ECE 6410 HW02

Brendon Simonsen

February 4, 2026

Question 1

Code

```python
def p1():
    w = 1        #rad/s
    m = 1

    t1 = 2*np.pi

    t = np.linspace(0, t1, 1000)

    for k in [1,2]:
        eq1 = np.sin(m*w*t) * np.cos(k*w*t)
        eq2 = np.cos(m*w*t) * np.cos(k*w*t)

        #integrate
        i_eq1 = np.trapz(eq1, t)
        i_eq2 = np.trapz(eq2, t)

        print(f"k = {k}")
        print(f"      sin({m}wt)cos({k}wt) dt = {i_eq1:.6e}")
        print(f"      cos({m}wt)cos({k}wt) dt = {i_eq2:.6e}")
```

Two functions are orthogonal on an interval $[a, b]$ if their inner product, defined as the integral of their product over that interval, is zero. In this case, we see that sin and cos are orthogonal for both values of k. In the case of $\cos(m\omega t)$ and $\cos(k\omega t)$, they are not orthogonal when $k = m$, but are orthogonal when $k \neq m$. Below is the terminal output from the code above that proves the orthogonality of the functions.

For $k = 1$:

$$\int \sin(1\omega t)\cos(1\omega t)\,dt = -1.110223 \times 10^{-16}$$

$$\int \cos(1\omega t)\cos(1\omega t)\,dt = 3.141593$$

For $k = 2$:

$$\int \sin(1\omega t)\cos(2\omega t)\,dt = -1.110223 \times 10^{-16}$$

$$\int \cos(1\omega t)\cos(2\omega t)\,dt = -5.551115 \times 10^{-17}$$

---

1 Question 2

Parseval's Theorem proof is provided on the attached document.

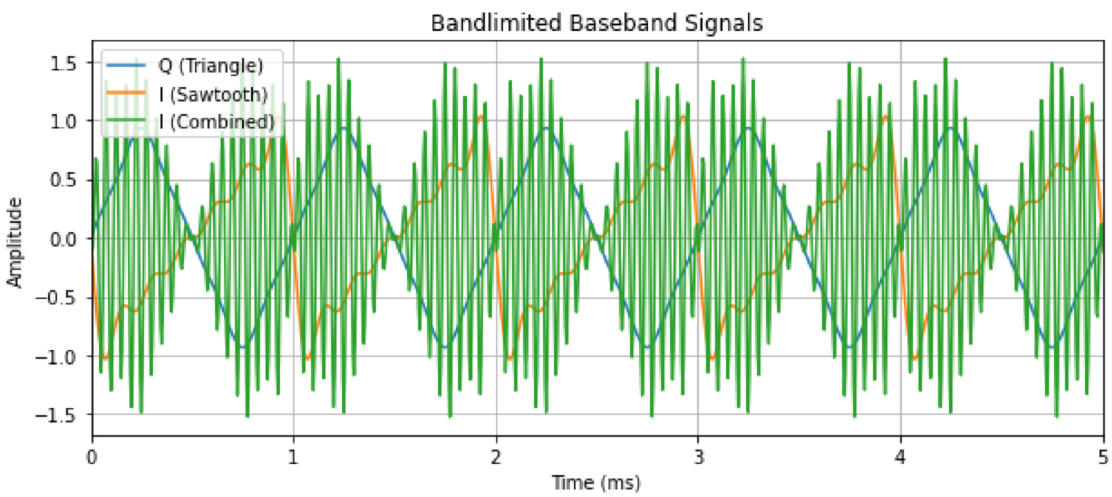
2 Question 3

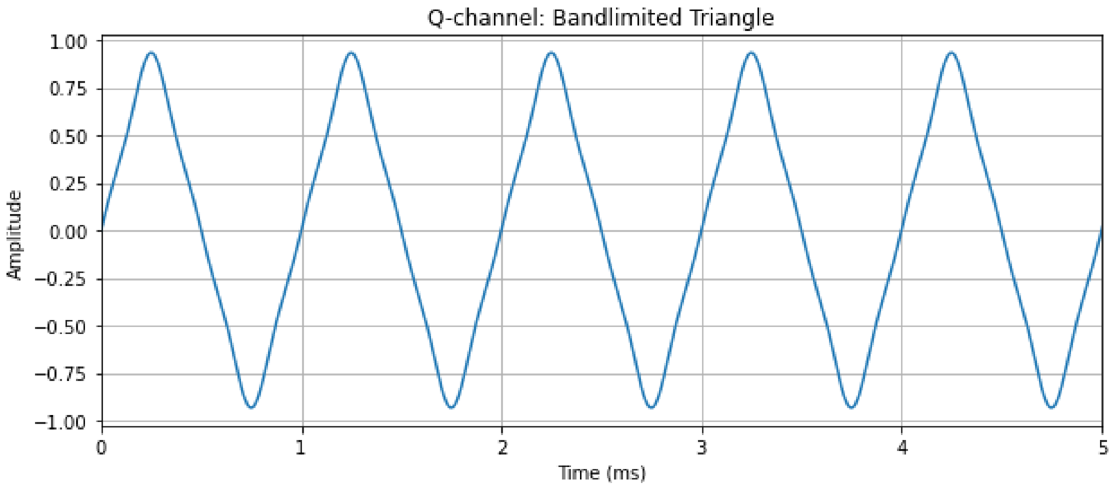
Code

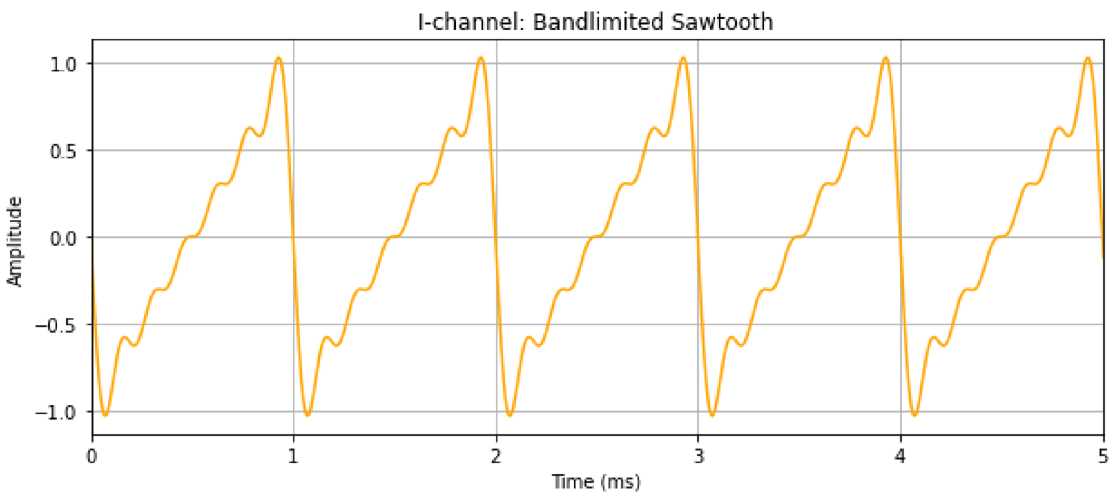
```
def p3():
    f1 = 1e3        # hertz
    fc = 20e3       # hertz
    fs = 200e3      # sampling frequency
    T = 20e-3
    t = np.arange(0, T, 1/fs)
    B = 6e3     # hertz

    # bandlimited sawtooth
    x_i = np.zeros_like(t)
    max_harm = int(B / f1)

    for n in range(1, max_harm + 1):
        x_i += (1/n) * np.sin(2 * np.pi * n * f1 * t)

    x_i *= 2 / np.pi  # normalization
    x_i *= -1

    # bandlimited triangle
    x_q = np.zeros_like(t)

    for n in range(1, max_harm + 1, 2):  # odd harmonics only
        x_q += ((-1)**((n-1)//2)) * (1/n**2) * np.sin(2 * np.pi * n * f1 *
            t)

    x_q *= 8 / (np.pi**2)  # normalization

    # modulation
    carrier_cos = np.cos(2 * np.pi * fc * t)
    carrier_sin = np.sin(2 * np.pi * fc * t)

    x_iq = x_i * carrier_cos - x_q * carrier_cos

    time = 5

    plt.figure(figsize=(10,4))
    plt.plot(t*1e3, x_q, label="Q (Triangle)")
    plt.plot(t*1e3, x_i, label="I (Sawtooth)")
    plt.plot(t*1e3, x_iq, label="I (Combined)")
    plt.xlim(0, time)
    plt.xlabel("Time (ms)")
    plt.ylabel("Amplitude")
    plt.legend()
    plt.title("Bandlimited Baseband Signals")
```

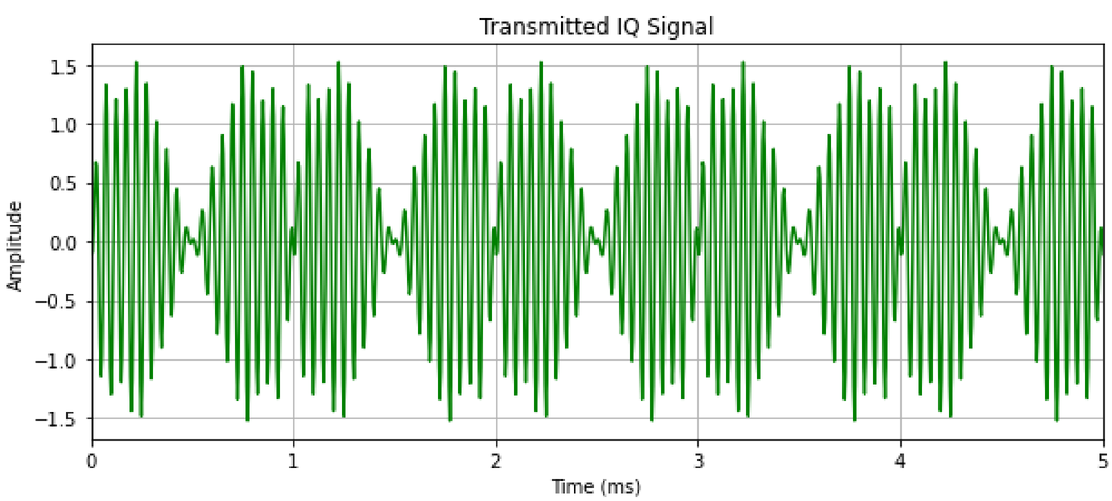
```
    plt.grid()
    plt.show()

    # --- Q-channel: Triangle ---
    plt.figure(figsize=(10, 4))
    plt.plot(t*1e3, x_q)
    plt.xlim(0, time)
    plt.xlabel("Time (ms)")
    plt.ylabel("Amplitude")
    plt.title("Q-channel: Bandlimited Triangle")
    plt.grid(True)

    # --- I-channel: Sawtooth ---
    plt.figure(figsize=(10, 4))
    plt.plot(t*1e3, x_i, color='orange')
    plt.xlim(0, time)
    plt.xlabel("Time (ms)")
    plt.ylabel("Amplitude")
    plt.title("I-channel: Bandlimited Sawtooth")
    plt.grid(True)

    # --- IQ-combined signal ---
    plt.figure(figsize=(10, 4))
    plt.plot(t*1e3, x_iq, color='green')
    plt.xlim(0, time)
    plt.xlabel("Time (ms)")
    plt.ylabel("Amplitude")
    plt.title("Transmitted IQ Signal")
    plt.grid(True)

    # Display all figures
    plt.show()
```

Generated Plot




Q-channel: Bandlimited Triangle
Amplitude
Time (ms)


I-channel: Bandlimited Sawtooth
Amplitude
Time (ms)


Transmitted IQ Signal
Amplitude
Time (ms)

Question 4

Attached document contains some notes used to help with question 4.

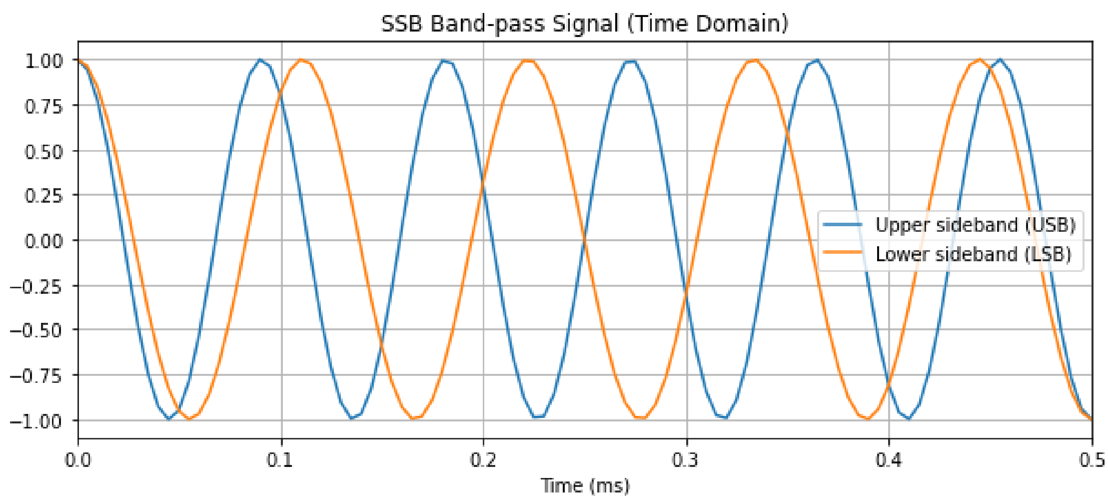
Code

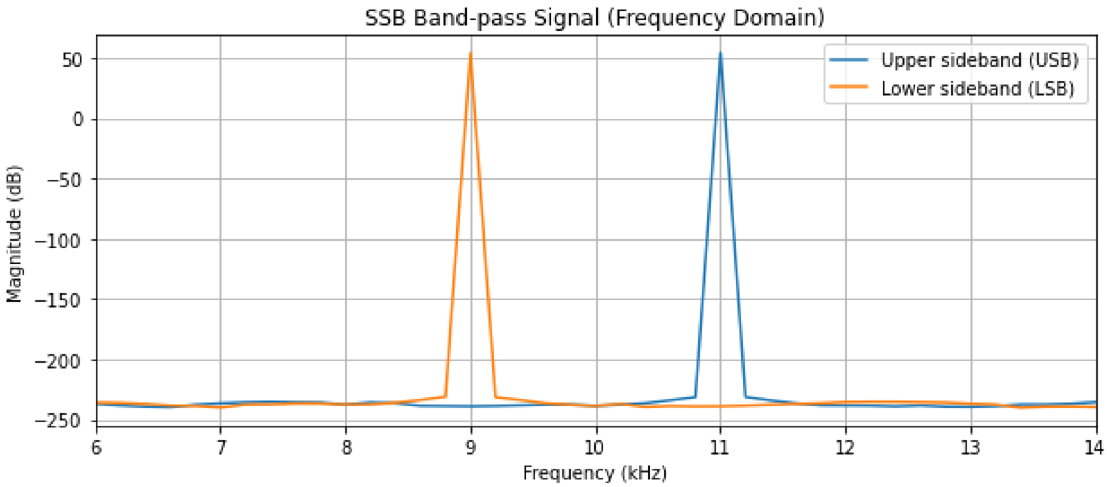
```
def p4():
    fs = 200e3          # sampling rate
    T = 5e-3
    t = np.arange(0, T, 1/fs)

    fm = 1e3            # message frequency
    fc = 10e3          # carrier frequency

    # m(t)
    m = np.cos(2*np.pi*fm*t)

    # hilbert transform
    m_hilbert = np.imag(hilbert(m))

    # single sideband modulation
    c = np.cos(2*np.pi*fc*t)
    s = np.sin(2*np.pi*fc*t)

    # upper sideband
    x_ssb = m*c - m_hilbert*s

    # lower sideband (LSB)
    x_lsb = m*c + m_hilbert*s

    # time-domain plot
    plt.figure(figsize=(10,4))
    plt.plot(t*1e3, x_ssb, label="Upper sideband (USB)")
    plt.plot(t*1e3, x_lsb, label="Lower sideband (LSB)")
    plt.xlim(0, 0.5)
    plt.xlabel("Time (ms)")
    plt.title("SSB Band-pass Signal (Time Domain)")
    plt.grid()
    plt.legend()
    plt.show()

    # frequency-domain plot (USB and LSB on same axes)

    N = len(x_ssb)

    X_usb = np.fft.fftshift(np.fft.fft(x_ssb))
    X_lsb = np.fft.fftshift(np.fft.fft(x_lsb))

    f = np.fft.fftshift(np.fft.fftfreq(N, 1/fs))

    plt.figure(figsize=(10,4))
    plt.plot(f/1000, 20*np.log10(np.abs(X_usb) + 1e-12),
```

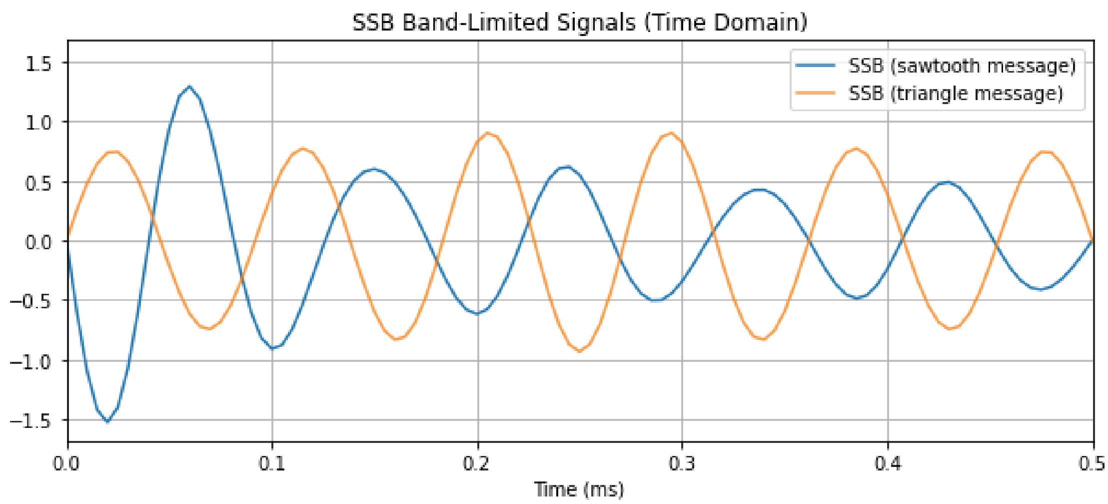
```
                label="Upper sideband (USB)")
    plt.plot(f/1000, 20*np.log10(np.abs(X_lsb) + 1e-12),
                label="Lower sideband (LSB)")
    plt.xlim(6, 14)
    plt.xlabel("Frequency (kHz)")
    plt.ylabel("Magnitude (dB)")
    plt.title("SSB Band-pass Signal (Frequency Domain)")
    plt.grid()
    plt.legend()
    plt.show()


    # expand to band-limited
    # bandlimited sawtooth
    f1 = 1e3     # hertz
    B = 6e3      # hertz
    x_i = np.zeros_like(t)
    max_harm = int(B / f1)

    for n in range(1, max_harm + 1):
        x_i += (1/n) * np.sin(2 * np.pi * n * f1 * t)

    x_i *= 2 / np.pi  # normalization
    x_i *= -1

    m = x_i
    m_hat = np.imag(hilbert(m))

    x_ssb = m*c - m_hat*s

    # bandlimited triangle
    x_q = np.zeros_like(t)

    for n in range(1, max_harm + 1, 2):  # odd harmonics only
        x_q += ((-1)**((n-1)//2)) * (1/n**2) * np.sin(2 * np.pi * n * f1 * t)

    x_q *= 8 / (np.pi**2)  # normalization

    m = x_q   # or x_q, from your earlier work
    m_hat = np.imag(hilbert(m))

    x_q_ssb = m*c - m_hat*s

    # time-domain plot
    plt.figure(figsize=(10,4))
    plt.plot(t*1e3, x_ssb, label="SSB (sawtooth message)")
    plt.plot(t*1e3, x_q_ssb, label="SSB (triangle message)", alpha=0.8)
    plt.xlim(0, 0.5)
    plt.xlabel("Time (ms)")
    plt.title("SSB Band-Limited Signals (Time Domain)")
    plt.grid()
    plt.legend()
    plt.show()
```

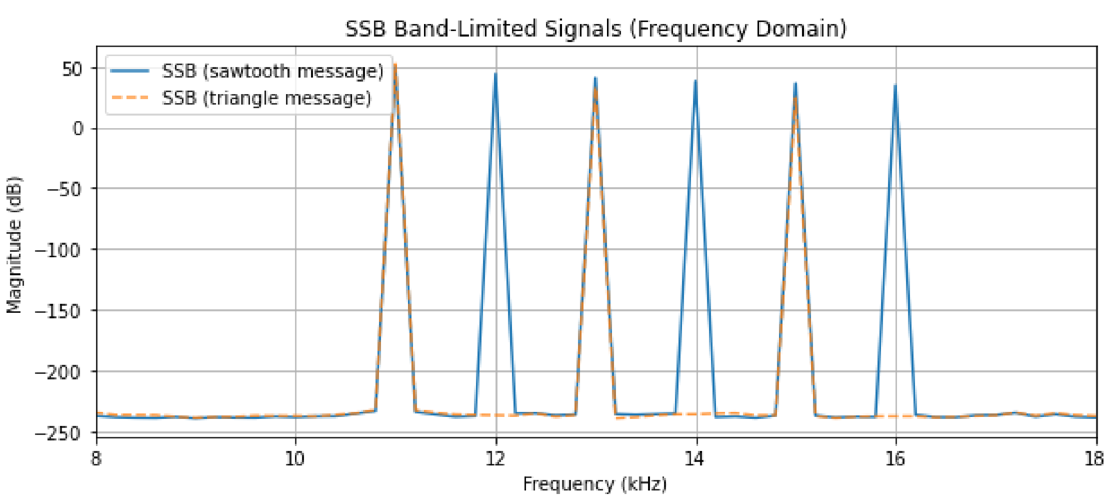
```python

# ----- frequency domain: both SSB signals on one plot -----

N = len(x_ssb)

X_saw = np.fft.fftshift(np.fft.fft(x_ssb))
X_tri = np.fft.fftshift(np.fft.fft(x_q_ssb))

f = np.fft.fftshift(np.fft.fftfreq(N, 1/fs))

plt.figure(figsize=(10,4))
plt.plot(f/1000, 20*np.log10(np.abs(X_saw) + 1e-12),
         label="SSB (sawtooth message)")
plt.plot(f/1000, 20*np.log10(np.abs(X_tri) + 1e-12),
         label="SSB (triangle message)",linestyle="--", alpha=0.8)

plt.xlim(8, 18)
plt.xlabel("Frequency (kHz)")
plt.ylabel("Magnitude (dB)")
plt.title("SSB Band-Limited Signals (Frequency Domain)")
plt.grid()
plt.legend()
plt.show()
```

Generated Plot




SSB Band-pass Signal (Frequency Domain)
Upper sideband (USB)
Lower sideband (LSB)
Magnitude (dB)
Frequency (kHz)


SSB Band-Limited Signals (Time Domain)
SSB (sawtooth message)
SSB (triangle message)
Time (ms)


SSB Band-Limited Signals (Frequency Domain)
SSB (sawtooth message)
SSB (triangle message)
Magnitude (dB)
Frequency (kHz)