TakeHome Endsemester Assignment Submission: EE 338, Spring 2024-25

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

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

April 08, 2025

1 Aim Of the Assignment

We are required to design an Finite-Impulse-Response (FIR) Filter, with bands from both Group I and Group II, acting as **passbands**. Since, the filter is FIR Multi-Band Pass Filter, we have to use a Kaiser Window.

2 Specifications Required

The analog signal is bandlimited to 280kHz and it's ideally sampled with a sampling rate of 630kHz. Now 2 x bandwidth < sampling rate. The sampling rate obeys the Nyquist criteria and can be reconstructed without any loss.

1. The tolerance of stopband and passband are $\delta = 0.15$ in magnitude.
2. The Filter number assigned to me is: M = 104
   (a) Q = floor (M / 11) = 9
   (b) R = M mod 11 = 5
3. Range of Group I frequency: (40 + 5D) to (70 + 5D); where D = Q
   (a) **Lower Edge** = 40 + 5D = 40 + 45 = **85kHz**
   (b) **Upper Edge** = 70 + 5D = 70 + 45 = **115kHz**
4. Range of Group II frequency: (170 + 5D) to (200 + 5D); where D = R
   (a) **Lower Edge** = 170 + 5D = 170 + 25 = **195kHz**
   (b) **Upper Edge** = 200 + 5D = 200 + 25 = **225kHz**
5. The transition band on either side of the passband is 5kHz. Group I and II are the passbands.

3 Desired Frequency Response

As per the above specifications, below is the plot of the desired frequency specifications.

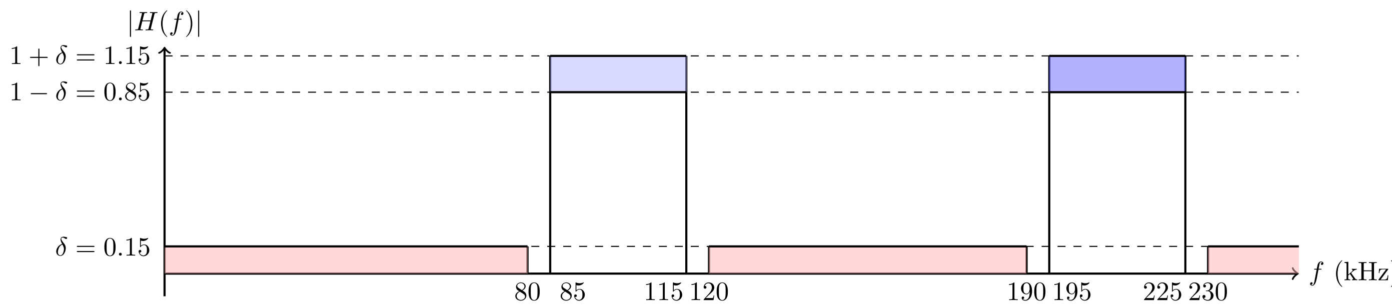


Figure 1: Desired Frequency Response

The light blue region, on the left, corresponds to the Group I of frequency bands, while the slightly darker region on the right, corresponds to the Group II of frequency bands.

3.1 Realization of the Frequency Response

The filter response can be realised by cascading a BandPass Filter with a BandPass Filter *(Series Connection)* OR by adding the result obtained by two BandPass Filters *(Parallel Connection)*. For the purpose of this assignment, we will be using the Parallel Addition option.
Each of the BandPass Filters, can be realised by subtraction of two Low pass filters with Kaiser Windows.

3.2 Realization of Group I Frequency Bands

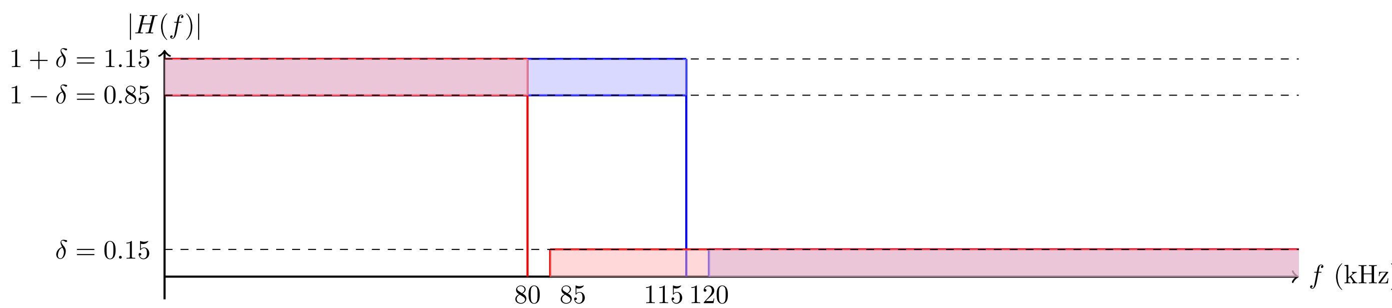


Figure 2: Realization of Group I Frequency Response using the difference of Two Low Pass Filters

The Blue filter represents the Low Pass Filter I with frequency cutoff at from 115kHz, and 5kHz transition band, across the stopband. The red filter represents the Low Pass Filter II with frequency cutoff at 80kHz, with a transition band of 5kHz across the passband.

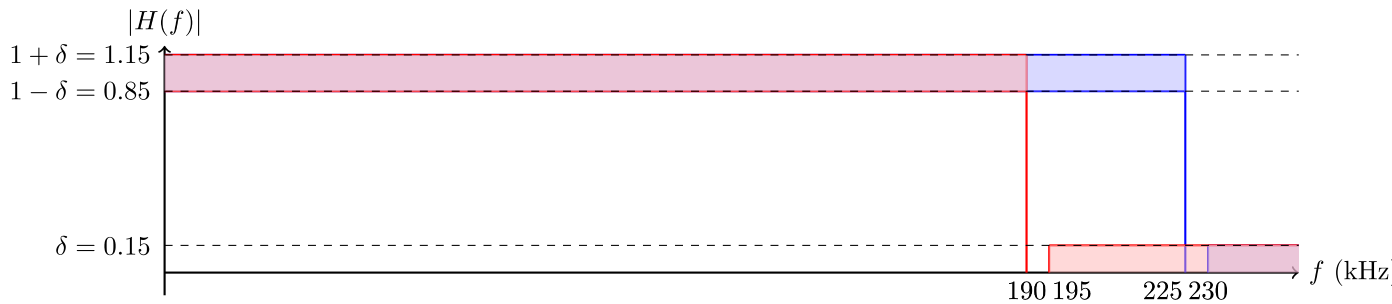


Figure 3: Realization of Group II Frequency Response using the difference of Two Low Pass Filters

The Blue filter represents the Low Pass Filter I with frequency cutoff at from 225kHz, and 5kHz transition band, across the stopband. The red filter represents the Low Pass Filter II with frequency cutoff at 190kHz, with a transition band of 5kHz across the passband.

Since, we're now adding up the filter responses (or their negatives) 4 times, so now the tolerance has to be one-fourth of the originally required tolerance. Thus the tolerance for each Filter becomes 0.15 / 4 = 0.0375

4 Frequency Values

4.1 Normalizing the Frequency

To normalize the frequency, so that the range of frequencies lie between -π and +π. We can use the below formula:

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

4.2 Converting the values to analog

We can convert these values by using the below formula:

$$\Omega = tan(\frac{\omega}{2})$$

4.3 Tabulated Values

Below is the table to convert between the formulas:

f(kHz)	80	85	115	120	190	195	225	230
Normalized Frequency (ω)	0.25π	0.27π	0.365π	0.38π	0.60π	0.62π	0.71π	0.73π
Analog Equivalent (Ω)	0.41	0.45	0.645	0.68	1.38	1.47	2.04	2.21

Table 1: Frequency values and their corresponding normalized and analog equivalents

5 Low Pass Filter

5.1 Common Kaiser Window Parameters

5.1.1 Transition Band

The transition width is given to be $\Delta f = 5\,\text{kHz}$. Thus, the normalized transition width $\Delta\omega_T$ is:

$$\Delta\omega_T = \frac{\Delta f \cdot 2\pi}{f_s} = \frac{5\,\text{kHz} \cdot 2\pi}{630\,\text{Hz}} = 0.0158\pi$$

5.1.2 Filter Order (N)

Substituting the values of attenuation and transition width into the formula for the filter order of a Kaiser window:

$$N_{\min} \geq \frac{A - 7.95}{2 \times 2.285 \times \Delta\omega_T}$$

$$N_{\min} \geq \frac{28.51 - 7.95}{2 \times 2.285 \times 0.0158\pi} = 90.635$$

Thus, the minimum value of N is taken as 91, and therefore the minimum window length is:

$$L = 2 \times N_{\min} + 1 = 2 \times 91 + 1 = 183$$

5.1.3 Attenuation (A)

The attenuation A is computed from the minimum of passband and stopband ripples δ using:

$$\text{For } \delta = 0.0375, \quad A = -20\log_{10}(0.0375) \approx 28.51\,\text{dB}$$

5.1.4 Calculating β

The value of β is determined based on A:

$$\beta = \begin{cases} 0 & \text{if } A < 21 \\ 0.5842(A-21)^{0.4} + 0.07886(A-21) & \text{if } 21 \leq A < 50 \\ 0.1102(A-8.7) & \text{if } A \geq 50 \end{cases}$$

$$\beta = 0.5842(28.51-21)^{0.4} + 0.07886(28.51-21) \approx 1.90$$

5.1.5 Ideal Impulse Response of Bandpass Filter

The ideal impulse response $h_d[n]$ is derived by superimposing sinc responses:

$$h_d[n] = \left(\frac{\sin(\omega_{c1}n)}{\pi n}\right) - \left(\frac{\sin(\omega_{c2}n)}{\pi n}\right) + \left(\frac{\sin(\omega_{c3}n)}{\pi n}\right) - \left(\frac{\sin(\omega_{c4}n)}{\pi n}\right), \quad n \neq 0$$

$$h_d[0] = \frac{\omega_{c1}}{\pi} - \frac{\omega_{c2}}{\pi} + \frac{\omega_{c3}}{\pi} - \frac{\omega_{c4}}{\pi}$$

5.1.6 Kaiser Window Function

The Kaiser window is given by:

$$w[n] = \frac{I_0\left(\beta\sqrt{1 - \left(\frac{n-N}{N}\right)^2}\right)}{I_0(\beta)}, \quad -N \leq n \leq N$$

where I_0 is the zeroth-order modified Bessel function of the first kind:

$$I_0(x) = \sum_{k=0}^{\infty} \frac{1}{(k!)^2}\left(\frac{x}{2}\right)^{2k}$$

5.1.7 Final FIR Filter Coefficients

The final filter coefficients are the product of the ideal response and the Kaiser window:

$$h[n] = h_d[n] \cdot w[n]$$

5.1.8 DTFT of Filter

The Discrete-Time Fourier Transform (DTFT) of the FIR filter is:

$$H(e^{j\omega}) = \sum_{n=-N}^{N} h[n] \cdot e^{-j\omega n}$$

5.1.9 Z-domain Transfer Function

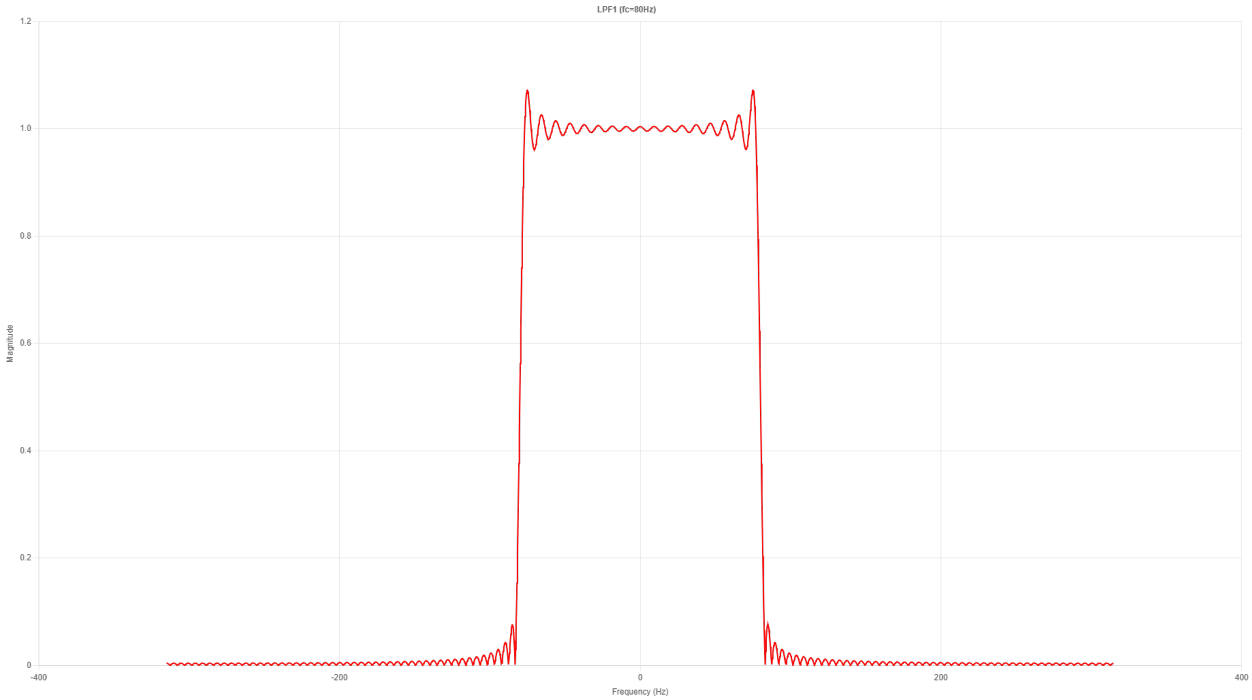
The Z-transform of the FIR filter is:

$$H(z) = \sum_{n=0}^{2N} h[n] \cdot z^{-n}$$

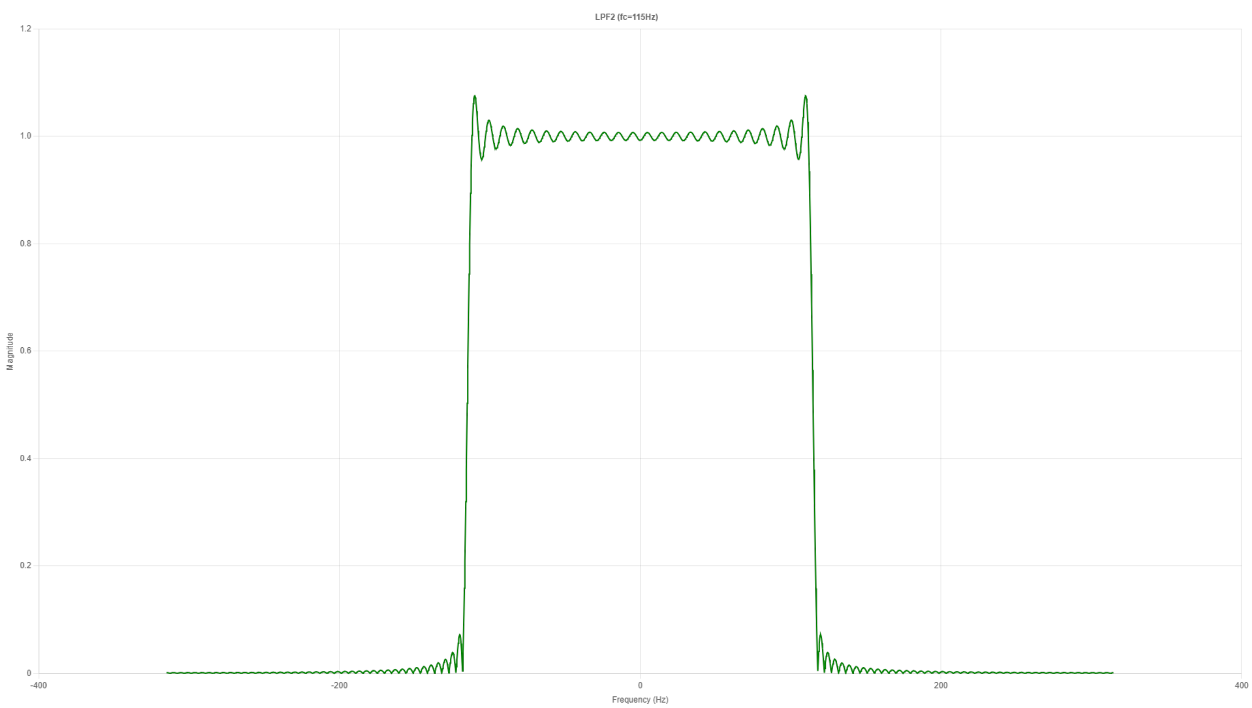
5.2 Low Pass Filters

We are using a total of four Low Pass Filters, with the cutoff frequencies as below:

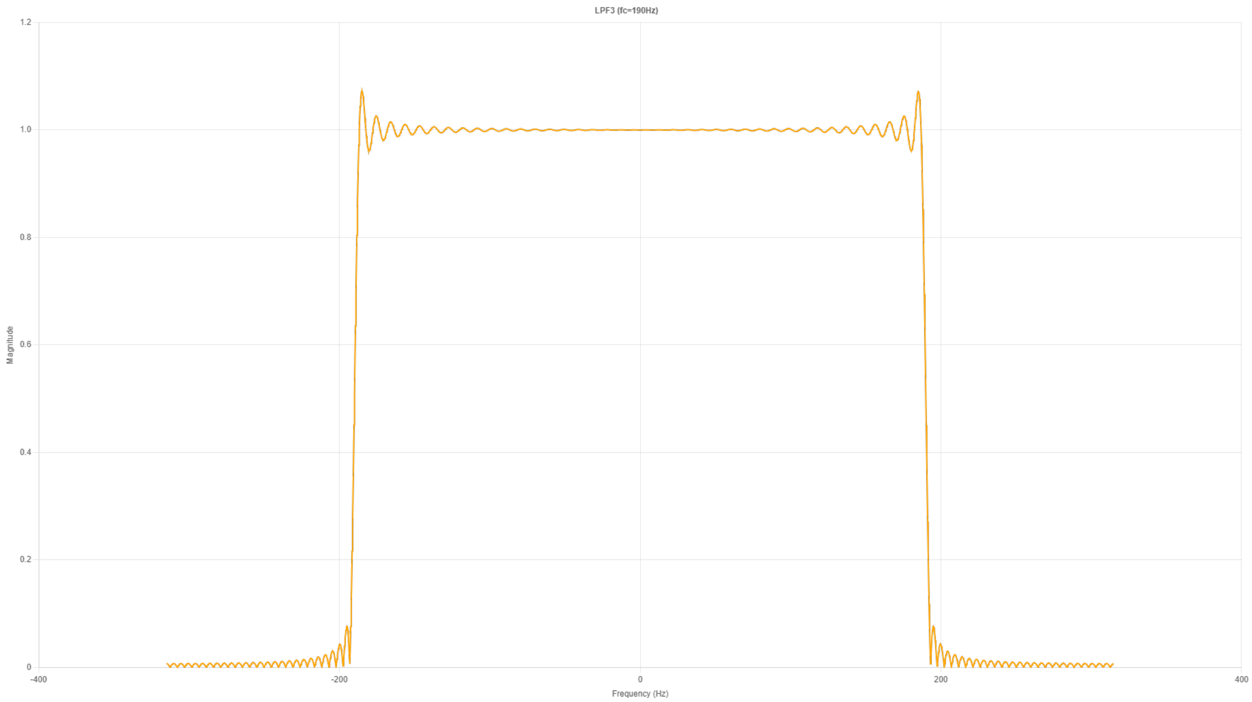
- LPF1: 80Hz



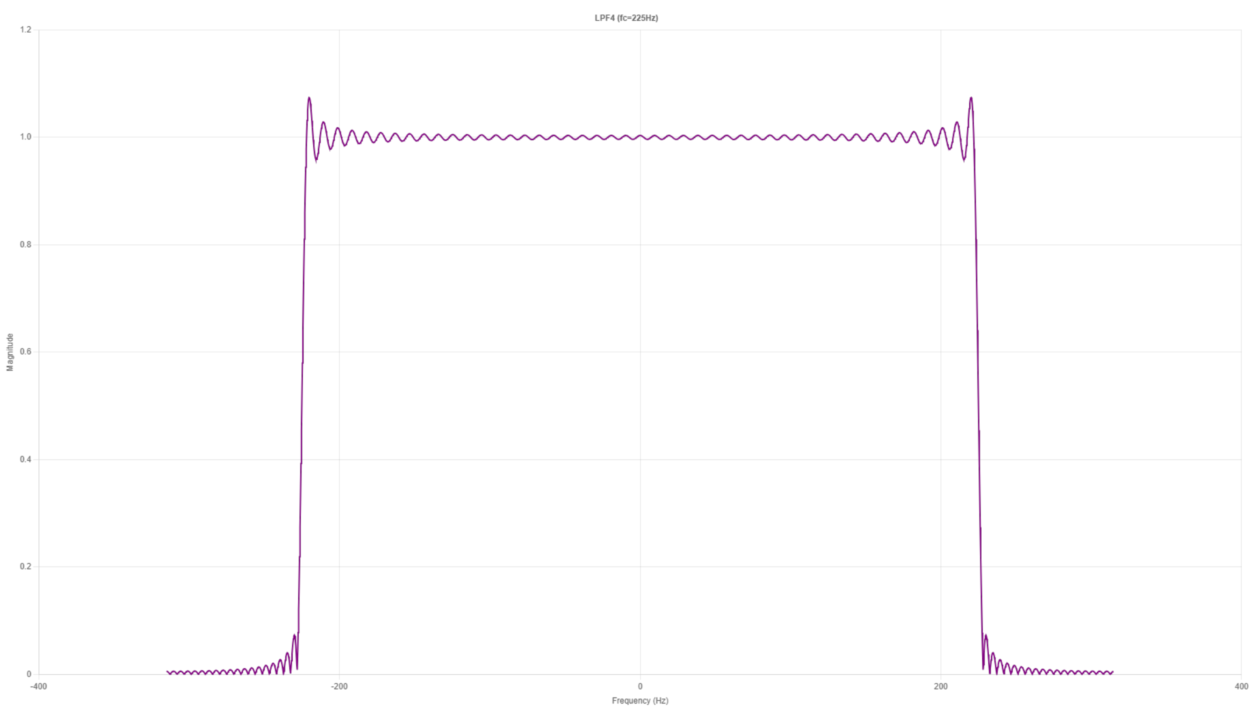
- LPF2: 115Hz



- LPF3: 190Hz



- LPF4: 225Hz



6 Combined Filter Response

To obtain the combined filter response we carry out the below operation:

LPF2 - LPF1 + LPF4 - LPF3

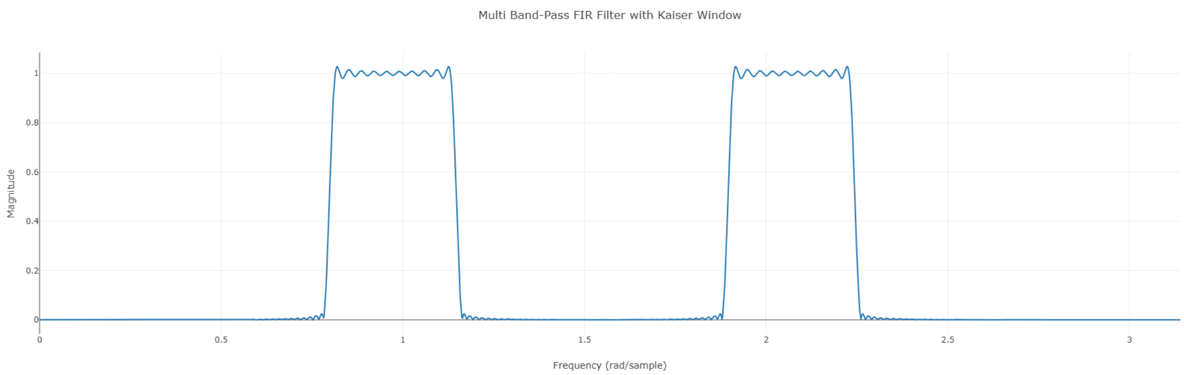


Figure 4: Multi-Band Pass Filter Response from 0 to π

7 Code for Plots

7.1 JavaScript Code for Low Pass Filter Plots

```
class FIRLowPassFilter {
  constructor(fs, fc, N, beta, deltaOmegaT, numPoints) {
    this.fs = fs;
    this.fc = fc;
    this.N = N;
    this.numTaps = 2 * N + 1;
    this.beta = beta;
    this.deltaOmegaT = deltaOmegaT;
    this.n = Array.from({ length: 2 * N + 1 }, (_, i) => i - N);
    this.numPoints = numPoints;
  }

  i0(x, numTerms = 50) {
    let result = 0;
    for (let n = 0; n < numTerms; n++) {
      let term = Math.pow(x / 2, 2 * n) / this.factorial(n);
      result += term;
    }
    return result;
  }

  factorial(n) {
    if (n === 0 || n === 1) return 1;
    let res = 1;
    for (let i = 2; i <= n; i++) res *= i;
    return res;
  }

  get_H_FIR() {
    let alpha = this.N;
    let temp_arr = new Array(this.numTaps).fill(1);
    for (let i = -this.N; i <= this.N; i++) {
      let idx = i + this.N;
      if (i !== 0) {
        let sinc = Math.sin(i * (this.fc / 315) * Math.PI) / (i * Math.PI);
        let window = this.i0(this.beta * Math.sqrt(1 - Math.pow(i / alpha, 2))) / this.i0(
  this.beta);
        temp_arr[idx] *= window * sinc;
      } else {
        temp_arr[idx] *= (this.fc / 315);
      }
    }
    return temp_arr;
  }

  DTFT(X_vals, time_series) {
    let dtft_arr = [];
    for (let omega of X_vals) {
      let sumReal = 0;
      let sumImag = 0;
      for (let k = -this.N; k <= this.N; k++) {
        let idx = k + this.N;
        let angle = omega * k;
        sumReal += time_series[idx] * Math.cos(angle);
        sumImag -= time_series[idx] * Math.sin(angle);
      }
      dtft_arr.push(Math.sqrt(sumReal * sumReal + sumImag * sumImag));
    }
    return dtft_arr;
  }

  plot_frequency_response() {
    const h = this.get_H_FIR();
```

```
    const X_vals = Array.from({ length: this.numPoints }, (_, i) => -Math.PI + 2 * Math.PI * i / this.numPoints);
    const dtft = this.DTFT(X_vals, h);
    return { X_vals, dtft };
  }
}
```

Listing 1: Filter Type Definition

7.2 JavaScript Code for Combined Filter Response

```
class CombinedResponse {
  constructor(fs = 630, N = 65, beta = null, delta_omega = 0.0158 * Math.PI) {
    this.fs = fs;
    this.N = N;
    this.numtaps = 2 * N + 1;
    this.delta_omega = delta_omega;
    this.n = Array.from({ length: 2 * N + 1 }, (_, i) => i - N);
    this.beta = beta !== null ? beta : 0.80 / N;
    this.h = this._get_H_FIR();
  }

  static compute_A(delta) {
    return -20 * Math.log10(delta);
  }

  static compute_beta(A) {
    if (A <= 21) return 0;
    else if (A <= 50)
      return 0.5842 * Math.pow(A - 21, 0.4) + 0.07886 * (A - 21);
    else
      return 0.1102 * (A - 8.7);
  }

  static compute_N(A, delta_omega) {
    return Math.ceil((A - 8) / (2.285 * delta_omega));
  }

  _i0(x, num_terms = 50) {
    let result = 0;
    for (let n = 0; n < num_terms; n++) {
      const term = Math.pow(x / 2, 2 * n) / this._factorial(n);
      result += term;
    }
    return result;
  }

  _factorial(n) {
    if (n === 0 || n === 1) return 1;
    let result = 1;
    for (let i = 2; i <= n; i++) result *= i;
    return result;
  }

  _get_H_FIR() {
    const alpha = this.N;
    const temp_arr = new Array(this.numtaps).fill(1);
    for (let i = -this.N; i <= this.N; i++) {
      const idx = i + this.N;
      if (i !== 0) {
        const bessel_ratio = this._i0(this.beta * Math.sqrt(1 - Math.pow(i / alpha, 2))) / this._i0(this.beta);
        const sinc_combination =
          (Math.sin(i * (115 / 315) * Math.PI) / (i * Math.PI)) -
          (Math.sin(i * (80 / 315) * Math.PI) / (i * Math.PI)) +
          (Math.sin(i * (225 / 315) * Math.PI) / (i * Math.PI)) -
```

```
        (Math.sin(i * (190 / 315) * Math.PI) / (i * Math.PI));
      temp_arr[idx] *= bessel_ratio * sinc_combination;
    } else {
      temp_arr[idx] *= ((115 - 80 + 225 - 190) / 315);
    }
  }
  return temp_arr;
}

compute_DTFT(X_vals) {
  const dtft_arr = [];
  for (const omega of X_vals) {
    let sum_val_real = 0;
    let sum_val_imag = 0;
    for (let k = -this.N; k <= this.N; k++) {
      const h_k = this.h[k + this.N];
      sum_val_real += h_k * Math.cos(omega * k);
      sum_val_imag -= h_k * Math.sin(omega * k);
    }
    dtft_arr.push(Math.sqrt(sum_val_real ** 2 + sum_val_imag ** 2));
  }
  return dtft_arr;
}

plot_response() {
  // const X_vals = Array.from({ length: 2000 }, (_, i) => -MathPI + (2 * Math.PI * i) /
  2000);
  const X_vals = Array.from({ length: 2000 }, (_, i) => (Math.PI * i) / 2000);

  const dtft_vals = this.compute_DTFT(X_vals);

  const trace = {
    x: X_vals,
    y: dtft_vals,
    mode: 'lines',
    name: 'Magnitude Response',
  };

  const layout = {
    title: 'Multi Band-Pass FIR Filter with Kaiser Window',
    xaxis: { title: 'Frequency (rad/sample)' },
    yaxis: { title: 'Magnitude' },
  };

  Plotly.newPlot('plot', [trace], layout);
}
}

// Example usage:
const delta = 0.0375;
const delta_omega = 0.0158 * Math.PI;

const A = CombinedResponse.compute_A(delta);
const beta = CombinedResponse.compute_beta(A);
const N = CombinedResponse.compute_N(A, delta_omega);

console.log(`A = ${A.toFixed(2)} dB`);
console.log(`Beta = ${beta.toFixed(4)}`);
console.log(`N = ${N}`);

const cr = new CombinedResponse(undefined, N, beta, delta_omega);
cr.plot_response();
```

Listing 2: Filter Type Definition

8 Comparison with Previous Filter Designs

Three filter types are considered that achieved the same goal:

- **Filter Type 1**: Butterworth Bandpass (Order 22) and Bandstop (Order 19).
- **Filter Type 2**: Chebyshev Bandpass (Order 7) and Bandstop (Order 7).
- **Filter Type 3**: FIR Kaiser Window Filter (Order 91)

Feature	**FIR with Kaiser Window**
Phase Response	**Linear Phase**
Stability	Always stable
Computational Cost	Higher (requires more taps)
Passband Response	Control over ripple using the window shape
Stopband Attenuation	Can be controlled via window shape
Transition Band	Adjustable (depends on window length)
Filter Order	Higher order required for sharper transitions
Filter Design Complexity	Complex (needs window parameters and coefficients)

Feature	**IIR with Butterworth**
Phase Response	Non-linear phase, introduces phase distortion
Stability	Stable (as long as designed correctly)
Computational Cost	Lower (more efficient than FIR)
Passband Response	Flat passband, smooth response
Stopband Attenuation	Smooth roll-off, not as sharp as Chebyshev
Transition Band	Gradual roll-off
Filter Order	Lower order for similar performance to FIR
Filter Design Complexity	Simpler, but requires pole-zero placement

Feature	**IIR with Chebyshev**
Phase Response	Non-linear phase, introduces phase distortion
Stability	Stable (as long as designed correctly)
Computational Cost	Lower (more efficient than FIR)
Passband Response	Ripple in the passband (Type I), or stopband (Type II)
Stopband Attenuation	Steep roll-off, better stopband attenuation (Type I)
Transition Band	Steeper roll-off than Butterworth
Filter Order	Lower order for sharper roll-off than Butterworth
Filter Design Complexity	Similar complexity to Butterworth, but with ripple control

Table 2: Comparison of Multi-band Pass Filter Design Methods (Vertical Version)

9 Conclusion

Each filter has its own advantages. The below points, help us choose the filter base on the use-case:

- FIR with Kaiser Window if linear phase is essential, especially in applications where phase integrity is critical.
- IIR with Butterworth if you need a smooth, flat passband and want a filter that is computationally efficient and stable.
- IIR with Chebyshev if you need a sharp roll-off and can accept passband or stopband ripple for the sake of more aggressive filtering.

10 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 FIR and required Kaiser Window approximations to implement the 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.