ECE 398-MA
Introduction to Modern Communication with Python and SDR
Python Lab 3

Noah Breit

February 27, 2025

1 Assignment 1

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import hilbert

# Define simulation parameters
N = 1000     # Number of samples
fs = 100e3   # Sampling rate (Hz)
dt = 1/fs    # Sampling period
fc = 20e3     # Carrier frequency (Hz)
t = np.arange(N) / fs  # Time vector

# FM/PM Modulation Parameters
kf = 500  # Frequency deviation constant (for FM)
kp = np.pi / 2  # Phase deviation constant (for PM)

# Message signal (square wave)
message = np.concatenate([np.ones(N//2), -1*np.ones(N//2)])

############## YOUR CODE STARTS HERE ##############
# FM Modulation (integrate message signal)
fm_signal = np.cos(2 * np.pi * fc * t + 2 * np.pi * kf * np.cumsum(message) * dt)

# PM Modulation (direct phase shift)
pm_signal = np.cos(2 * np.pi * fc * t + kp * message)
############## YOUR CODE ENDS HERE ##############
```

```
# Plot results
plt.figure()
plt.subplot(3, 1, 1)
plt.plot(t, message, label='Message')
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(t, fm_signal, label='FM')
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.legend()

plt.subplot(3, 1, 3)
plt.plot(t, pm_signal, label='PM')
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.legend()
plt.tight_layout()
plt.savefig("assignment1a.png")
plt.show()

import numpy as np
import matplotlib.pyplot as plt

# Define simulation parameters
N = 1000    # Number of samples
fs = 100e3  # Sampling rate (Hz)
dt = 1/fs   # Sampling period
fc = 20e3   # Carrier frequency (Hz)
t = np.arange(N) / fs  # Time vector

# FM/PM Modulation Parameters
kf = 500  # Frequency deviation constant (for FM)
kp = np.pi / 2  # Phase deviation constant (for PM)

# Original Message Signal (square wave)
original_message = np.concatenate([np.ones(N//2), -1*np.ones(N//2)])

# Modified Message Signal (integral of the original square wave)
modified_message = np.cumsum(original_message) * dt

# FM Modulation (integrate original message signal)
integrated_message = np.cumsum(original_message) * dt
fm_signal = np.cos(2 * np.pi * fc * t + 2 * np.pi * kf * integrated_message)
```

```
# PM Modulation (use modified message signal)
pm_signal = np.cos(2 * np.pi * fc * t + kp * modified_message)

# Plot results
plt.figure()
plt.subplot(3, 1, 1)
plt.plot(t, modified_message, label='Modified Message')
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.legend()

plt.subplot(3, 1, 2)
plt.plot(t, fm_signal, label='FM')
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.legend()

plt.subplot(3, 1, 3)
plt.plot(t, pm_signal, label='PM')
plt.xlabel("Time [s]")
plt.ylabel("Amplitude")
plt.legend()
plt.tight_layout()
plt.savefig('assignment1b.png')
plt.show()
```

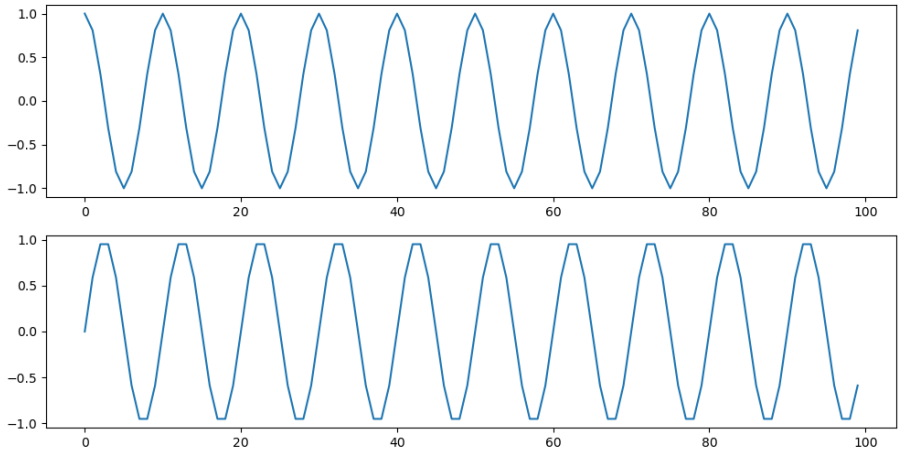

Figure 1: Unmodified FM and PM Signals

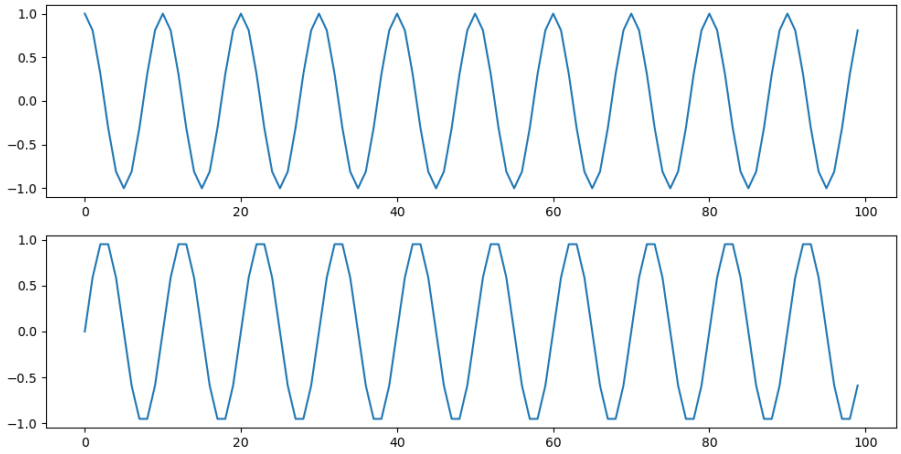

Figure 2: Modified FM and PM Signals

2 Assignment 2

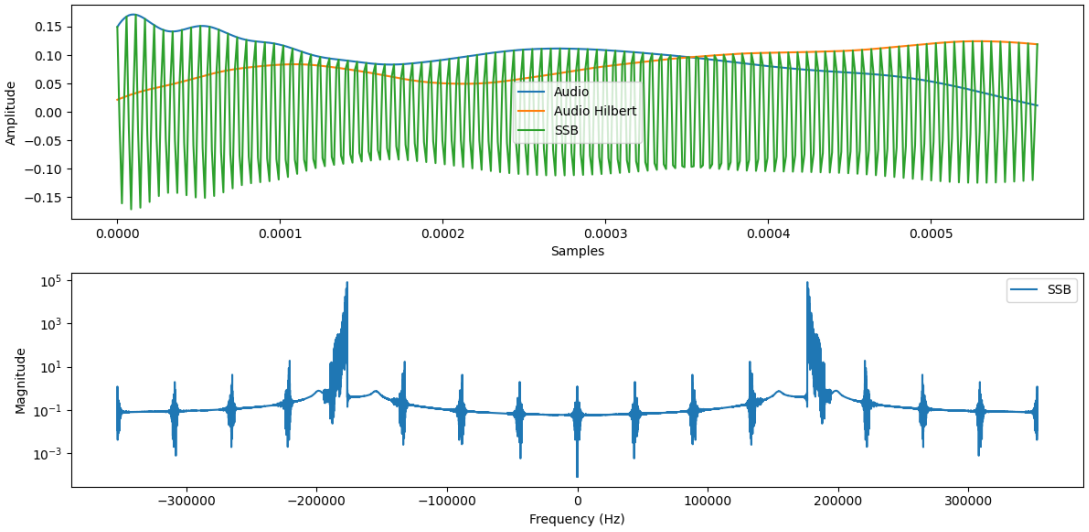

Figure 3: Upper Side Band (USSB) of Audio Signal

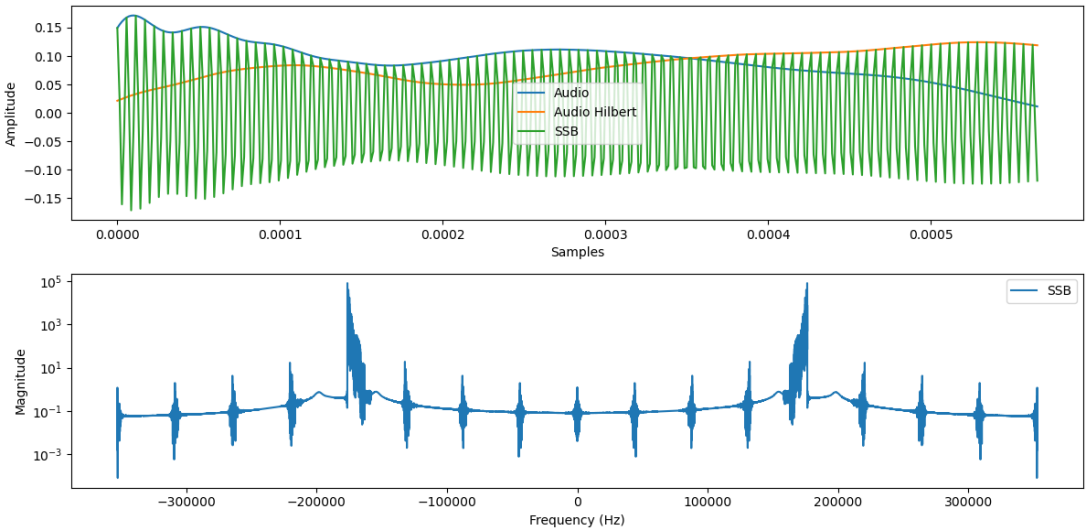

Figure 4: Lower Side Band (LSSB) of Audio Signal

3 Assignment 3

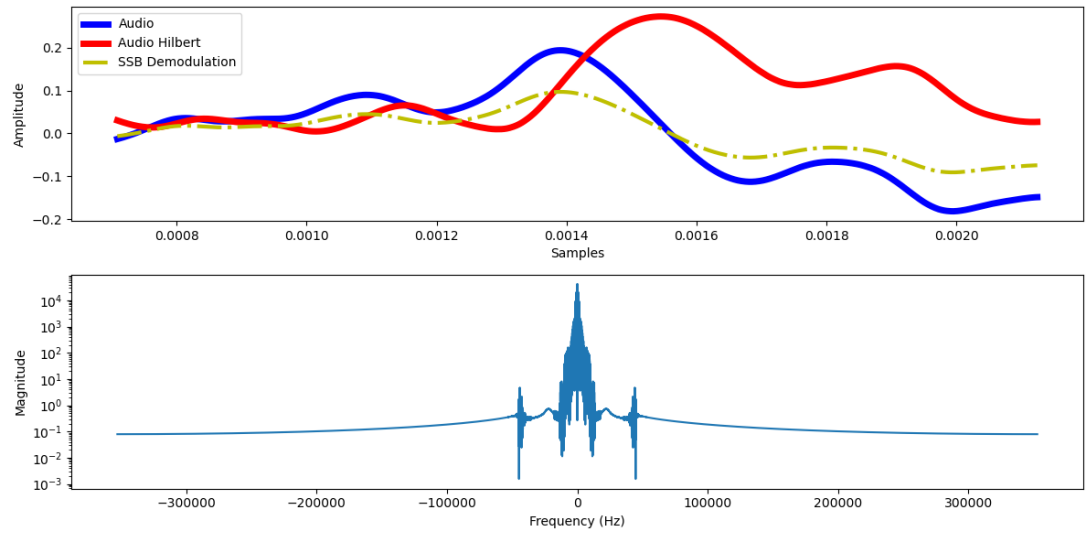

Figure 5: Demodulated Audio Signal: No Phase Offset

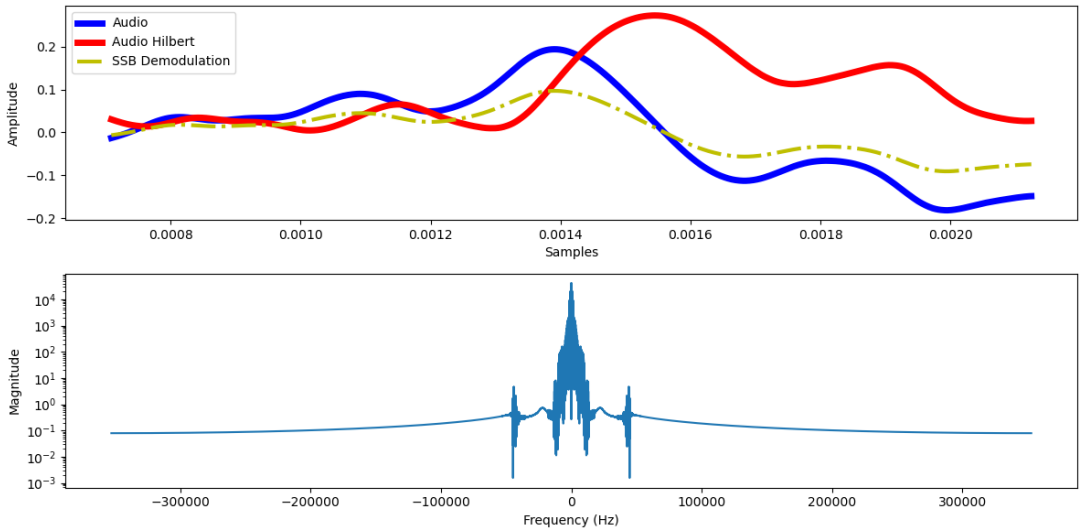

Figure 6: Demodulated Audio Signal: Phase Offset of pi/2

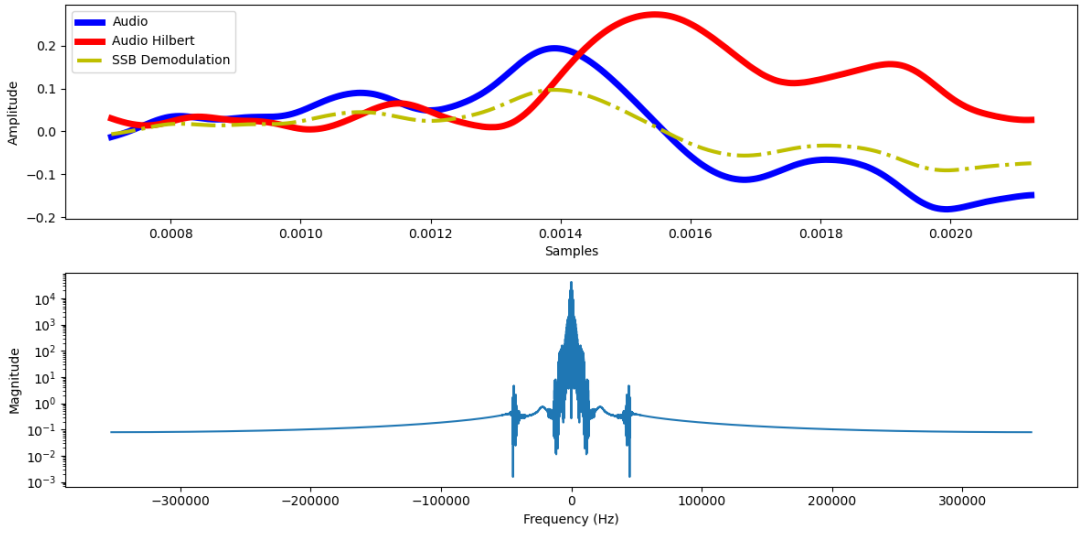

Figure 7: Demodulated Audio Signal: Phase Offset of pi/4