ECE 6410 HW01

Brendon Simonsen

January 22, 2026

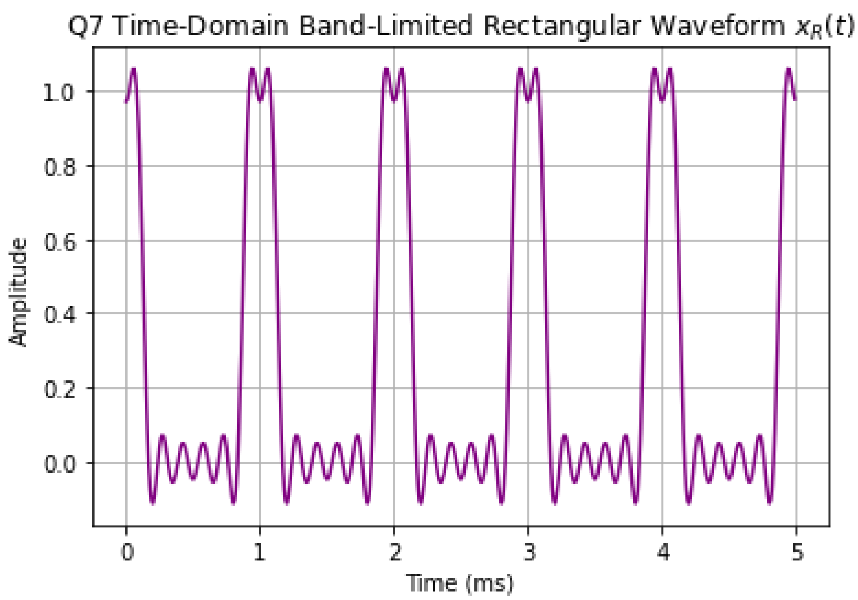
1 Question 7

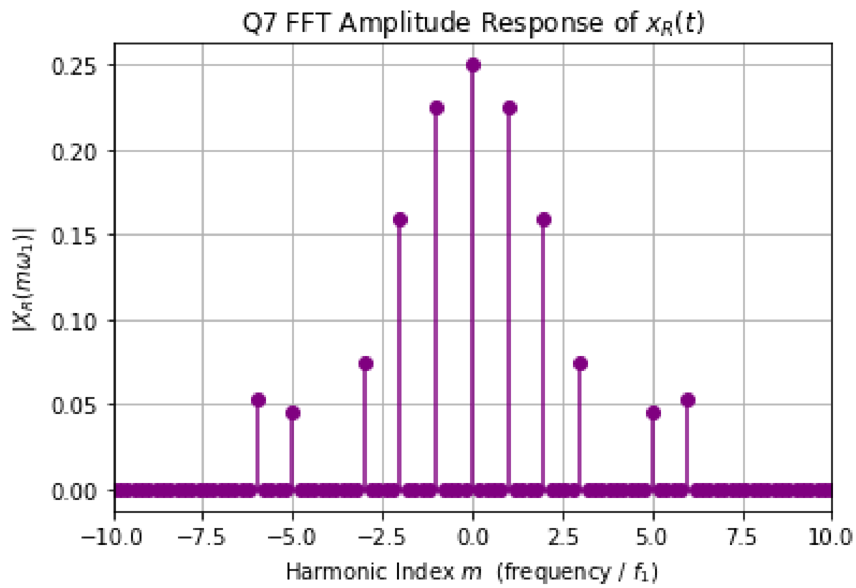
Code

```
def question7():
    fs = 100e3          # Sampling frequency (Hz)
    f1 = 1e3            # Fundamental / PRF (Hz)
    D = 0.25            # Duty cycle (25%)
    BW = 6e3            # Bandwidth (Hz)

    T = 5e-3            # Signal duration (5 ms)
    t = np.arange(0, T, 1/fs)

    # calculate the harmonics allowed by bandwitdth
    n_max = int(BW / f1)
    #n_max = 20

    x_R = D * np.ones_like(t)  # DC term (unipolar)

    # Fourier series summation
    for n in range(1, n_max + 1):
        coeff = (2 / (n * np.pi)) * np.sin(n * np.pi * D)
        x_R += coeff * np.cos(2 * np.pi * n * f1 * t)

    # Time-domain plot
    plt.figure()
    plt.plot(t * 1e3, x_R, color='purple')
    plt.xlabel("Time (ms)")
    plt.ylabel("Amplitude")
    plt.title("Q7 Time-Domain Band-Limited Rectangular Waveform $x_R(t)$")
    plt.grid(True)
    plt.show()

    # Compute the FFT
    X_R = np.fft.fft(x_R)
    freqs = np.fft.fftfreq(len(X_R), d=1/fs)

    # Shift FFT for centered spectrum
    X_R_shift = np.fft.fftshift(X_R)
    freqs_shift = np.fft.fftshift(freqs)

    # Magnitude normalization
```

```
    X_mag = np.abs(X_R_shift) / len(X_R)

    # FFT outpu
    plt.figure()
    markerline, stemlines, baseline = plt.stem(freqs_shift / f1, X_mag,
        basefmt=" ")    # divide by f1 to normalize values to multiples of
        f1
    plt.setp(markerline, color='purple')   # top markers
    plt.setp(stemlines, color='purple')   # vertical stems
    plt.setp(baseline, color='black')     # baseline (optional)
    plt.xlim(-10, 10)  # show harmonics around DC
    plt.xlabel("Harmonic Index $m$  (frequency / $f_1$)")
    plt.ylabel(r"$|X_R(m\omega_1)|$")
    plt.title("Q7 FFT Amplitude Response of $x_R(t)$")
    plt.grid(True)
    plt.show()
```

Generated Plot





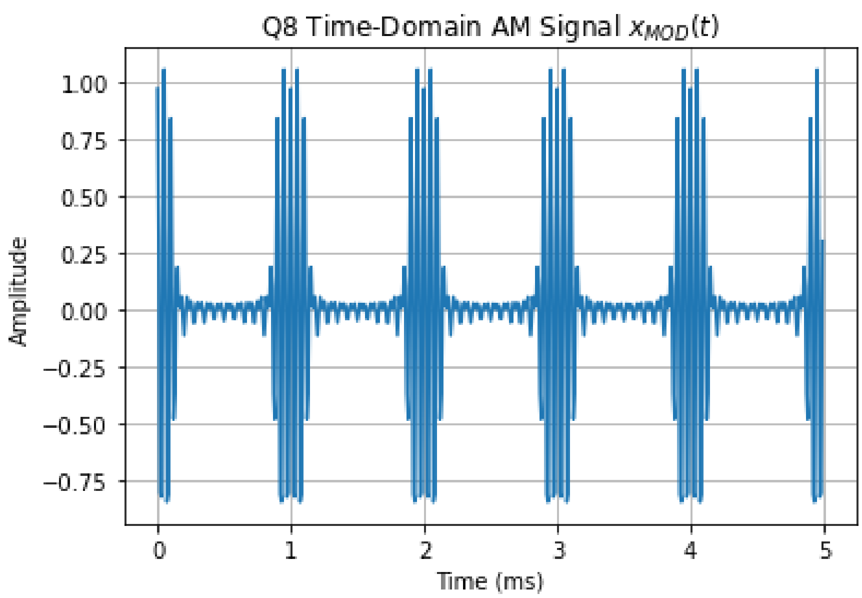
2 Question 8

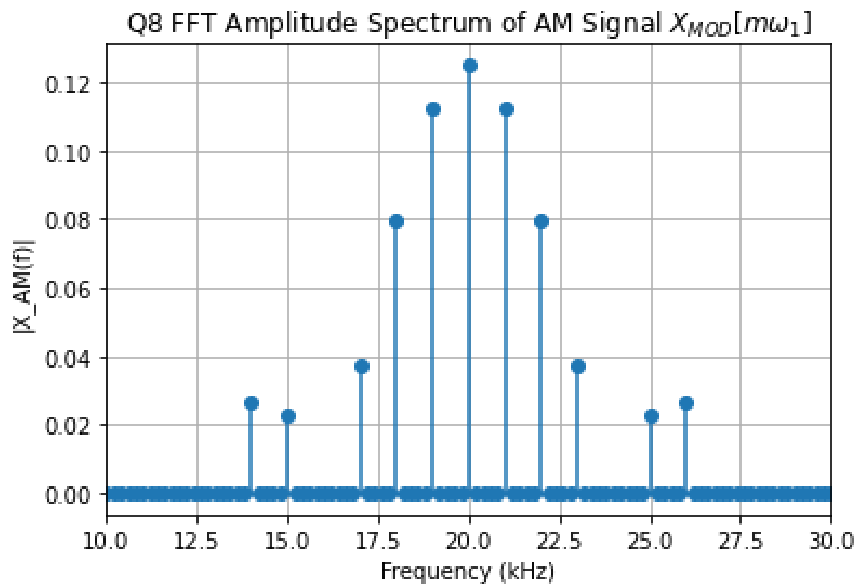
Code

```
def question8():
    # Constants
    fs = 100e3              # Sampling frequency
    f1 = 1e3                # Pulse repetition frequency (PRF)
    D = 0.25                # Duty cycle
    BW = 6e3                # Bandwidth of pulse
    T = 5e-3                # Duration (5 ms)
    fc = 20e3               # Carrier frequency (Hz)

    # Time vector
    t = np.arange(0, T, 1/fs)

    # generate x_R(t), rect wave form
    n_max = int(BW / f1)
    x_R = D * np.ones_like(t)  # DC term

    for n in range(1, n_max + 1):
        coeff = (2 / (n * np.pi)) * np.sin(n * np.pi * D)
        x_R += coeff * np.cos(2 * np.pi * n * f1 * t)

    # AM modulation
    x_AM = x_R * np.cos(2 * np.pi * fc * t)

    # time-domain plot
    plt.figure()
    plt.plot(t * 1e3, x_AM)
    plt.xlabel("Time (ms)")
    plt.ylabel("Amplitude")
    plt.title("Q8 Time-Domain AM Signal x_AM(t)")
```

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

    # FFT of modulated signal
    X_AM = np.fft.fft(x_AM)
    freqs = np.fft.fftfreq(len(X_AM), d=1/fs)

    # shift for centered spectrum
    X_AM_shift = np.fft.fftshift(X_AM)
    freqs_shift = np.fft.fftshift(freqs)

    # Magnitude normalization
    X_mag = np.abs(X_AM_shift) / len(X_AM)

    # Frequency-domain plot
    plt.figure()
    plt.stem(freqs_shift / f1, X_mag, basefmt=" ", use_line_collection=
       True)
    plt.xlim(10, 30)  # show 0  50   kHz
    plt.xlabel("Frequency (kHz)")
    plt.ylabel("|X_AM(f)|")
    plt.title("Q8 FFT Amplitude Spectrum of AM Signal")
    plt.grid(True)
    plt.show()
```

Generated Plot



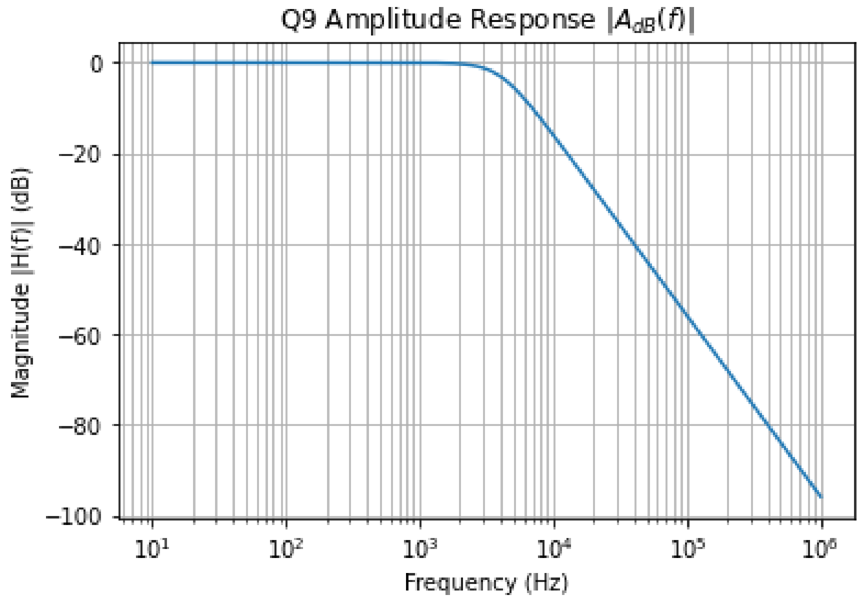


3 Question 9

Code

```python
def question9():
    R = 1.0
    L = 56.28e-6
    C = 28.13e-6

    # freq axis, log scaled
    f = np.logspace(1,6,3000)
    omega = 2 * np.pi * f
    s = 1j * omega

    # transfer function
    num = 1 / (L * C)
    den = s**2 + (1 / (R * C)) * s + (1 / (L * C))

    H = num / den

    # convert to dB
    H_dB = 20 * np.log10(np.abs(H))

    # plot
    plt.figure()
    plt.semilogx(f, H_dB)
    plt.xlabel("Frequency (Hz)")
    plt.ylabel("Magnitude |H(f)| (dB)")
    plt.title("Q9 Amplitude Response |A_dB(f)|")
    plt.grid(True, which="both")
    plt.show()
```

Generated Plot



4 Question 12

Code

```python
def q12b():

    # parameters
    f1 = 1e3            # signal frequency
    fs = 16e3           # sampling frequency
    Ts = 1/fs
    N = 16              # samples per period

    n = np.arange(N)
    t = n * Ts

    # create waveform
    duty = 0.25
    high_samples = int(duty * N)

    x = np.zeros(N)
    x[:high_samples] = 1.0     # 1 for first 25%, 0 otherwise

    # filter params
    R = 1.0
    L = 56.28e-6
    C = 28.13e-6

    num_a = [1/(L*C)]
    den_a = [1, 1/(R*C), 1/(L*C)]

```

```
    # bilinear transformation
    b_d, a_d = signal.bilinear(num_a, den_a, fs)

    # digital filtering
    y = signal.lfilter(b_d, a_d, x)

    # PLOT
    Nplot = 16
    t1 = t[:Nplot]
    x1 = x[:Nplot]
    y1 = y[:Nplot]

    n1 = np.arange(Nplot)

    # -----------------------
    # Plot (both as stem plots)
    # -----------------------
    plt.figure(figsize=(8,4))

    # plot using time
    #m1, s1, b1 = plt.stem(t1*1e3, x1, basefmt=" ", label="Input x[n]")
    #m2, s2, b2 = plt.stem(t1*1e3, y1, basefmt=" ", label="Filtered y[n]")

    # plot using nodes
    m1, s1, b1 = plt.stem(n1, x1, basefmt=" ", label="Input x[n]")
    m2, s2, b2 = plt.stem(n1, y1, basefmt=" ", label="Filtered y[n]")

    # Change color of second stem plot
    plt.setp(m2, color='red')
    plt.setp(s2, color='red')

    plt.xlabel("Time (ms)")
    plt.ylabel("Amplitude")
    plt.title("Q12 Digital LPF Input and Output ")
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()
```

Generated Plot