EE338: Take-Home Midsemester Examination

Submission By ⇒ Anupam Rawat
22B3982

M = 104

Reviewed By ⇒ Jatin Kumar 22B3922
Rishabh Bharadwaj 22B3962

TASK

We are required to design an Infinite-Impulse-Response (IIR) filter with bands lying in two frequency ranges (Group I & II). The overall filter should be a multi-band pass filter and the pass-bands and stop-bands are required to be monotonic.

The filter type that satisfies these criterion is ButterWorth filter, which would be used as baseline.

SPECIFICATIONS REQUIRED

- The analog signal is bandlimited to 280kHz, and it's ideally sampled with a sampling rate 630kHz.

  now, ~~bandwidth~~ $2 \times$ bandwidth < sampling rate;

  the sampling obeys Nyquist Criteria & can be reconstructed w/o loss.

- The tolerance for stopband & passband are $\delta = 0.15$ in magnitude
- $M = 104$ ⟶ $Q = \text{floor}(M/11) = 9$
  ⟶ $R = M \pmod{11} = 5$
- Range of Group I frequency: $(40+5D)$ to $(70+5D)$; where $D = Q$

  lower edge $= 40 + 5\times 9 = 85\text{kHz}$

  upper edge $= 70 + 5\times 9 = 115\text{kHz}$

- Range of Group II frequency:

$$(170+5D) \text{ to } (200+5D) \text{ where } D=R=5.$$

Lower edge $= 170 + 5\times5 = 195$ kHz

Upper edge $= 200 + 5\times5 = 225$ kHz

- The transition on each side of the passbands is 5kHz. Group I & II ranges represent the two passband.

DESIRED FREQUENCY RESPONSE

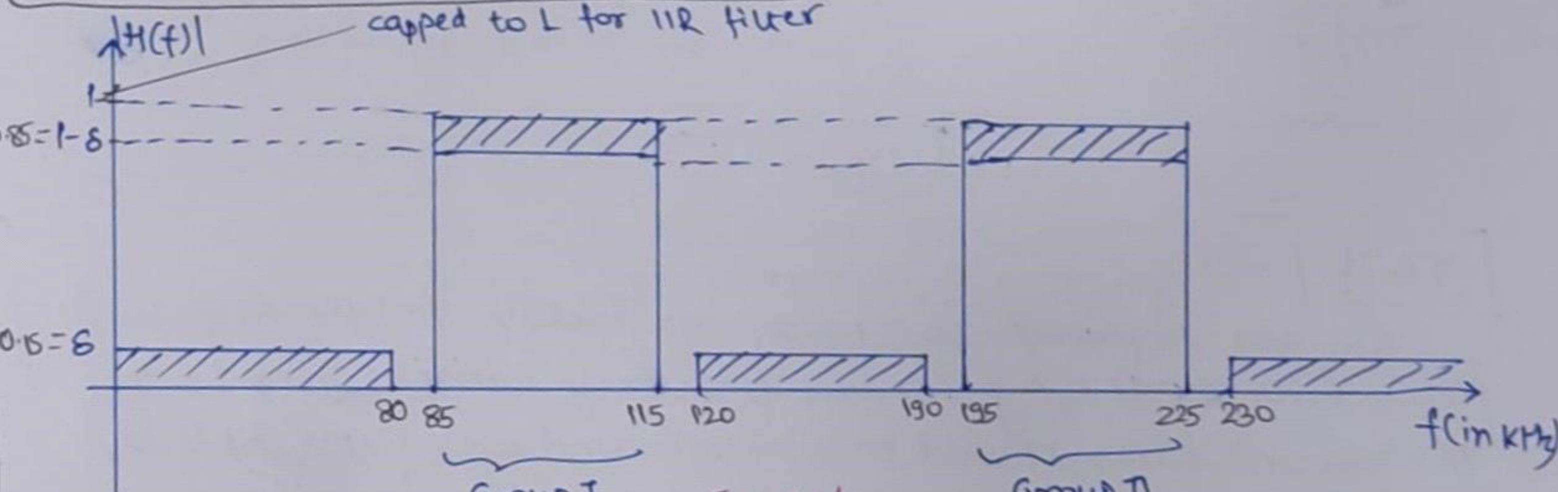


Figure-1

Realization of the filter response

The filter response can be realised cascading a Bandpass filter with Bandstop filter in series OR by adding two Bandpass filters in parallel. For the purpose of this assignment, we'll be using the two cascading filters, the details of both of them are plotted below.

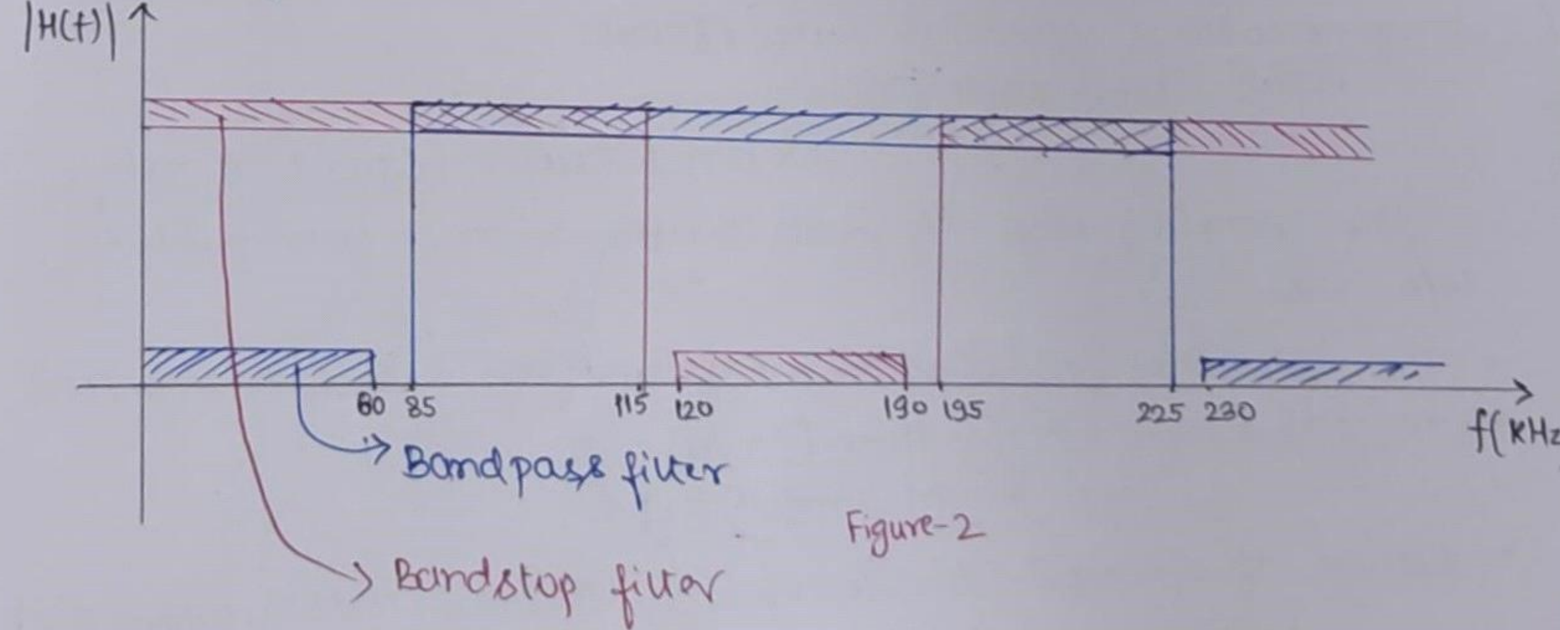


Figure-2

DESIGNING FILTER

We now require to design the Bandpass and Bandstop filter

- <u>Bandpass filter</u>
  - → passband range required: 85kHz − 225kHz
  - → passband tolerance : δ_1 → bandpass tolerance: 5kHz
  - → stopband tolerance : δ_2
- <u>Bandstop filter</u>
  - → stopband range required: 120kHz − 190kHz
  - → passband tolerance: δ_3 → stopband tolerance: 5kHz
  - → stopband tolerance: δ_4

After combining these two filters in cascade; following will be the value of tolerances & the respective f ⇒

$$\begin{cases} \delta_2(1-\delta_3) & 0^+ \le f < 80\,\text{kHz} \\ (1-\delta_1)(1-\delta_3) & 85\,\text{kHz} \le f < 115\,\text{kHz} \\ \delta_4(1-\delta_1) & 120\,\text{kHz} \le f < 190\,\text{kHz} \\ (1-\delta_1)(1-\delta_3) & 195\,\text{kHz} \le f < 225\,\text{kHz} \\ \delta_2(1-\delta_3) & 230\,\text{kHz} \le f < 315\,\text{kHz} \end{cases}$$

Equating these tolerance values to the required tolerance values as per figure-1

⇒ equating for $85\,\text{kHz} \le f < 115\,\text{kHz}$ and $195\,\text{kHz} \le f < 225\,\text{kHz}$

$$(1-\delta_1)(1-\delta_3) = 1-\delta = 0.85$$

$$1-\delta_1-\delta_3+\delta_1\delta_3 = 0.85$$

$$\boxed{\delta_1+\delta_3-\delta_1\delta_3 = 0.15}$$

⇒ equating for $0^+\text{kHz} \le f < 80\,\text{kHz}$ and $230\,\text{kHz} \le f < 315\,\text{kHz}$

$$\delta_2(1-\delta_3) = \delta = 0.15$$

But equating for $120\,\text{kHz} \le f < 190\,\text{kHz}$ gives us $\boxed{\delta_4(1-\delta_1) = \delta = 0.15}$

~~Since $1-\delta_3$ can't be zero;~~ $\boxed{\delta_2 = \delta_4}$ & ~~$\boxed{\delta_2(1-\delta_3) = \delta_4(1-\delta_3) = 0.15}$~~

~~Assu By symmetry~~

Assuming symmetry; $\delta_1 = \delta_3$ & $\delta_2 = \delta_4$

$$\delta_1+\delta_3-\delta_1\delta_3 = \delta_1+\delta_1-\delta_1^2 = 0.15 \Rightarrow \delta_1^2 - 2\delta_1 + 0.15 = 0$$

$$\delta_1 = \frac{+2 \pm \sqrt{4-0.60}}{2}$$

$$= \frac{2 \pm 1.84}{2} = 1 \pm 0.92$$

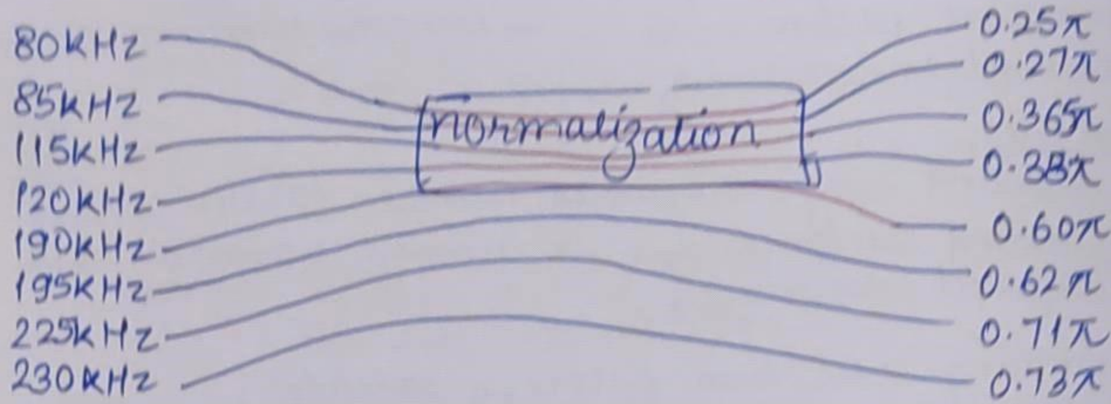
$$= 1.92 \text{ OR } 0.08$$

δ_1 can't be > 1 ⇒ $\boxed{\boxed{\delta_1 = 0.08} = \delta_3}$

$$\delta_2(1-\delta_3) = 0.15 \Rightarrow \delta_2 = \frac{0.15}{0.92} = 0.16 \qquad \boxed{\delta_2 = \delta_4 = 0.16}$$

Normalizing the frequency

$$\omega = \frac{\Omega \times 2\pi}{\Omega_{sampling}} = \frac{(\Omega)\pi}{315 \times 10^3}$$



Converting the values to analog

$$\Omega = \tan(\omega/2).$$

f(KHz)	80	85	115	120	190	195	225	230
normalized frequency	0.25π	0.27π	0.365π	0.38π	0.60π	0.62π	0.71π	0.73π
analog equivalent	0.41	0.45	0.645	0.68	1.38	1.47	2.04	2.21

Calculating Bandpass filter Analog characteristics

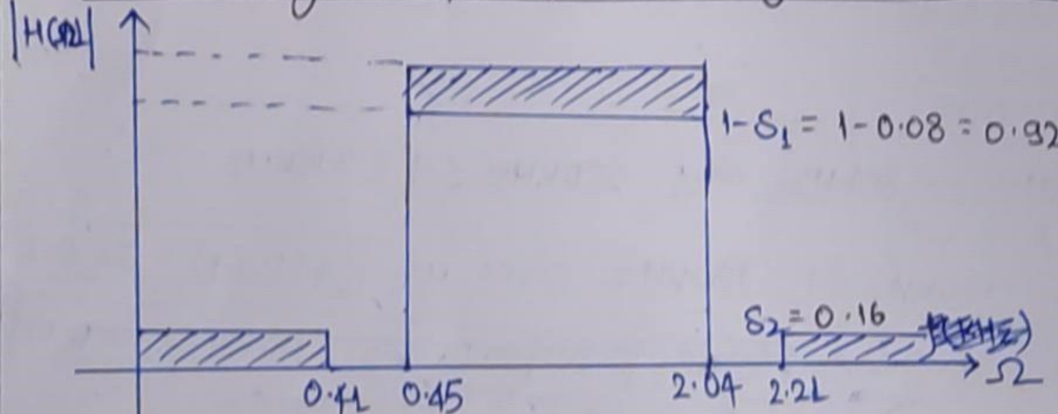
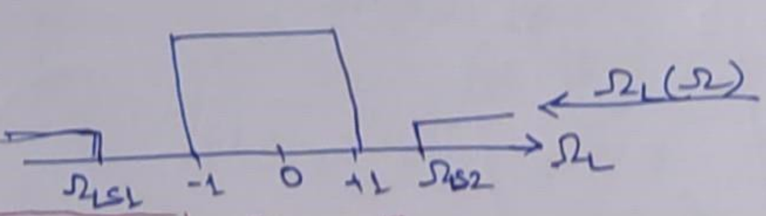


Figure-3

Using the result for converting a Bandpass filter to ButterWorth LPF; as taught in class on "10th February 2025 (Monday) (L16)":

$$\Omega_L = \frac{\Omega^2 - \Omega_0^2}{B\Omega} \quad \& \quad \Omega_0^2 = \Omega_{p1}\Omega_{p2} \quad \& \quad B = \Omega_{p2} - \Omega_{p1}$$

for



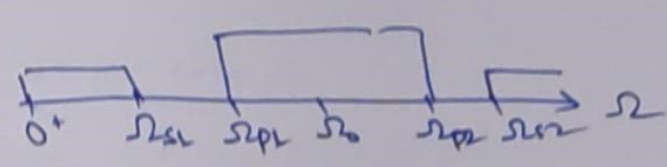


Figure - 4

Figure - 5

Equating the values of Ω by comparing the graphs in Fig-3 & Fig-4

$\Omega_{s1} = 0.41$ $\quad\quad$ $\Omega_{p1} = 0.45$

$\Omega_{s2} = 2.21$ $\quad\quad$ $\Omega_{p2} = 2.04$

substituting the above values of Ω_{p1} & Ω_{p2} into the formulae of $\Omega_L \Rightarrow$

$$\Omega_L = \frac{\Omega^2 - (0.45)(2.04)}{\Omega(-0.45 + 2.04)} = \frac{\Omega^2 - 0.918}{1.59\Omega}$$

As per figure-4 & figure-5;

Ω_{p2} gets mapped to $+1$

$\Omega_{p1} \longrightarrow -1$

$\Omega_{s1} \longrightarrow \Omega_{Ls1}$

$\Omega_{s2} \longrightarrow \Omega_{Ls2}$

calculating the values of Ω_{Ls1} & Ω_{Ls2};

$$\Omega_{Ls1}(\Omega) = \Omega_{Ls1}(0.41) = \frac{(0.41)^2 - 0.918}{1.59 \times 0.41} = -1.15$$

$$\Omega_{Ls2}(\Omega) = \Omega_{Ls2}(02.21) = \frac{(2.21)^2 - 0.918}{1.59 \times 2.21} = 1.128 \sim 1.13$$

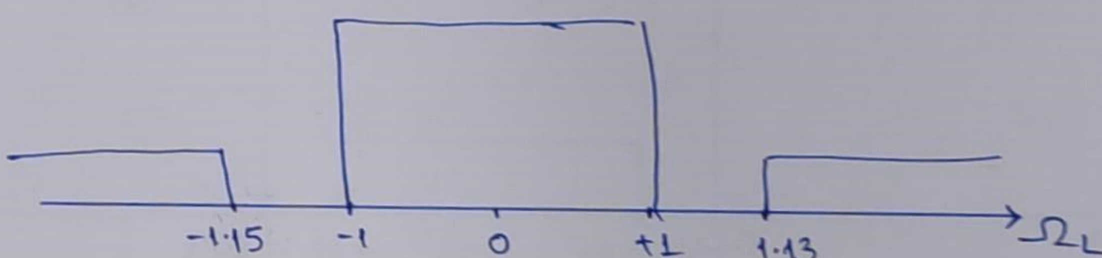


Figure - 6

<u>specifications of low pass filter</u>

Ω_p passband edge $= +1$

Ω_s stopband edge $= \min(|1.13|, |-1.15|) = 1.13$

$$D_1 = \frac{1}{(1-\delta_1)^2} - 1 = \frac{1}{(0.92)^2} - 1 = 0.18$$

$$D_2 = \frac{1}{\delta_2^2} - 1 = \frac{1}{(0.16)^2} - 1 = 38$$

Using the result for calculating the order of a ~~two~~ Butterworth filter; as obtained on '<u>04th February, 2025 (Tuesday) (L-14)</u>'

$$N \geq \left\lceil \frac{1}{2} \cdot \frac{\log(D_2/D_1)}{\log(\Omega_s/\Omega_p)} \right\rceil$$

substituting the values calculated to obtain order;

$$N \geq \left\lceil \frac{1}{2} \cdot \frac{\log(38/0.18)}{\log(1.13/1)} \right\rceil = \left\lceil \frac{1}{2} \cdot \frac{(2.3245)}{(0.053)} \right\rceil$$

$$N \geq \lceil 21.89 \rceil$$

$\boxed{N = 22}$ → N should be taken as minimum as possible for resource efficiency.

Values for Bandpass filter → Butterworth filter

$D_1 = 0.18$ $\quad \Omega_s = 1.13$ $\quad N = 22.$

$D_2 = 38$ $\quad \Omega_p = 1$

Calculating Bandstop filter Analog characteristics

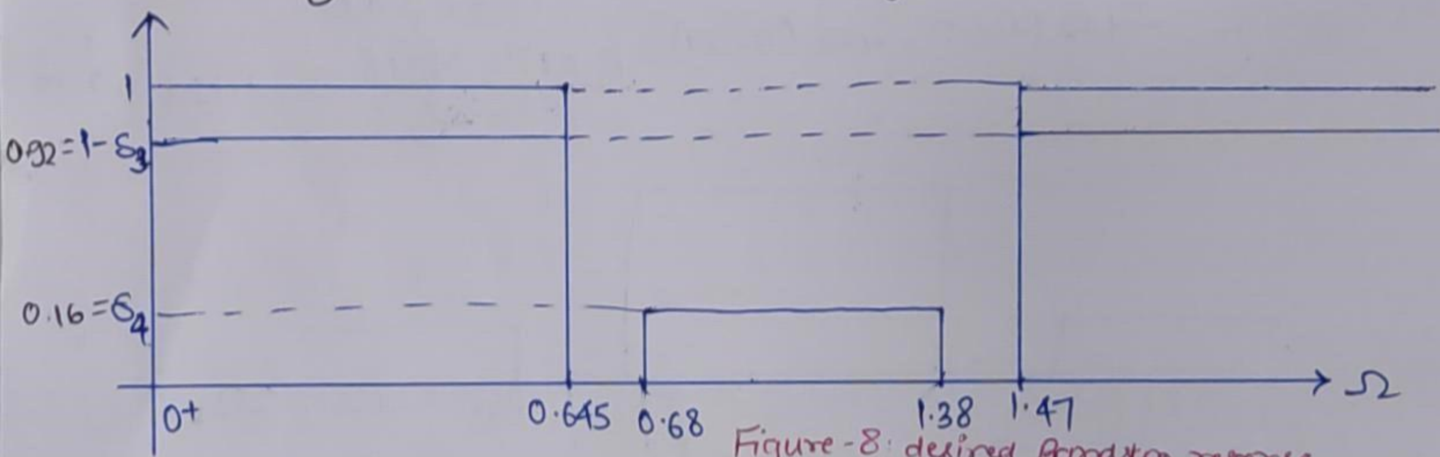


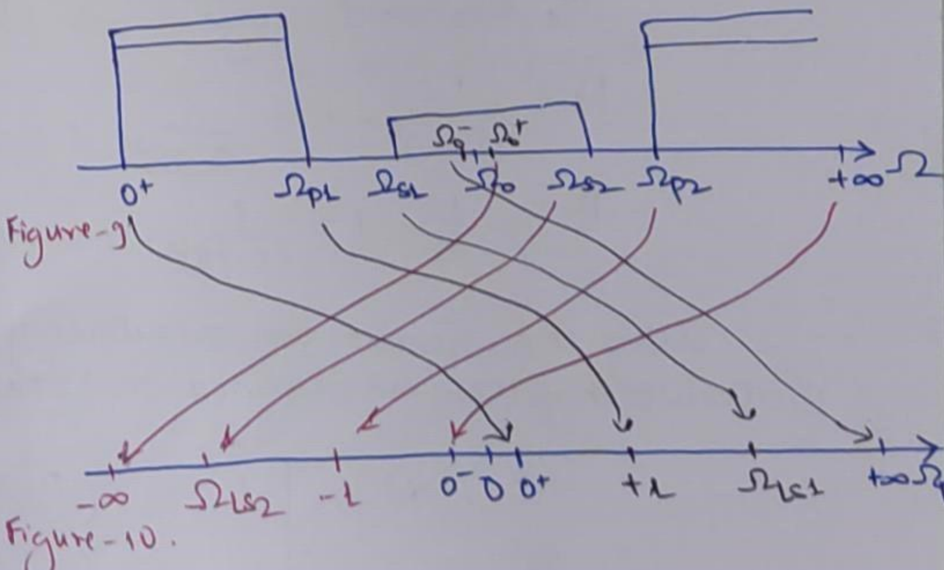
Figure-8: desired Bandstop response.

Using the result of Bandstop filter specifications as discussed in class on "10th February, 2025 (Monday) (L-16)".

$$\Omega_L = \frac{B\Omega}{\Omega_0^2 - \Omega^2}$$

→ $B = \Omega_{p2} - \Omega_{p1}$

→ $\Omega_0^2 = \Omega_{p1}\Omega_{p2}$



comparing figure 8 & figure-9; we obtain :-

$$\Omega_{p1} = 0.645 \qquad \Omega_{s1} = 0.68$$
$$\Omega_{p2} = 1.47 \qquad \Omega_{s2} = 1.38$$

$$\Omega_0^2 = \Omega_{p1} \times \Omega_{p2} = 0.645 \times 1.47 = 0.94815$$
$$B = \Omega_{p2} - \Omega_{p1} = 1.47 - 0.645 = 0.825$$

$$\Omega_{LS1} = \frac{B\Omega_{s1}}{\Omega_0^2 - \Omega_{s1}^2} = \frac{(0.825)(0.68)}{(0.94815) - 0.68^2} = 1.154$$

$$\Omega_{LS2} = \frac{B\Omega_{s2}}{\Omega_0^2 - \Omega_{s2}^2} = \frac{(0.825)(1.38)}{0.94815 - 1.38^2} = -1.19$$

for the lowpass filter

(Ω_p) passband edge $= 1$

(Ω_s) stopband edge $= \min(|\Omega_{LS1}|, |\Omega_{LS2}|) = 1.154$

$$D_1 = \frac{1}{(1-\delta_3)^2} - 1 = \frac{1}{(1-0.08)^2} - 1 = 0.18$$

$$D_2 = \frac{1}{\delta_4^2} - 1 = \frac{1}{(0.16)^2} - 1 = 38$$

$$N \geq \left\lceil \frac{1}{2} \cdot \frac{\log(D_2/D_1)}{\log(\Omega_s/\Omega_p)} \right\rceil = \left\lceil \frac{1}{2} \cdot \frac{\log(38/0.18)}{\log(1.154/1)} \right\rceil$$
$$= \left\lceil \frac{1}{2} \cdot \frac{(2.3245)}{(0.062)} \right\rceil = \lceil 18.745 \rceil$$

once again; N should be chosen as less as possible to save on resources; $\boxed{N = 19}$

Bandpass filter

$$H_{analog, LPF_1}(s) = \frac{1}{1 + \left(\frac{s_L}{j\Omega_c}\right)^{2N}}$$

$$\frac{\Omega_p}{(D_1)^{1/2N}} \leq \Omega_c \leq \frac{\Omega_s}{(D_2)^{1/2N}}$$

$$\frac{1}{(0.18)^{1/44}} \leq \Omega_c \leq \frac{1.13}{(38)^{1/44}} \rightarrow 1.0397 \leq \Omega_c \leq 1.04$$

$1.0397 \leq \Omega_c \leq 1.0403 \rightarrow$ let's take $\boxed{\Omega_c = 1.04}$

$$H_{analog, filter 1}(s) = \frac{1}{1 + \left(\frac{s}{j(1.04)}\right)^{2 \times 22}}$$

<u>Bandstop filter</u>

$$\frac{\Omega_p}{(D_1)^{1/2N}} \leq \Omega_c \leq \frac{\Omega_s}{(D_2)^{1/2N}}$$

$$\frac{1}{(0.18)^{1/38}} \leq \Omega_c \leq \frac{1.154}{(38)^{1/38}}$$

$$1.04615 \leq \Omega_c \leq 1.04865$$

let's take $\boxed{\Omega_c = 1.047}$

$$H_{analog, filter 2}(s) = \frac{1}{1 + (s/j.1.047)^{2 \times 19}}$$

The poles "s_k" of a Butterworth filter are given by

$$s_k = j\Omega_c e^{-j\frac{(2k+1)}{2N}\pi} \quad \text{where } k \in [0, N-1] \;\&\; k \in \mathbb{Z}, \; N \in \mathbb{Z}$$

$$H_{filter}(s) = \frac{\prod_{k=1}^{N} s_k}{\prod_{k=1}^{N}(s - s_k)}$$

<u>for digital filter</u>

$$s \leftarrow \frac{1 - z^{-1}}{1 + z^{-1}}$$

now, since N is very large; we'll compute the poles using python.

Midsemester Assignment Submission: EE 338, Spring 2024-25

Submission By:
Anupam Rawat, 22b3982
Filter Number (M): 104

Reviewed By:
Jatin Kumar, 22b3922
Rishabh Bhardwaj, 22b3962

February 20, 2025

1 Introduction

Due to the order of the system being very large, it was not feasible to draw all the poles by hand, hence this report includes the plot of poles and the response of the system. The code was written in Python, and the code which was used is included at the end of the file.

2 Plotting the Poles

Poles determine the stability and frequency response of filters. Poles for the BandPass, BandStop and Cascaded Overall filter are included below:-

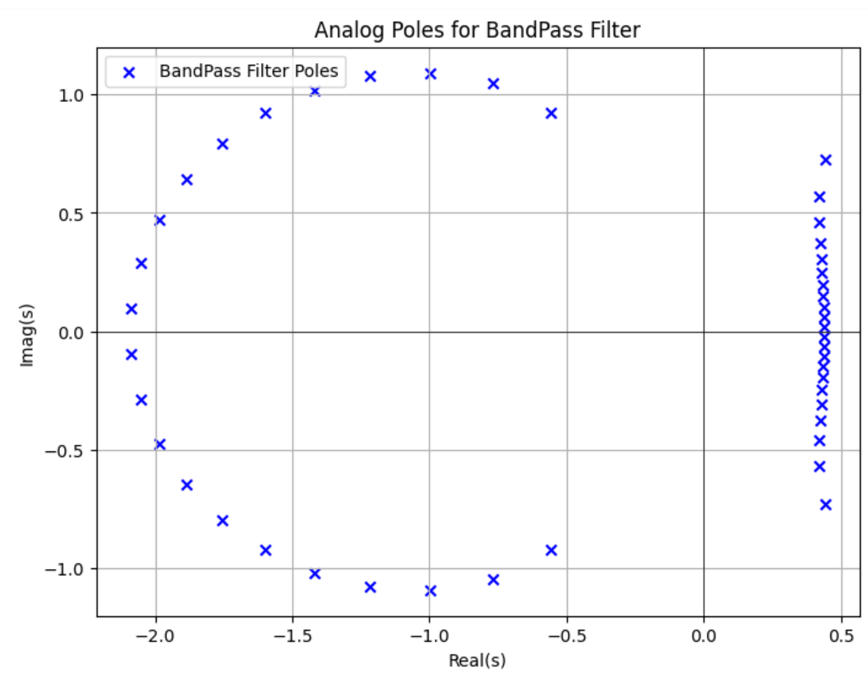


Figure 1: Poles for Bandpass Filter

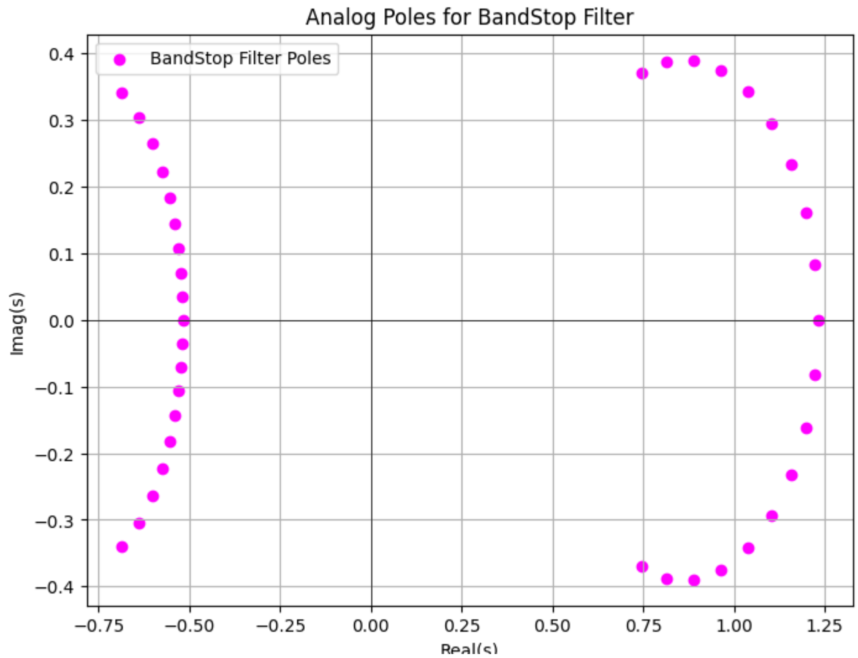


Figure 2: Poles for Bandstop Filter

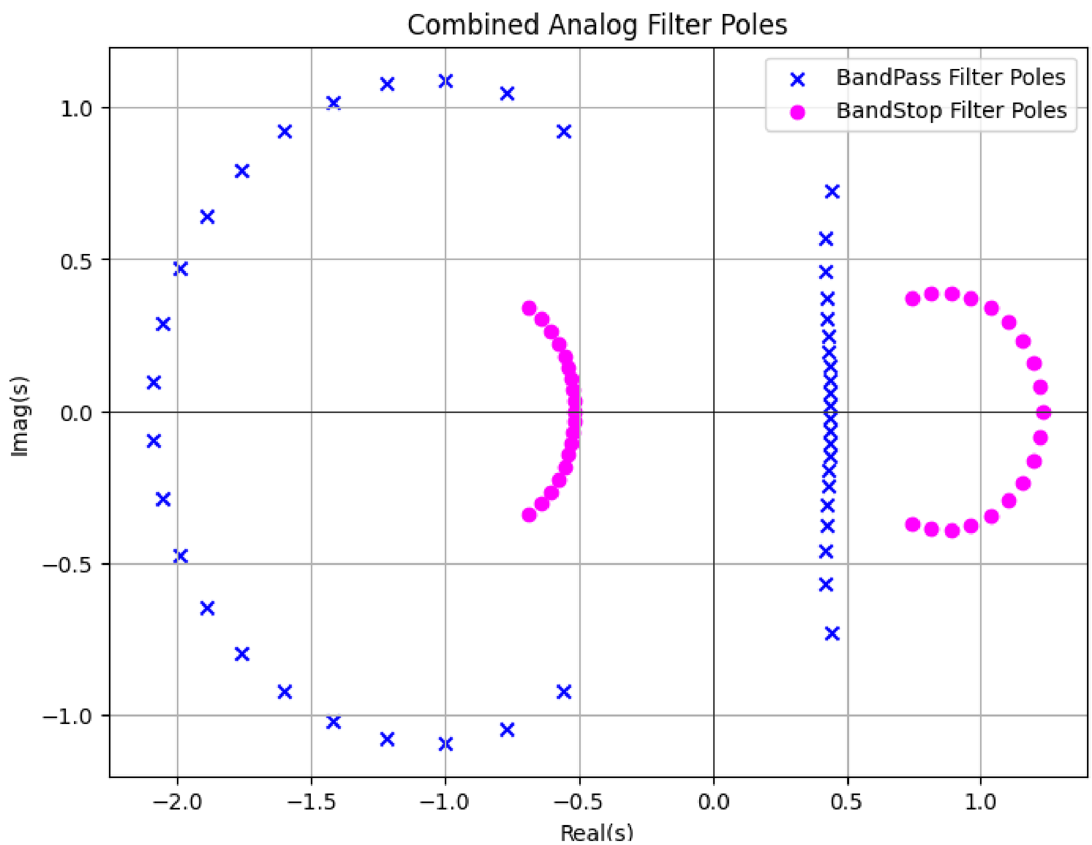


Figure 3: Combined Filter Poles

3 Analog Filter Responses

The response for BandPass and BandStop filter is given below:-

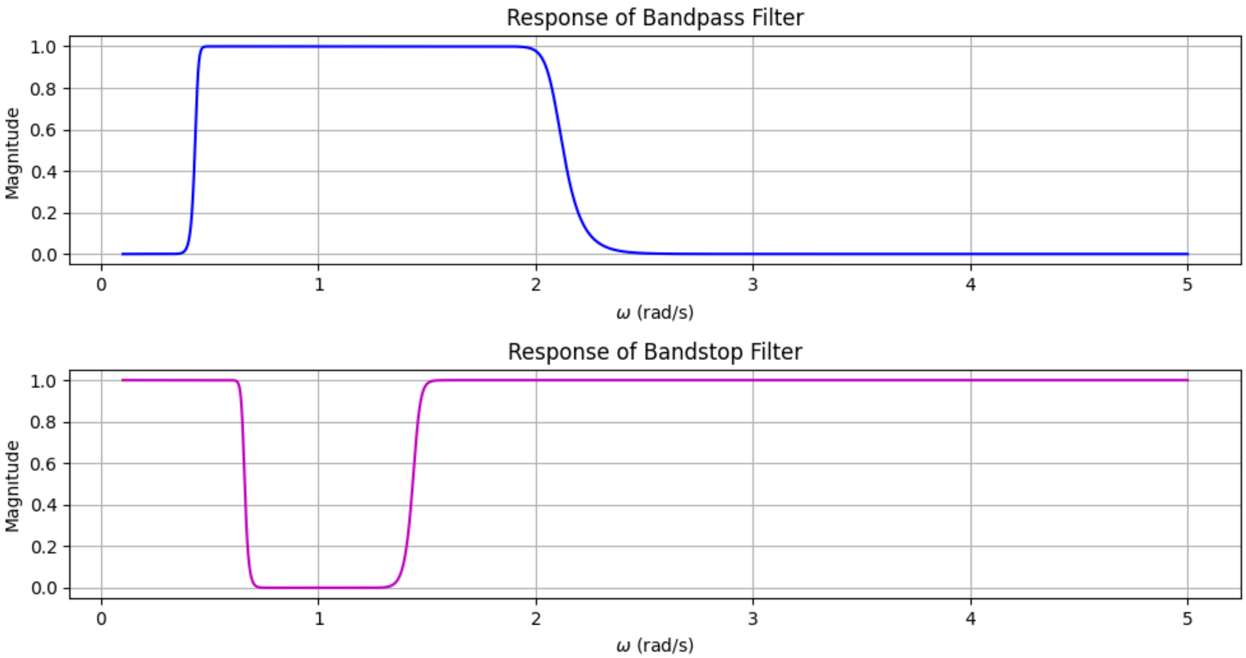


Figure 4: Bandpass and Bandstop Filter Response

The combined transfer function for the overall cascaded filter is given as below:

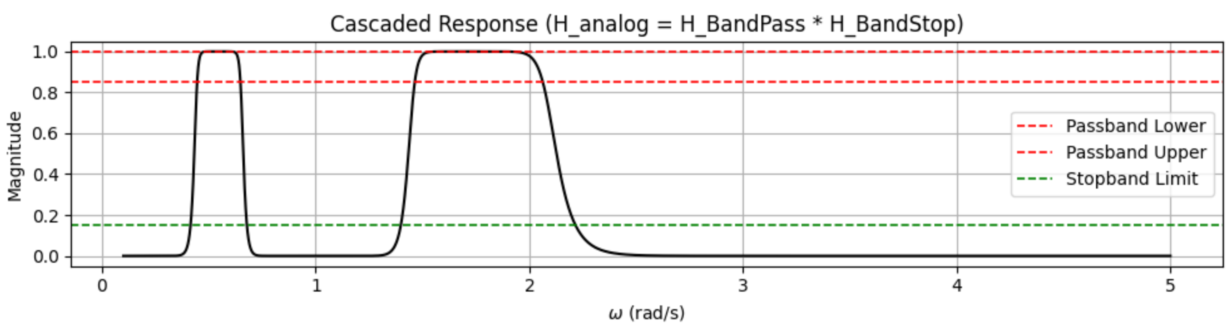


Figure 5: Overall Transfer Function of the Analog Filters.

4 Multi-Band Pass Digital Filter Response

Lastly, we implemented the Multi-Band Pass Digital Filter, and below attached is the digital frequency response of the system

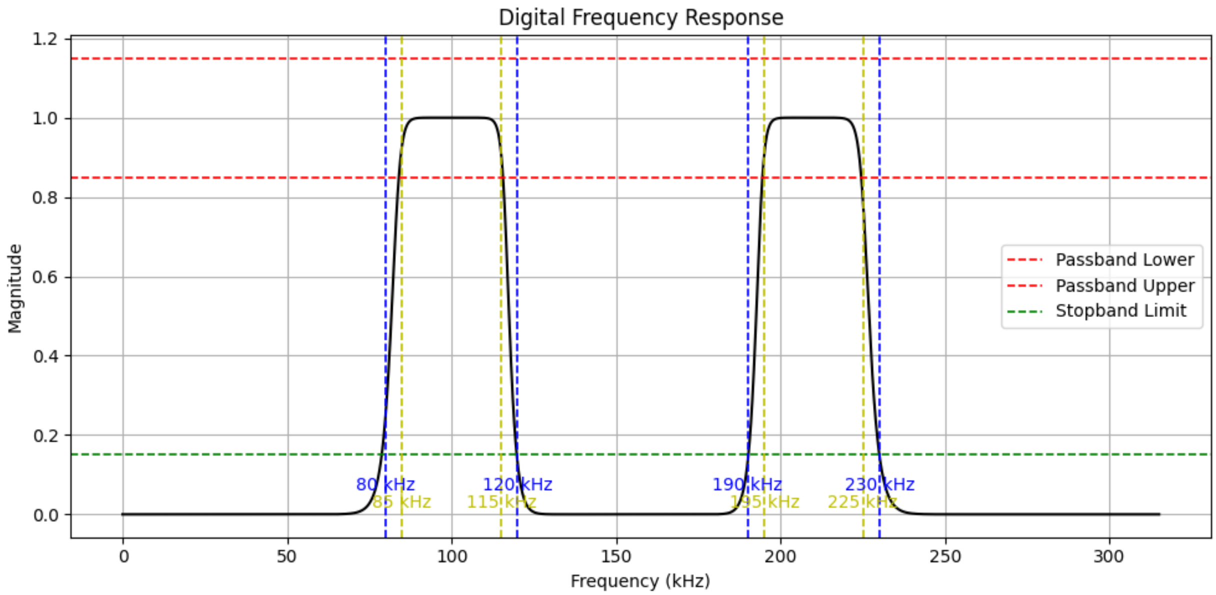


Figure 6: Multi-Band Pass Filter Response

5 Code for Plot Generation

The following section provides the code used to generate the plots in Python.

5.1 Python Code for defining the filter type

```
import numpy as np
import matplotlib.pyplot as plt

class BandPassFilter:
    """Class for designing a BandPass filter and computing its poles and frequency response."""

    def __init__(self, N, omega_C, omega_0__2, BandWidth):
        self.N = N
        self.omega_C = omega_C
        self.omega_0__2 = omega_0__2
        self.BandWidth = BandWidth
        self.poles = self.compute_poles()

    def compute_poles(self):
        """Compute the bandpass filter poles."""
        k = np.arange(self.N)
        poles_lowpass = self.omega_C * np.exp(1j * (np.pi/2 + (2*k + 1)*np.pi/(2*self.N)))

        poles = np.zeros(2 * self.N, dtype=complex)
        for i, p in enumerate(poles_lowpass):
            sqrt_term = np.lib.scimath.sqrt((self.BandWidth * p)**2 + 4 * self.omega_0__2)
            poles[2 * i] = (self.BandWidth * p + sqrt_term) / 2.0
            poles[2 * i + 1] = (self.BandWidth * p - sqrt_term) / 2.0

        return poles

    def butterworth_response(self, omega):
        """Compute the Butterworth magnitude response for the bandpass filter."""
        omegaL = (omega**2 - self.omega_0__2) / (omega * self.BandWidth)
        return 1 / np.sqrt(1 + (omegaL / self.omega_C) ** (2 * self.N))

    def plot_poles(self):
        """Plot the poles of the bandpass filter."""
        plt.scatter(np.real(self.poles), np.imag(self.poles), color='blue', marker='x', label='BandPass Filter Poles')

class BandStopFilter:
    """Class for designing a BandStop filter and computing its poles and frequency response."""

    def __init__(self, N, omega_C, omega_0__2, BandWidth):
        self.N = N
        self.omega_C = omega_C
        self.omega_0__2 = omega_0__2
        self.BandWidth = BandWidth
        self.poles = self.compute_poles()

    def compute_poles(self):
        """Compute the bandstop filter poles."""
        k = np.arange(self.N)
        poles_lowpass = self.omega_C * np.exp(1j * (np.pi/2 + (2*k + 1)*np.pi/(2*self.N)))

        poles = np.zeros(2 * self.N, dtype=complex)
        epsilon = 1e-9  # Small value to prevent division by zero

        for i, p in enumerate(poles_lowpass):
            sqrt_term = np.lib.scimath.sqrt(self.BandWidth**2 + 4 * self.omega_0__2 * (p**2))
            poles[2 * i] = (-self.BandWidth + sqrt_term) / (2 * (p + epsilon))
```

```python
            poles[2 * i + 1] = (-self.BandWidth - sqrt_term) / (2 * (p + epsilon))

        return poles

    def butterworth_response(self, omega):
        """Compute the Butterworth magnitude response for the bandstop filter."""
        omegaL = (omega * self.BandWidth) / (self.omega_0__2 - omega**2)
        return 1 / np.sqrt(1 + (omegaL / self.omega_C) ** (2 * self.N))

    def plot_poles(self):
        """Plot the poles of the bandstop filter."""
        plt.scatter(np.real(self.poles), np.imag(self.poles), color='magenta', marker='o',
    label='BandStop Filter Poles')
```

Listing 1: Filter Type Definition

5.2 Plotting the Poles

```python
class FilterSystem:
    """Class that combines both filters into a single system."""

    def __init__(self, bandpass_filter, bandstop_filter):
        self.bandpass_filter = bandpass_filter
        self.bandstop_filter = bandstop_filter
        self.poles_combined = np.concatenate((bandpass_filter.poles, bandstop_filter.poles))

    def plot_poles(self):
        """Plot the poles of the combined filter system."""
        plt.figure(figsize=(8, 6))
        bandpass.plot_poles()
        plt.axhline(0, color='black', linewidth=0.5)
        plt.axvline(0, color='black', linewidth=0.5)
        plt.title('Analog Poles for BandPass Filter')
        plt.xlabel('Real(s)')
        plt.ylabel('Imag(s)')
        plt.grid(True)
        plt.legend()
        plt.show()

        plt.figure(figsize=(8, 6))
        bandstop.plot_poles()
        plt.axhline(0, color='black', linewidth=0.5)
        plt.axvline(0, color='black', linewidth=0.5)
        plt.title('Analog Poles for BandStop Filter')
        plt.xlabel('Real(s)')
        plt.ylabel('Imag(s)')
        plt.grid(True)
        plt.legend()
        plt.show()

        plt.figure(figsize=(8, 6))
        self.bandpass_filter.plot_poles()
        self.bandstop_filter.plot_poles()
        plt.axhline(0, color='black', linewidth=0.5)
        plt.axvline(0, color='black', linewidth=0.5)
        plt.title('Combined Analog Filter Poles')
        plt.xlabel('Real(s)')
        plt.ylabel('Imag(s)')
        plt.grid(True)
        plt.legend()
        plt.show()

bandpass = BandPassFilter(N=22, omega_C=1.04, omega_0__2=0.918, BandWidth=1.59)
bandstop = BandStopFilter(N=19, omega_C=1.037, omega_0__2=0.6375, BandWidth=0.74)
filter_system.plot_poles()
```

Listing 2: Plotting Poles for Filters

5.3 Plotting the Analog Response

```python
class FilterAnalysis:
    """Class to analyze and plot the combined filter system."""

    def __init__(self, bandpass, bandstop, omega_range):
        self.bandpass = bandpass
        self.bandstop = bandstop
        self.omega = omega_range
        self.H1 = self.bandpass.butterworth_response(self.omega)
        self.H2 = self.bandstop.butterworth_response(self.omega)
        self.H_analog = self.H1 * self.H2

    def plot_response(self, passband_lower, passband_upper, stopband_limit):
        """Plot the filter responses."""
        plt.figure(figsize=(10, 8))

        plt.subplot(3, 1, 1)
        plt.plot(self.omega, self.H1, 'b', linewidth=1.5)
        plt.title('Response of Bandpass Filter')
        plt.xlabel(r'$\omega$ (rad/s)')
        plt.ylabel('Magnitude')
        plt.grid(True)

        plt.subplot(3, 1, 2)
        plt.plot(self.omega, self.H2, 'm', linewidth=1.5)
        plt.title('Response of Bandstop Filter')
        plt.xlabel(r'$\omega$ (rad/s)')
        plt.ylabel('Magnitude')
        plt.grid(True)

        plt.subplot(3, 1, 3)
        plt.plot(self.omega, self.H_analog, 'k', linewidth=1.5)
        plt.title('Cascaded Response (H_analog = H_BandPass * H_BandStop)')
        plt.xlabel(r'$\omega$ (rad/s)')
        plt.ylabel('Magnitude')
        plt.grid(True)

        plt.axhline(passband_lower, color='r', linestyle='--', linewidth=1.2, label='Passband Lower')
        plt.axhline(passband_upper, color='r', linestyle='--', linewidth=1.2, label='Passband Upper')
        plt.axhline(stopband_limit, color='g', linestyle='--', linewidth=1.2, label='Stopband Limit')
        plt.legend()

        plt.tight_layout()
        plt.show()

omega = np.linspace(0.1, 5, 10000)
bandpass = BandPassFilter(22, 1.04, 0.918, 1.59)
bandstop = BandStopFilter(19, 1.037, 0.94815, 0.825)
analysis = FilterAnalysis(bandpass, bandstop, omega)
analysis.plot_response(0.85, 1, 0.15)
```

Listing 3: Plot of Analog Response

5.4 Digital Multi-Band Filter Response

```python
class DigitalFilterAnalysis:
    def __init__(self, omega_range, fs):
        self.omega = omega_range
        self.fs = fs  # Sampling frequency
        self.fNyq = fs / 2.0  # Nyquist frequency
        self.w = np.linspace(0, np.pi, 10000)
        self.Omega = np.tan(self.w / 2)
```

```python

    def butterworth_response(self, omegaC, N, omega0_sq, BandWidth, filter_type):
        """Compute the Butterworth magnitude response."""
        if filter_type == "bandpass":
            omegaL = (self.omega**2 - omega0_sq) / (self.omega * BandWidth)
        elif filter_type == "bandstop":
            omegaL = (self.omega * BandWidth) / (omega0_sq - self.omega**2)
        return 1 / np.sqrt(1 + (omegaL / omegaC) ** (2 * N))

    def digital_response(self, omegaC, N, omega0_sq, BandWidth, filter_type):
        """Compute the digital Butterworth magnitude response using direct substitution."""
        Omega_safe = np.where(np.abs(self.Omega) > 1e-12, self.Omega, 1e-12)
        if filter_type == "bandpass":
            omegaL_sub = (Omega_safe**2 - omega0_sq) / (Omega_safe * BandWidth)
        elif filter_type == "bandstop":
            omegaL_sub = (Omega_safe * BandWidth) / (omega0_sq - Omega_safe**2)
        omegaL_sub = np.clip(omegaL_sub, -1e6, 1e6)
        return 1 / np.sqrt(1 + (omegaL_sub / omegaC) ** (2 * N))

    def analyze_and_plot(self, passband_lower, passband_upper, stopband_limit):
        # Compute Digital Responses
        H1_digital = self.digital_response(1.04, 22, 0.918, 1.59, "bandpass")
        H2_digital = self.digital_response(1.047, 19, 0.94815, 0.825, "bandstop")
        H_digital = H1_digital * H2_digital

        # Convert normalized digital frequency to kHz
        f_axis_khz = (self.w / np.pi) * self.fNyq / 1e3

        # Plot Responses
        plt.figure(figsize=(10, 5))
        plt.plot(f_axis_khz, H_digital, 'k', linewidth=1.5)
        plt.title('Digital Frequency Response')
        plt.xlabel('Frequency (kHz)')
        plt.ylabel('Magnitude')
        plt.grid(True)

        # Specification Lines
        plt.axhline(passband_lower, color='r', linestyle='--', linewidth=1.2, label='Passband Lower')
        plt.axhline(passband_upper, color='r', linestyle='--', linewidth=1.2, label='Passband Upper')
        plt.axhline(stopband_limit, color='g', linestyle='--', linewidth=1.2, label='Stopband Limit')

        # Mark Frequencies
        frequencies_to_mark = [80, 120, 190, 230]  # in kHz
        for f in frequencies_to_mark:
            plt.axvline(x=f, color='b', linestyle='--', linewidth=1.2)
            plt.text(f, 0.05, f'{f} kHz', color='b', ha='center', va='bottom', fontsize=10)

        frequencies_to_mark = [85, 115, 195, 225]  # in kHz
        for f in frequencies_to_mark:
            plt.axvline(x=f, color='y', linestyle='--', linewidth=1.2)
            plt.text(f, 0.05, f'{f} kHz', color='y', ha='center', va='top', fontsize=10)

        plt.legend()
        plt.tight_layout()
        plt.show()

fs = 630e3  # Sampling frequency
omega = np.linspace(0.1, 5, 10000)  # Frequency range
analysis = DigitalFilterAnalysis(omega, fs)
analysis.analyze_and_plot(0.85, 1.15, 0.15)
```

Listing 4: Digital Multi-Band Filter Response

6 Peer Review

Name of student: Anupam Rawat
Name and Roll Number of the reviewer : Jatin Kumar 22B3922
Group Number : 34
Review Comments :
I have reviewed the filter design assignment of Anupam Rawat, Roll Number 22B3982. The filter number assigned to him is 104. Following are my comments on his assignment:
He has correctly implemented the filter, and the response are in accordance to the expected frequency response. He has also correctly used the butterworth approximations to implement the IIR designs for each filter. He has included all the code, results and their plots in the report.

Acknowledgements

I would like to express my sincere gratitude to **Professor V. M. Gadre** for introducing and teaching me the fundamental and profoundly important course on *Digital Signal Processing*. The structured tasks and exercises in this course have greatly enhanced my understanding of various concepts in this field.
Furthermore, I am deeply thankful to my group members, **Mr. Jatin Kumar** and **Mr. Rishabh Bhardwaj**, for their unwavering support, insightful discussions, and seamless collaboration throughout this course. Their contributions have been invaluable in refining my learning experience.
I truly appreciate the guidance and teamwork that is making this journey both enriching and intellectually stimulating.