Experiment No. 3 Date:15/1022

Aim: To simulate Frequency Division Multiplexing (FDM) & Demultiplexing using Python.

Objectives:

1. To develop a simulation model for time-domain representation of signals.

2. To plot & observe the FDM spectrum for two signals.

3. To develop a simulation model for frequency response of the ideal discrete Band Pass Filter.

4. To plot & observe the demultiplexed signal spectrum and corresponding waveforms.

Resources/Specifications:

1. Desktop/Laptop System

2. Python 3 and necessary libraries

Algorithm:

1. Define the amplitude and frequency parameters for the two sinusoidal signals to be multiplexed. Also define sampling frequency and create the array for time indices.

2. Add the two signals to create the FDM signal.

3. Plot and observe the time-domain waveforms for the two sinusoidal signals.

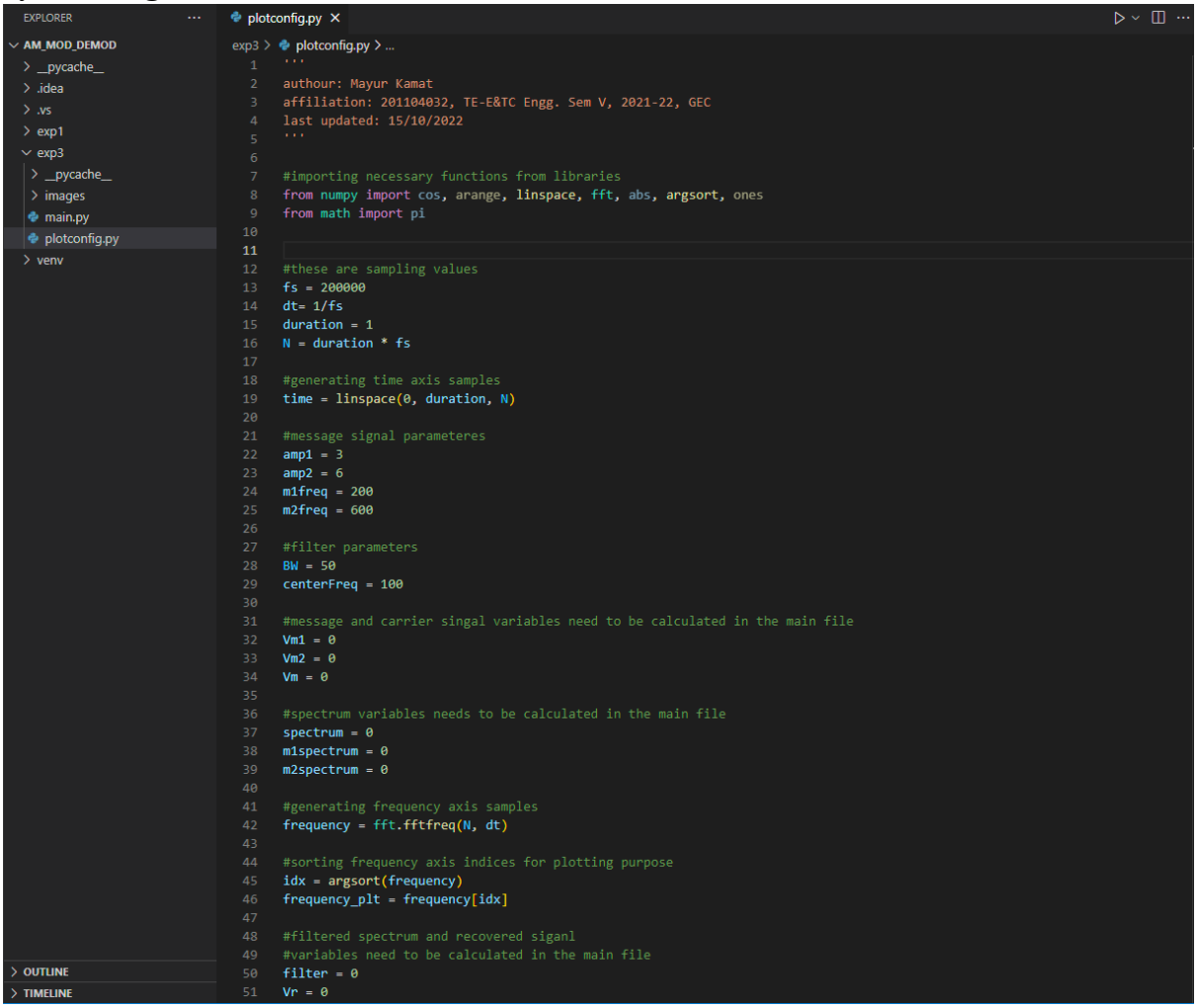
4. Implement the Discrete Fourier Transform (DFT) operation on signals and FDM signal.

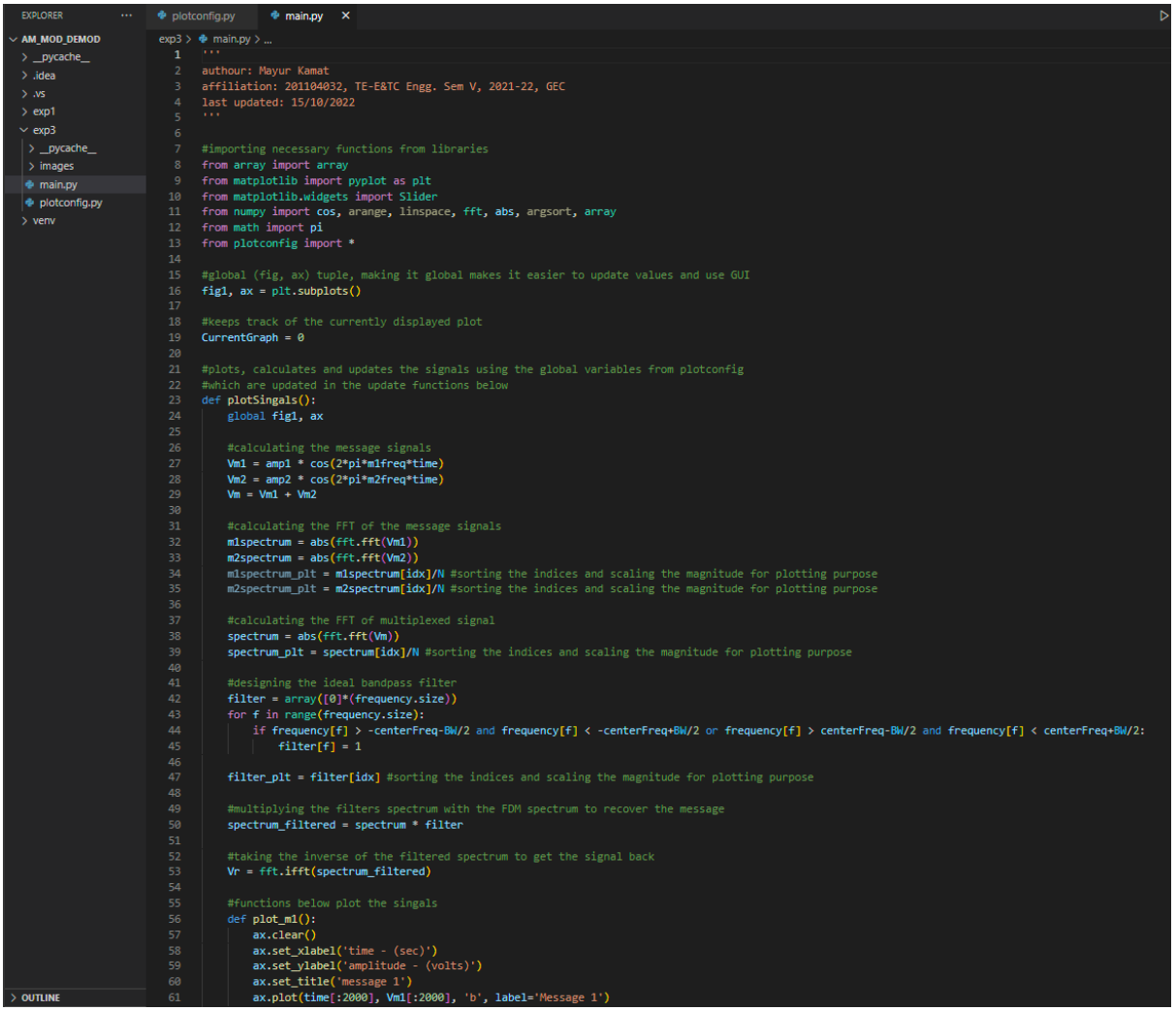
5. Shift, scale, plot and observe the spectrum by defining appropriate frequency indices.

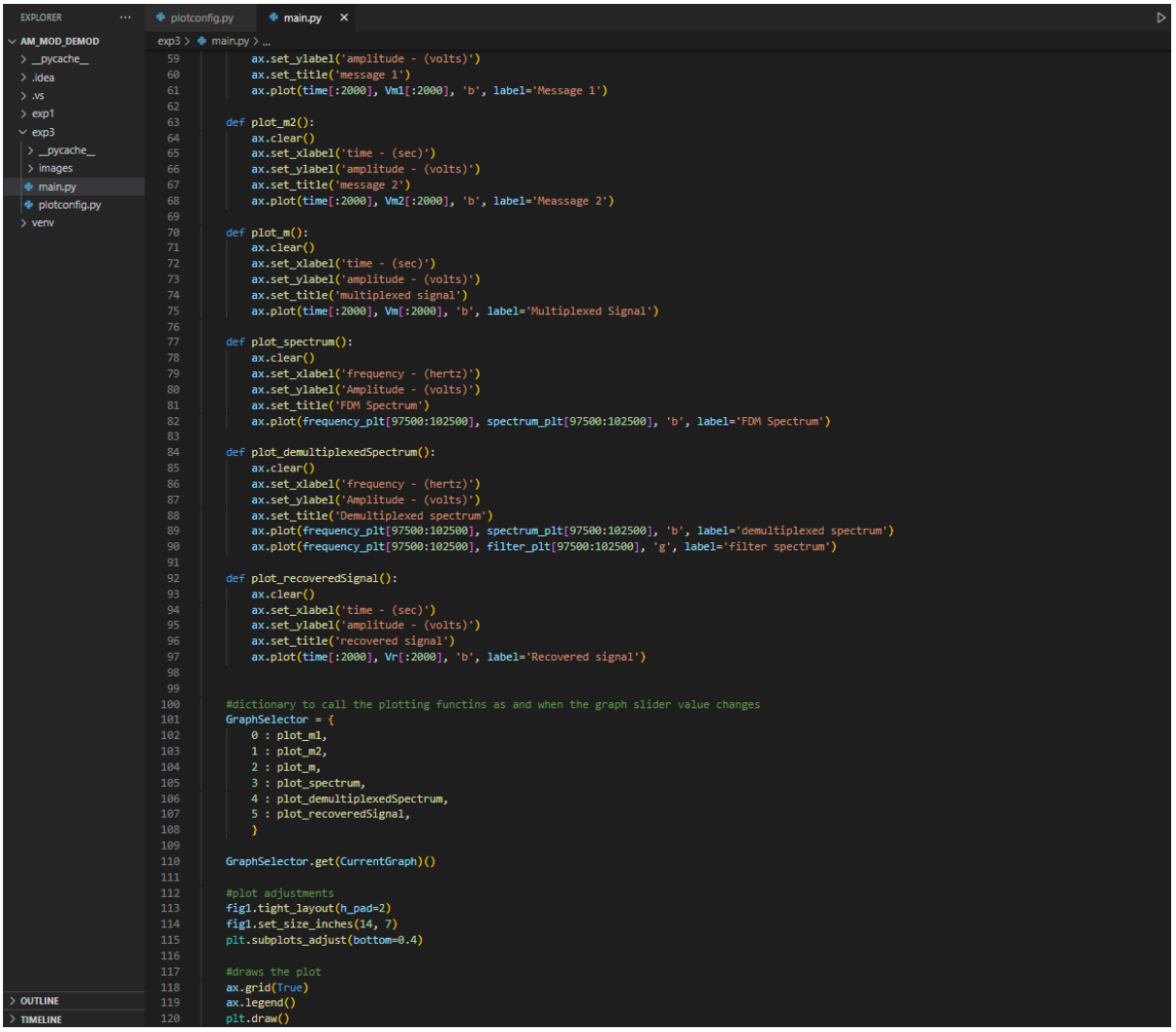
6. Define the ideal Discrete Band Pass Filter (BPF) response, and extract the original signal spectrum by multiplying the BPF response with the DFT of the FDM signals separately.

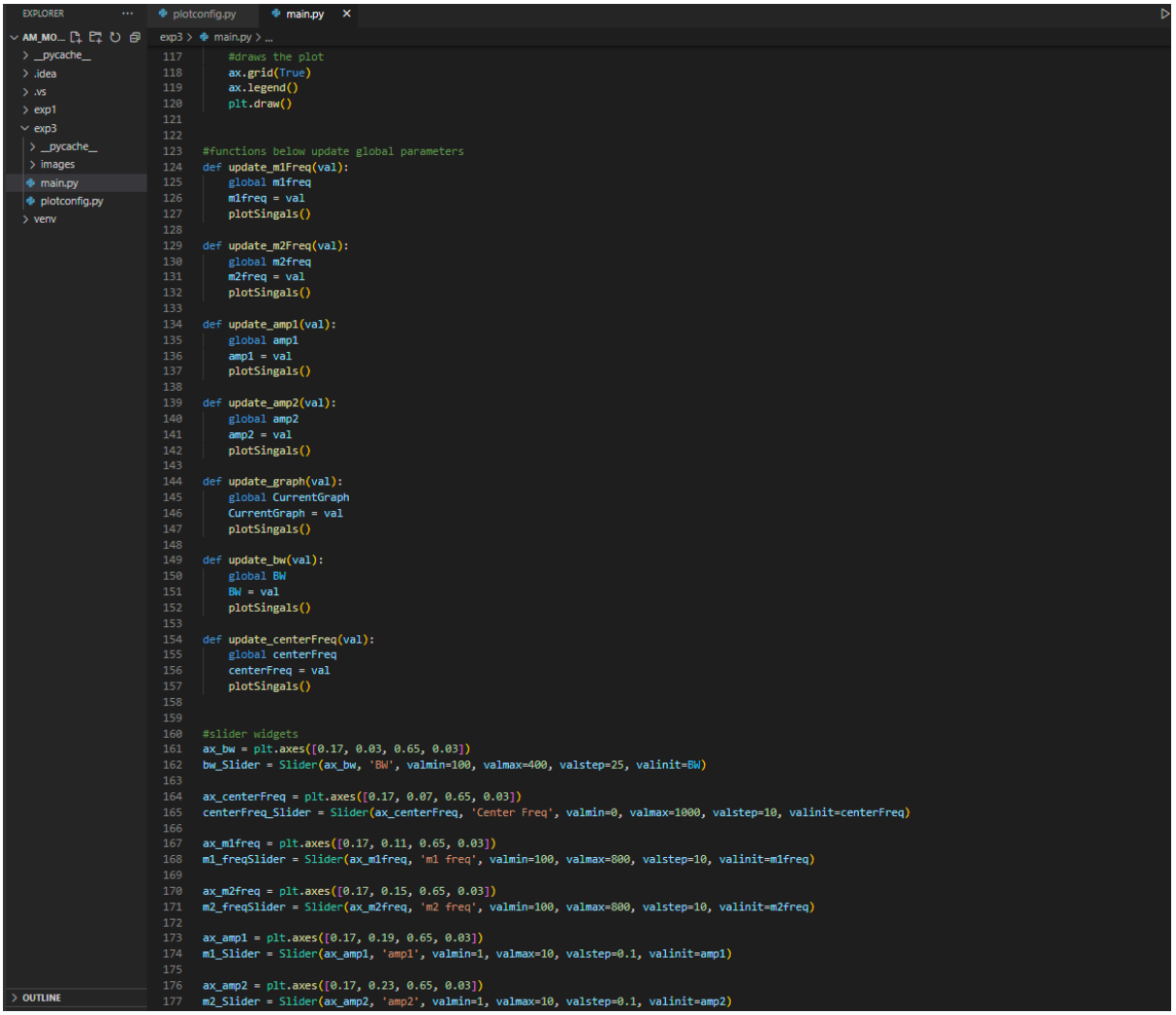
7. Find the inverse transform of the spectrum extracted in Step 6 and plot the recovered signal waveform.

PythonProgram:

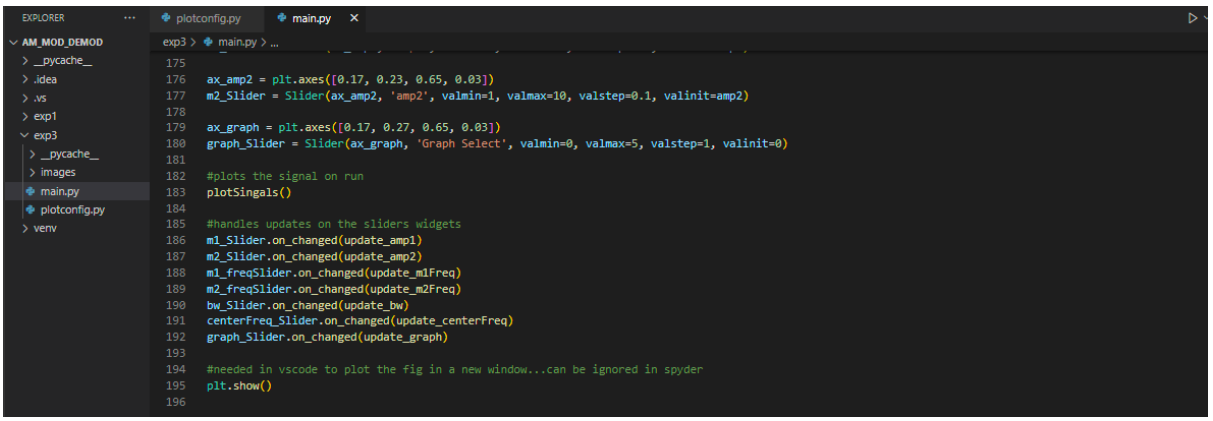



EXPLORER
plotconfig.py
main.py
AM_MOD_DEMOD
__pycache__
.idea
.vs
exp1
exp3
__pycache__
images
main.py
plotconfig.py
venv
OUTLINE
exp3 > main.py > ...
'''
authour: Mayur Kamat
affiliation: 201104032, TE-E&TC Engg. Sem V, 2021-22, GEC
last updated: 15/10/2022
'''
#importing necessary functions from libraries
from array import array
from matplotlib import pyplot as plt
from matplotlib.widgets import Slider
from numpy import cos, arange, linspace, fft, abs, argsort, array
from math import pi
from plotconfig import *
#global (fig, ax) tuple, making it global makes it easier to update values and use GUI
fig1, ax = plt.subplots()
#keeps track of the currently displayed plot
CurrentGraph = 0
#plots, calculates and updates the signals using the global variables from plotconfig
#which are updated in the update functions below
def plotSingals():
global fig1, ax
#calculating the message signals
Vm1 = amp1 * cos(2*pi*m1freq*time)
Vm2 = amp2 * cos(2*pi*m2freq*time)
Vm = Vm1 + Vm2
#calculating the FFT of the message signals
m1spectrum = abs(fft.fft(Vm1))
m2spectrum = abs(fft.fft(Vm2))
m1spectrum_plt = m1spectrum[idx]/N #sorting the indices and scaling the magnitude for plotting purpose
m2spectrum_plt = m2spectrum[idx]/N #sorting the indices and scaling the magnitude for plotting purpose
#calculating the FFT of multiplexed signal
spectrum = abs(fft.fft(Vm))
spectrum_plt = spectrum[idx]/N #sorting the indices and scaling the magnitude for plotting purpose
#designing the ideal bandpass filter
filter = array([0]*(frequency.size))
for f in range(frequency.size):
if frequency[f] > -centerFreq-BW/2 and frequency[f] < -centerFreq+BW/2 or frequency[f] > centerFreq-BW/2 and frequency[f] < centerFreq+BW/2:
filter[f] = 1
filter_plt = filter[idx] #sorting the indices and scaling the magnitude for plotting purpose
#multiplying the filters spectrum with the FDM spectrum to recover the message
spectrum_filtered = spectrum * filter
#taking the inverse of the filtered spectrum to get the signal back
Vr = fft.ifft(spectrum_filtered)
#functions below plot the singals
def plot_m1():
ax.clear()
ax.set_xlabel('time - (sec)')
ax.set_ylabel('amplitude - (volts)')
ax.set_title('message 1')
ax.plot(time[:2000], Vm1[:2000], 'b', label='Message 1')


EXPLORER
plotconfig.py
main.py
exp3 > main.py > ...
AM_MOD_DEMOD
__pycache__
.idea
.vs
exp1
exp3
__pycache__
images
main.py
plotconfig.py
venv
OUTLINE
TIMELINE
59 ax.set_ylabel('amplitude - (volts)')
60 ax.set_title('message 1')
61 ax.plot(time[:2000], Vm1[:2000], 'b', label='Message 1')
62
63 def plot_m2():
64 ax.clear()
65 ax.set_xlabel('time - (sec)')
66 ax.set_ylabel('amplitude - (volts)')
67 ax.set_title('message 2')
68 ax.plot(time[:2000], Vm2[:2000], 'b', label='Meassage 2')
69
70 def plot_m():
71 ax.clear()
72 ax.set_xlabel('time - (sec)')
73 ax.set_ylabel('amplitude - (volts)')
74 ax.set_title('multiplexed signal')
75 ax.plot(time[:2000], Vm[:2000], 'b', label='Multiplexed Signal')
76
77 def plot_spectrum():
78 ax.clear()
79 ax.set_xlabel('frequency - (hertz)')
80 ax.set_ylabel('Amplitude - (volts)')
81 ax.set_title('FDM Spectrum')
82 ax.plot(frequency_plt[97500:102500], spectrum_plt[97500:102500], 'b', label='FDM Spectrum')
83
84 def plot_demultiplexedSpectrum():
85 ax.clear()
86 ax.set_xlabel('frequency - (hertz)')
87 ax.set_ylabel('Amplitude - (volts)')
88 ax.set_title('Demultiplexed spectrum')
89 ax.plot(frequency_plt[97500:102500], spectrum_plt[97500:102500], 'b', label='demultiplexed spectrum')
90 ax.plot(frequency_plt[97500:102500], filter_plt[97500:102500], 'g', label='filter spectrum')
91
92 def plot_recoveredSignal():
93 ax.clear()
94 ax.set_xlabel('time - (sec)')
95 ax.set_ylabel('amplitude - (volts)')
96 ax.set_title('recovered signal')
97 ax.plot(time[:2000], Vr[:2000], 'b', label='Recovered signal')
98
99
100 #dictionary to call the plotting functins as and when the graph slider value changes
101 GraphSelector = {
102 0 : plot_m1,
103 1 : plot_m2,
104 2 : plot_m,
105 3 : plot_spectrum,
106 4 : plot_demultiplexedSpectrum,
107 5 : plot_recoveredSignal,
108 }
109
110 GraphSelector.get(CurrentGraph)()
111
112 #plot adjustments
113 fig1.tight_layout(h_pad=2)
114 fig1.set_size_inches(14, 7)
115 plt.subplots_adjust(bottom=0.4)
116
117 #draws the plot
118 ax.grid(True)
119 ax.legend()
120 plt.draw()



```
    #draws the plot
    ax.grid(True)
    ax.legend()
    plt.draw()


#functions below update global parameters
def update_m1Freq(val):
    global m1freq
    m1freq = val
    plotSingals()

def update_m2Freq(val):
    global m2freq
    m2freq = val
    plotSingals()

def update_amp1(val):
    global amp1
    amp1 = val
    plotSingals()

def update_amp2(val):
    global amp2
    amp2 = val
    plotSingals()

def update_graph(val):
    global CurrentGraph
    CurrentGraph = val
    plotSingals()

def update_bw(val):
    global BW
    BW = val
    plotSingals()

def update_centerFreq(val):
    global centerFreq
    centerFreq = val
    plotSingals()


#slider widgets
ax_bw = plt.axes([0.17, 0.03, 0.65, 0.03])
bw_Slider = Slider(ax_bw, 'BW', valmin=100, valmax=400, valstep=25, valinit=BW)

ax_centerFreq = plt.axes([0.17, 0.07, 0.65, 0.03])
centerFreq_Slider = Slider(ax_centerFreq, 'Center Freq', valmin=0, valmax=1000, valstep=10, valinit=centerFreq)

ax_m1freq = plt.axes([0.17, 0.11, 0.65, 0.03])
m1_freqSlider = Slider(ax_m1freq, 'm1 freq', valmin=100, valmax=800, valstep=10, valinit=m1freq)

ax_m2freq = plt.axes([0.17, 0.15, 0.65, 0.03])
m2_freqSlider = Slider(ax_m2freq, 'm2 freq', valmin=100, valmax=800, valstep=10, valinit=m2freq)

ax_amp1 = plt.axes([0.17, 0.19, 0.65, 0.03])
m1_Slider = Slider(ax_amp1, 'amp1', valmin=1, valmax=10, valstep=0.1, valinit=amp1)

ax_amp2 = plt.axes([0.17, 0.23, 0.65, 0.03])
m2_Slider = Slider(ax_amp2, 'amp2', valmin=1, valmax=10, valstep=0.1, valinit=amp2)
```



```

ax_amp2 = plt.axes([0.17, 0.23, 0.65, 0.03])
m2_Slider = Slider(ax_amp2, 'amp2', valmin=1, valmax=10, valstep=0.1, valinit=amp2)

ax_graph = plt.axes([0.17, 0.27, 0.65, 0.03])
graph_Slider = Slider(ax_graph, 'Graph Select', valmin=0, valmax=5, valstep=1, valinit=0)

#plots the signal on run
plotSingals()

#handles updates on the sliders widgets
m1_Slider.on_changed(update_amp1)
m2_Slider.on_changed(update_amp2)
m1_freqSlider.on_changed(update_m1Freq)
m2_freqSlider.on_changed(update_m2Freq)
bw_Slider.on_changed(update_bw)
centerFreq_Slider.on_changed(update_centerFreq)
graph_Slider.on_changed(update_graph)

#needed in vscode to plot the fig in a new window...can be ignored in spyder
plt.show()

```

Plots:

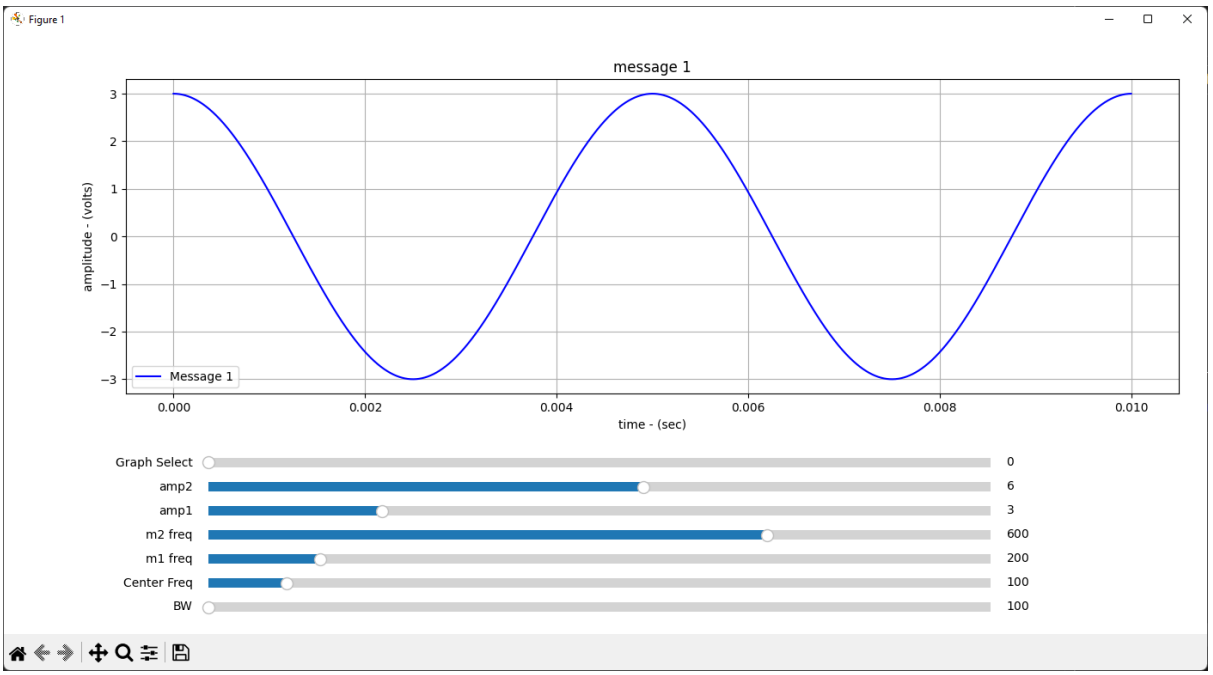


Figure 1
message 1
amplitude - (volts)
time - (sec)
Message 1
Graph Select 0
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 100
BW 100

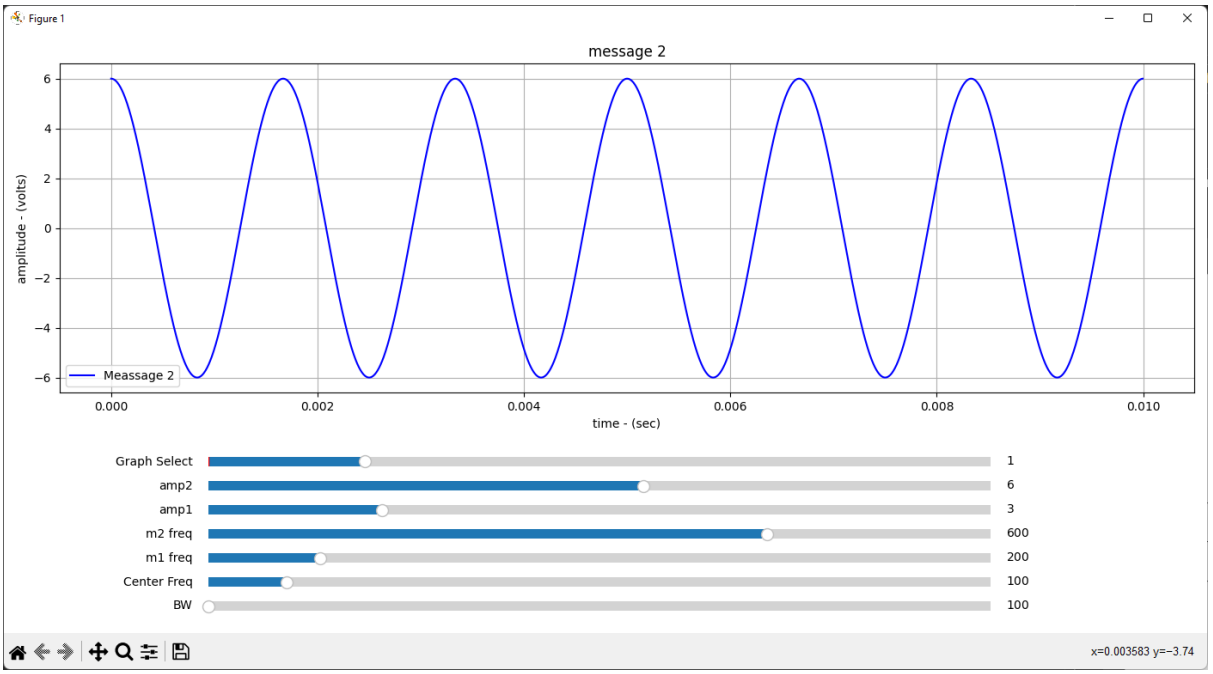


Figure 1
message 2
amplitude - (volts)
time - (sec)
Meassage 2
Graph Select 1
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 100
BW 100
x=0.003583 y=−3.74

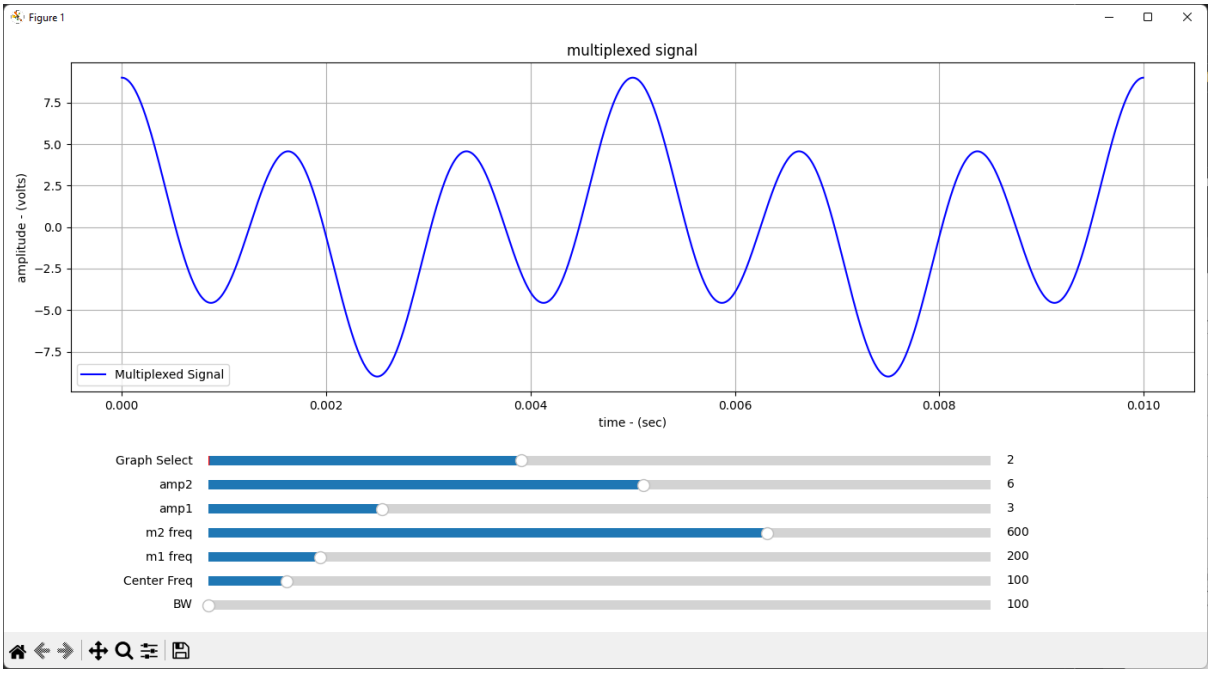


Figure 1
multiplexed signal
amplitude - (volts)
time - (sec)
Multiplexed Signal
Graph Select 2
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 100
BW 100

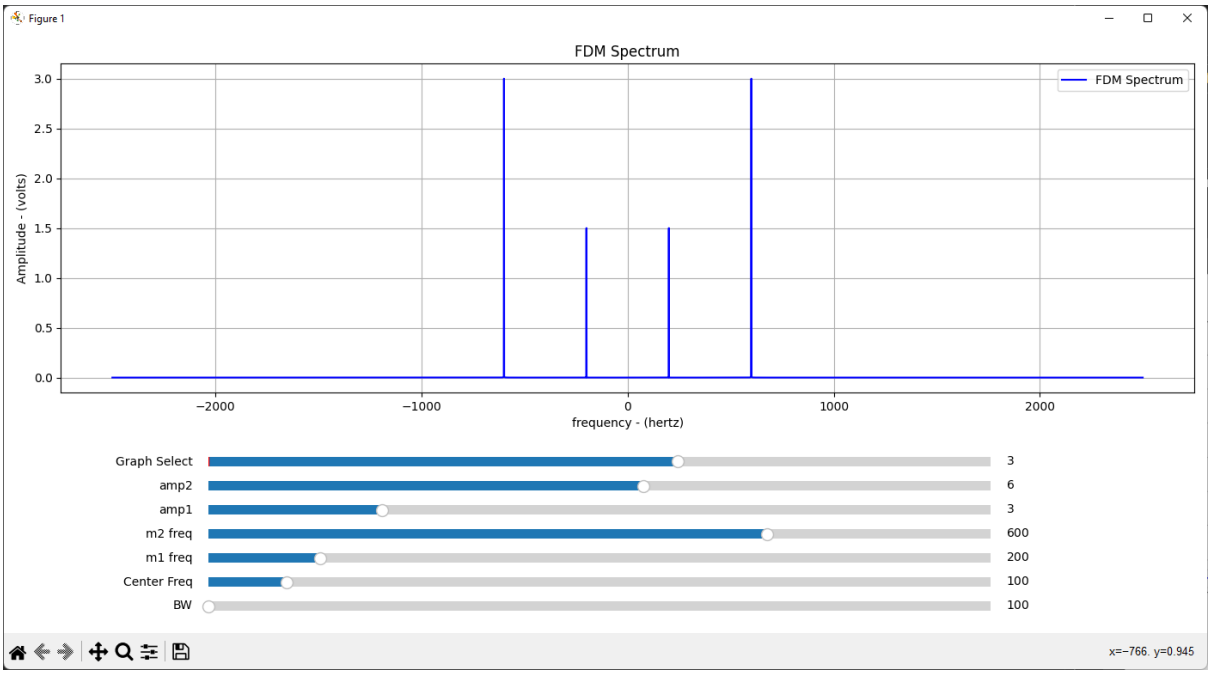


Figure 1
FDM Spectrum
Amplitude - (volts)
frequency - (hertz)
FDM Spectrum
Graph Select 3
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 100
BW 100
x=−766. y=0.945

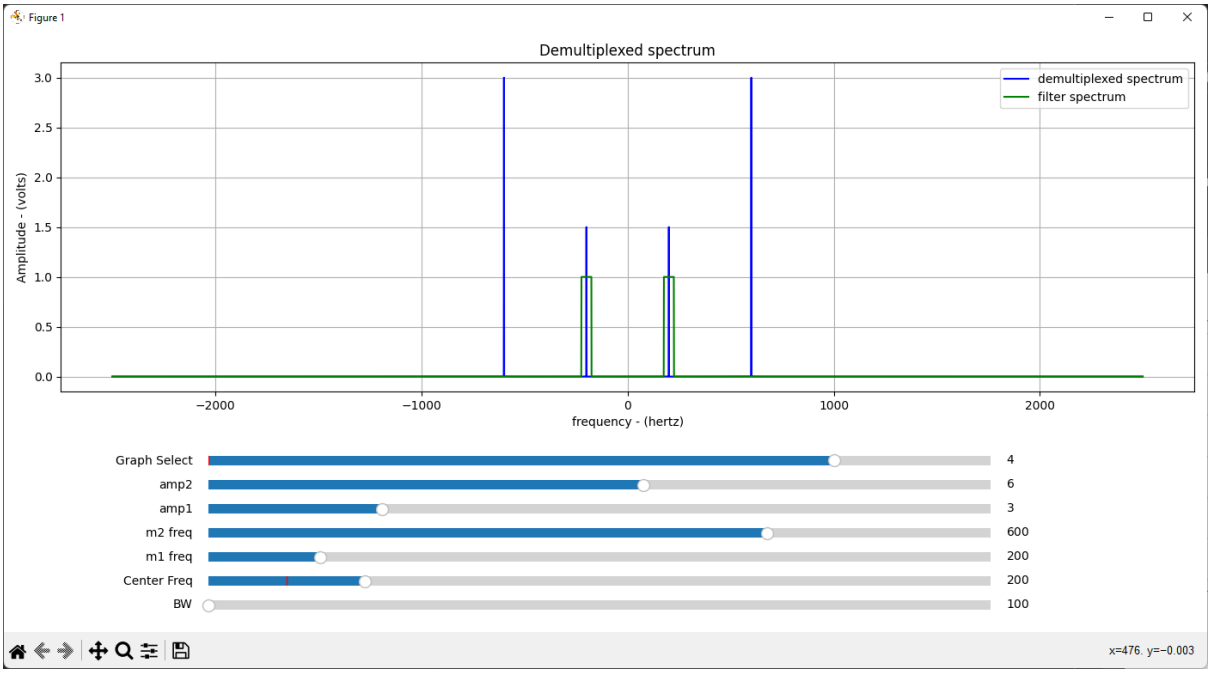


Figure 1
Demultiplexed spectrum
demultiplexed spectrum
filter spectrum
Amplitude - (volts)
frequency - (hertz)
Graph Select 4
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 200
BW 100
x=476. y=−0.003

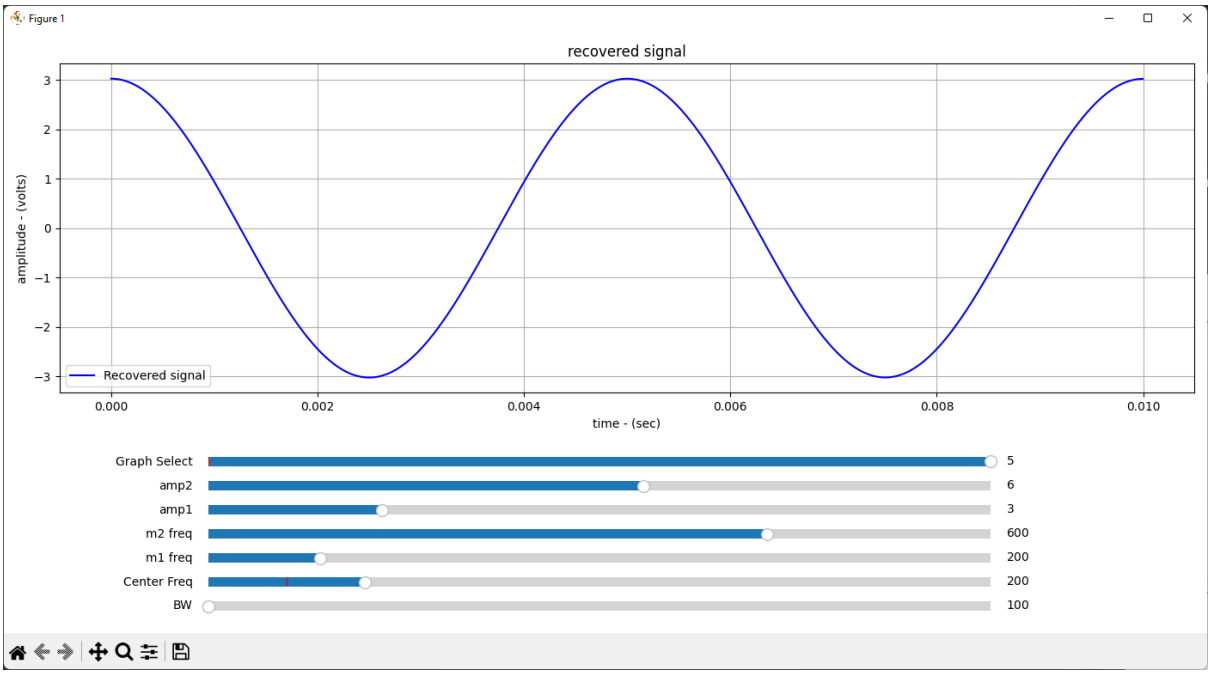


Figure 1
recovered signal
Recovered signal
amplitude - (volts)
time - (sec)
Graph Select 5
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 200
BW 100

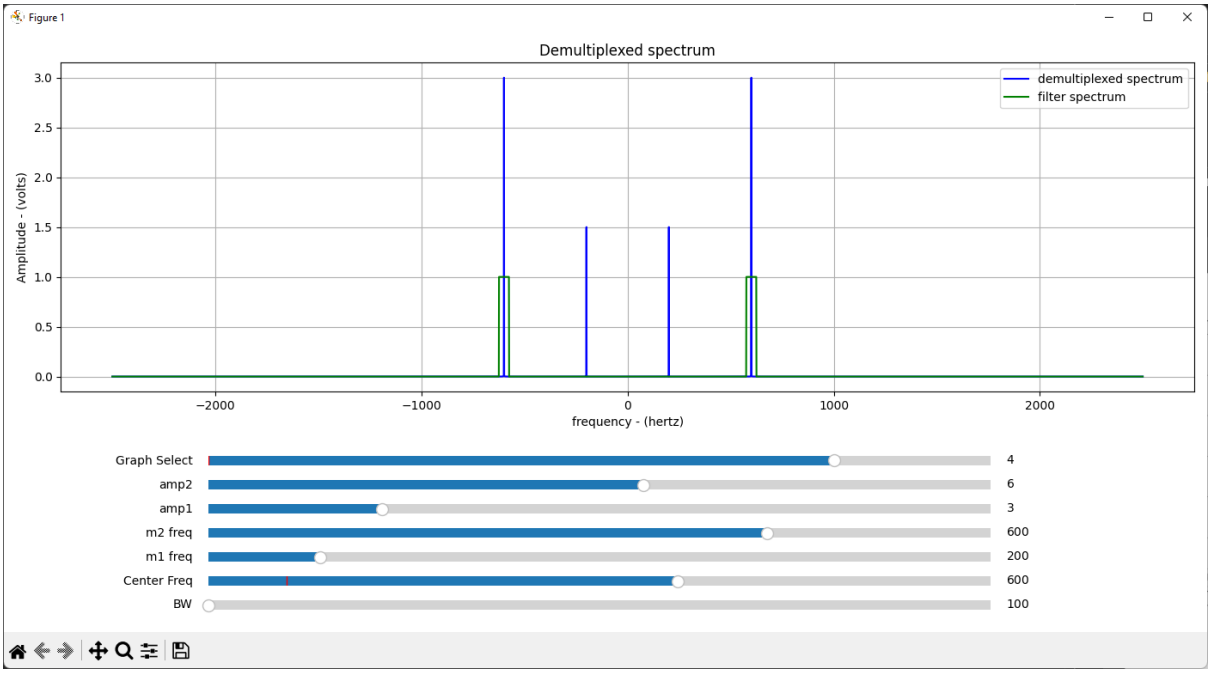


Figure 1
Demultiplexed spectrum
demultiplexed spectrum
filter spectrum
Amplitude - (volts)
frequency - (hertz)
Graph Select 4
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 600
BW 100

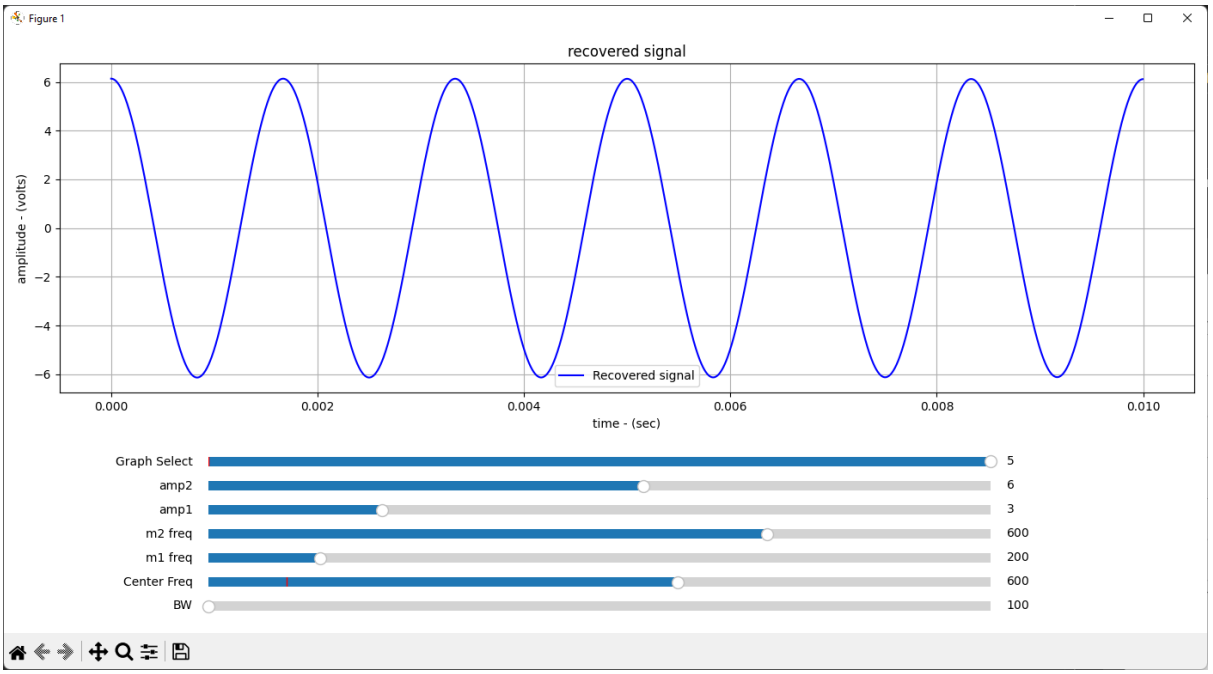


Figure 1
recovered signal
Recovered signal
amplitude - (volts)
time - (sec)
Graph Select 5
amp2 6
amp1 3
m2 freq 600
m1 freq 200
Center Freq 600
BW 100

Parameter Settings:

Sinusoidal Signal 1: Sinusoid with $V_{m1} = 3$ Volts , $f_{m1} = 200$ Hz .

Sinusoidal Signal 2: Sinusoid with $V_{m2} = 6$ Volts , $f_{m2} = 600$ Hz .

BPF filter design parameters for m1: Center Frequency = 200 Hz , BW =100 Hz
BPF filter design parameters for m2: Center Frequency = 600 Hz , BW =100 Hz

Conclusion:

Signature of the Instructor