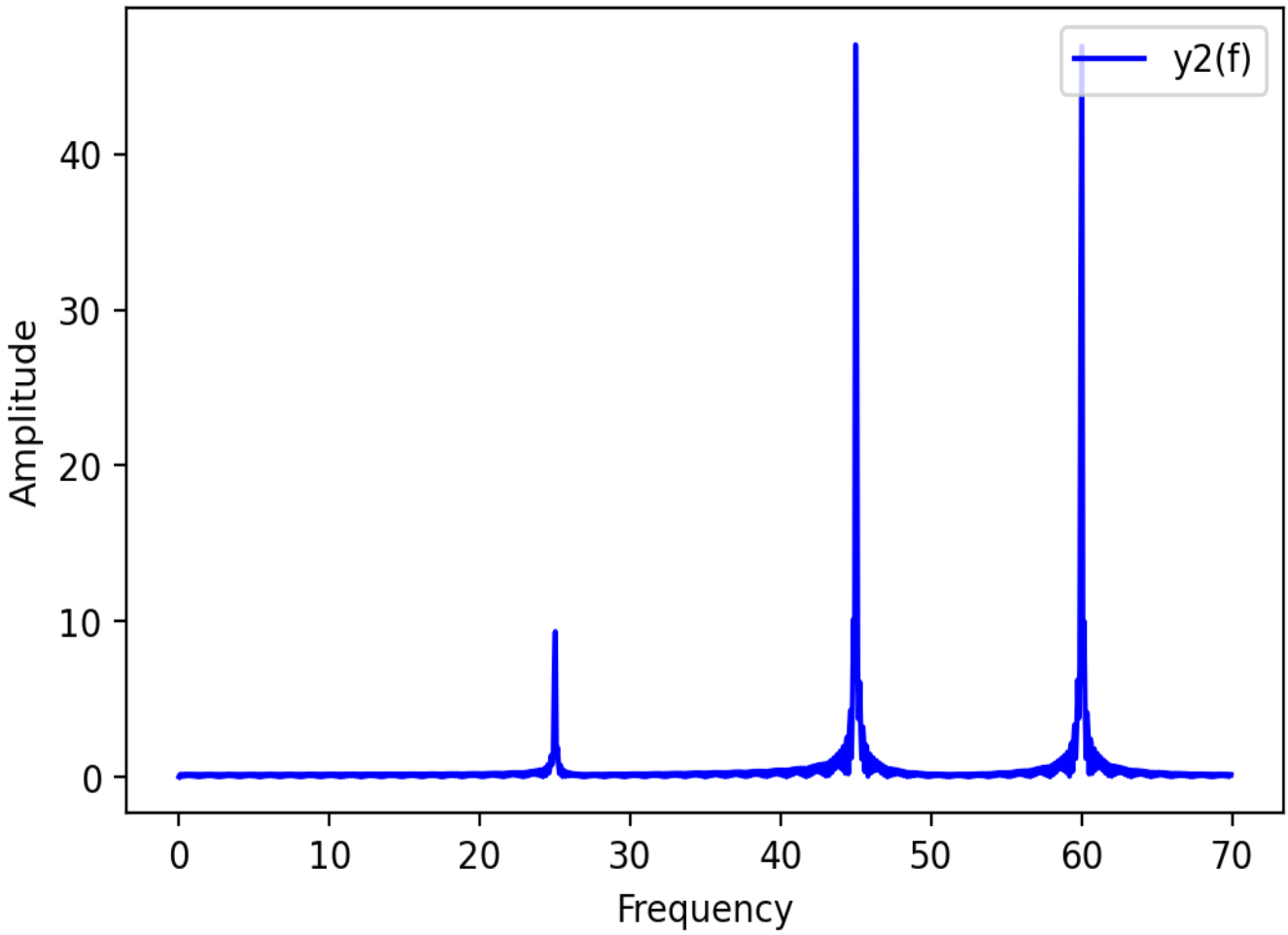
Yes, we can measure all three frequencies: 25 Hz, 45 Hz, and 80 Hz. The sampling rate is 200 Hz, so the nyquist frequency is 100 Hz, which is greater than all three frequencies.

(c) y2 is a discrete signal, which is obtained by digitizing y(t) with a sampling rate of 140 Hz for 10 seconds. Please plot y2 in the time domain (include a proper time axis). Please connect the sampled points.




```
fig, ax = plt.subplots()
fig.set_dpi(200)

A1 =2
A2 = 10
A3 = 10
fs = 140 # sampling rate
t = np.arange(0,10,1/fs) # 10 seconds
y2 = lambda t: A1*np.cos(2*np.pi*25*t) + A2*np.cos(2*np.pi*45*t) +
A3*np.cos(2*np.pi*80*t)
ax.plot(t, y2(t), 'b', label = "y2(t)")
plt.xlim([0,0.1])
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.legend(loc = 'upper right')
```

(d) Perform the discrete Fourier transform of y2 and plot your graph in the frequency domain (include a proper frequency axis). Plot only a positive frequency signal. Can we measure all three frequencies?




```
fig, ax = plt.subplots()
fig.set_dpi(200)

A1 =2
A2 = 10
A3 = 10
fs = 140 # sampling rate
t = np.arange(0,10,1/fs) # 10 seconds
dt = 1/fs
L = len(t)
n = np.int(2**np.ceil(np.log2(np.abs(L)))) # 'nextpow2' equivalent in
python to improve fft accuracy

# 'np.fft.fft' function
y2 = lambda t: A1*np.cos(2*np.pi*25*t) + A2*np.cos(2*np.pi*45*t) +
A3*np.cos(2*np.pi*80*t)
y2_k = np.fft.fft(y2(t),n, axis = -1,norm=None)
f = 1/(n*dt)*np.arange(0,n/2,1) # 'np.arange()' does not include the end
number

# python indexing: end index not incuded --> indexing till n/2-1
```

```
plt.plot(f,1/fs*np.abs(y2_k[0:int(n/2)]), 'b-', label = "y2(f)")
plt.xlabel('Frequency')
plt.ylabel('Amplitude')
plt.legend()
plt.legend(loc = 'upper right')
```

No, we cannot measure all three frequencies: 25 Hz, 45 Hz, and 80 Hz. The sampling rate is 140 Hz, so the nyquist frequency is 70 Hz, which is smaller than the 80 Hz frequency. In this case, aliasing occurs.

(e) If you digitize a longer-duration signal (let's say 100 seconds) with a sampling rate of 140 Hz, can you measure and extract all three frequencies contained in the original signal, y(t)? Please explain your answer.

No, you cannot measure and extract all three frequncies contained in the original signal. The sampling rate remians the same as 140 Hz, so the nyquist frequency is 70 Hz, which is smaller than the 80 Hz frequency. In this case, aliasing occurs.

(f) If you digitize the signal with a sampling rate of 150 Hz for 20 seconds, can you measure and extract all frequencies contained in the original signal, y(t)? please explain your answer.

No, you cannot measure and extract all three frequncies contained in the original signal. The sampling rate remians the same as 150 Hz, so the nyquist frequency is 75 Hz, which is smaller than the 80 Hz frequency. In this case, aliasing occurs.

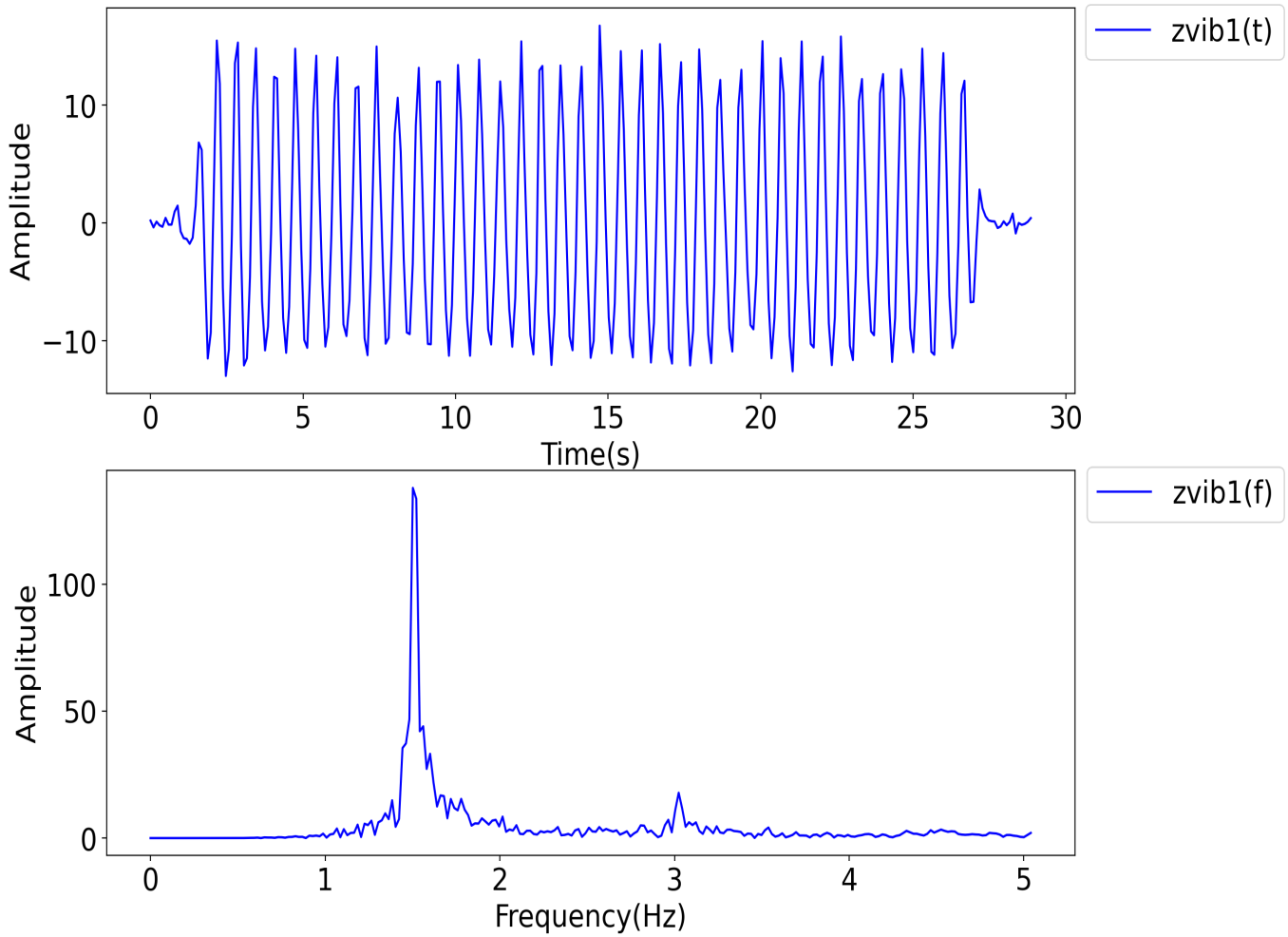
(g) If you digitize the signal with a sampling rate of 161 Hz for 20 seconds, can you measure and extract all frequencies contained in the original signal, y(t)? please explain your answer.

Yes, you can measure and extract all three frequncies contained in the original signal.The sampling rate remians the same as 161 Hz, so the nyquist frequency is 80.5 Hz, which is higher than the 80 Hz frequency.

Problem 6: Frequency Analysis (10 points)

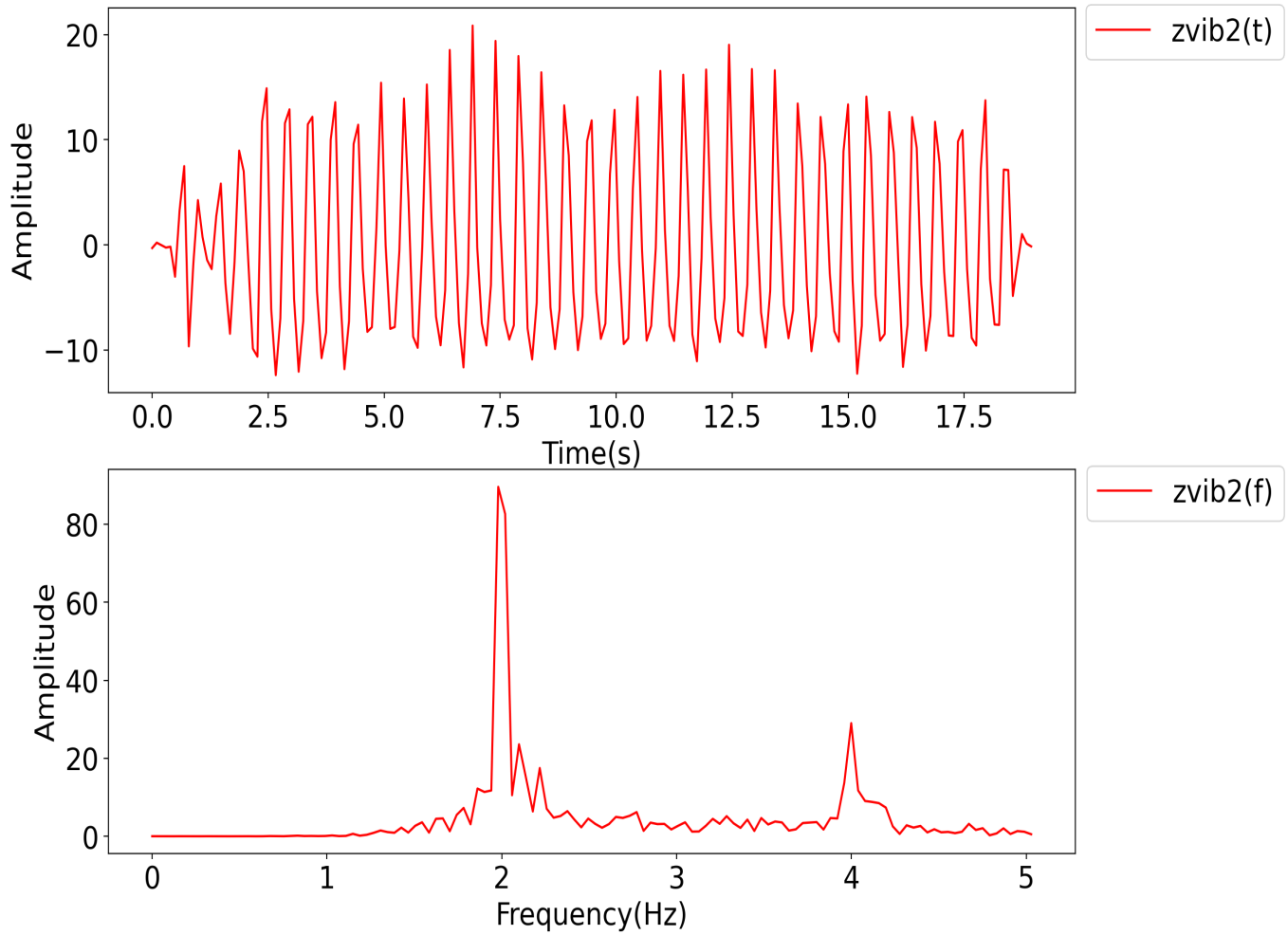
Two sinusoidal accelerations are measured using an accelerometer in a smartphone. Each of the waves is stored at data1.mat and data2.mat.

(a) Load data1.mat and plot the acceleration signal in a z direction (zvib). What is the main frequency of this wave? The sampling frequency is 10.1355 Hz.



It can be observed from the plot above that the main frequency is approximately 1.5 Hz.

(b) Load data2.mat and plot the acceleration signal in a z direction (zvib). What is the main frequency of this wave? The sampling frequency is 10.1192 Hz.



It can be observed from the plot above that the main frequency is approximately 2 Hz.