THE UNIVERSITY OF DANANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
Faculty of Advanced Science and Technology

Digital Signal Processing

Midterm Examination Report

Instructor : TS. Hồ Phước Tiến
Class : 21ES
Group members : Lê Quốc Duy - 123210016
Trần Vũ Hồng Phúc - 123210021

Da Nang, 27th March 2025

We will design a high-pass filter with the following specifications:

$$\varpi_s = 0.6\pi, \varpi_p = 0.75\pi, R_p = 1 \text{ dB}, A_s = 40 \text{ dB}$$

Problem 1: Using the window technique, design the above filter.

a) Which window do you choose? Explain.

- We choose the **Kaiser window** because:
    - It provides **adjustable sidelobe attenuation** through the β parameter.

      Algorithms

      The coefficients of a Kaiser window are computed from the following equation:

      $$w(n) = \frac{I_0\left(\beta\sqrt{1-\left(\frac{n-N/2}{N/2}\right)^2}\right)}{I_0(\beta)}, \quad 0 \le n \le N,$$

      where I_0 is the zeroth-order modified Bessel function of the first kind. The length $L = N + 1$. `kaiser(L,beta)` is equivalent to

      ```
      besseli(0,beta*sqrt(1-(((0:L-1)-(L-1)/2)/((L-1)/2)).^2))/besseli(0,beta)
      ```

      To obtain a Kaiser window that represents an FIR filter with sidelobe attenuation of α dB, use the following β.

      $$\beta = \begin{cases} 0.1102(\alpha - 8.7), & \alpha > 50 \\ 0.5842(\alpha - 21)^{0.4} + 0.07886(\alpha - 21), & 50 \ge \alpha \ge 21 \\ 0, & \alpha < 21 \end{cases}$$

    - It offers better **control over transition width**, making it suitable for meeting stopband attenuation (`As`) and passband ripple (`Rp`) requirements.
    - Compared to other windows like Hamming, it results in **shorter filter lengths** while still meeting the design criteria.

b) What is the length of the resulting filter?

- The filter order (`N`) is computed using the **empirical formula**:

$$N = \frac{A_s - 8}{2.285 \times \Delta\omega}$$

where `Δω = wp - ws` is the transition width. The result is displayed in the **command window**.

```
Estimated filter order N = 31
Kaiser beta parameter = 3.3953
Actual stopband attenuation: 39.42 dB (Target: 40 dB)
Actual passband ripple: 0.15 dB (Target: 1 dB)
Filter Order (N): 31
First 5 and last 5 coefficients:
   -0.0012    0.0047   -0.0043   -0.0028    0.0118
    0.0118   -0.0028   -0.0043    0.0047   -0.0012
```

Figure 1: Calculation basic parameters

- With above formula and use the following specification, we can calculate N is 31 (as seen in the above image)

c) Does the resulting filter meet the above requirements? Explain.

- Yes, it does. Because:
  - ➔ **Passband Ripple (`Rp_actual`)** is calculated from the peak-to-peak variation in the passband.
  - ➔ **Stopband Attenuation (`As_actual`)** is measured from the highest sidelobe level in the stopband.

  If `Rp_actual < 1 dB` and `As_actual ≤ 40 dB`, the filter meets the requirements. In our design Rp_actual < 1 dB (0.15 db < 1 dB) and `As_actual ≤ 40 dB`

d) Code and Result:

➔ MATLAB:

```
clc; clear; close all;
% Given filter specifications
omega_s = 0.6 * pi;  % Stopband edge frequency (rad/sample)
omega_p = 0.75 * pi; % Passband edge frequency (rad/sample)
A_s = 40;  % Stopband attenuation in dB
R_p = 1;   % Passband ripple in dB
% Normalized frequencies (0 to 1, where 1 is Nyquist frequency)
f_s = omega_s / pi; % Normalized stopband edge frequency
f_p = omega_p / pi; % Normalized passband edge frequency
% Compute transition bandwidth
bw = f_p - f_s;
% Compute Kaiser window beta parameter
if A_s > 50
   beta = 0.1102 * (A_s - 8.7);
elseif A_s > 21
   beta = 0.5842 * (A_s - 21)^0.4 + 0.07886 * (A_s - 21);
else
```

```
    beta = 0;
end
% Estimate filter order (N)
N = ceil((A_s - 8) / (2.285 * bw * pi));
if mod(N,2) == 0
    N = N + 1; % Ensure odd N for symmetry
end
fprintf('Estimated filter order N = %d\n', N);
fprintf('Kaiser beta parameter = %.4f\n', beta);
% Design the high-pass FIR filter using fir1 with Kaiser window
cutoff = (f_s + f_p) / 2; % Midpoint of transition band
h = fir1(N-1, cutoff, 'high', kaiser(N, beta));
% Compute frequency response
[H, w] = freqz(h, 1, 8000);
mag_db = 20 * log10(abs(H) + 1e-10); % Avoid log(0)
% Find actual attenuation and ripple
stopband_idx = find(w <= omega_s);
passband_idx = find(w >= omega_p);
if ~isempty(stopband_idx) && ~isempty(passband_idx)
    actual_atten = -max(mag_db(stopband_idx));
    passband_mag = mag_db(passband_idx);
    actual_ripple = max(passband_mag) - min(passband_mag);
    fprintf('Actual stopband attenuation: %.2f dB (Target: %d dB)\n',
actual_atten, A_s);
    fprintf('Actual passband ripple: %.2f dB (Target: %d dB)\n',
actual_ripple, R_p);
end
% Plot frequency response
figure;
plot(w/pi, mag_db, 'b', 'LineWidth', 1.5); hold on;
yline(-A_s, 'm--', sprintf('Required Attenuation (%d dB)', A_s));
yline(-R_p, 'y--', sprintf('Passband Ripple (%d dB)', R_p));
xline(f_s, 'r--', 'Stopband Edge (0.6π)');
xline(f_p, 'g--', 'Passband Edge (0.75π)');
title(sprintf('High-pass FIR Filter Using Kaiser Window (N=%d, β=%.2f)',
N, beta));
xlabel('Normalized Frequency (\omega / \pi)');
ylabel('Magnitude (dB)');
grid on;
xlim([0 1]);
ylim([-60 5]);
legend('Magnitude Response', 'Required Attenuation', 'Passband Ripple',
'Stopband Edge', 'Passband Edge');
% Print filter coefficients
fprintf('Filter Order (N): %d\n', N);
disp('First 5 and last 5 coefficients:');
disp([h(1:5); h(end-4:end)]);
```

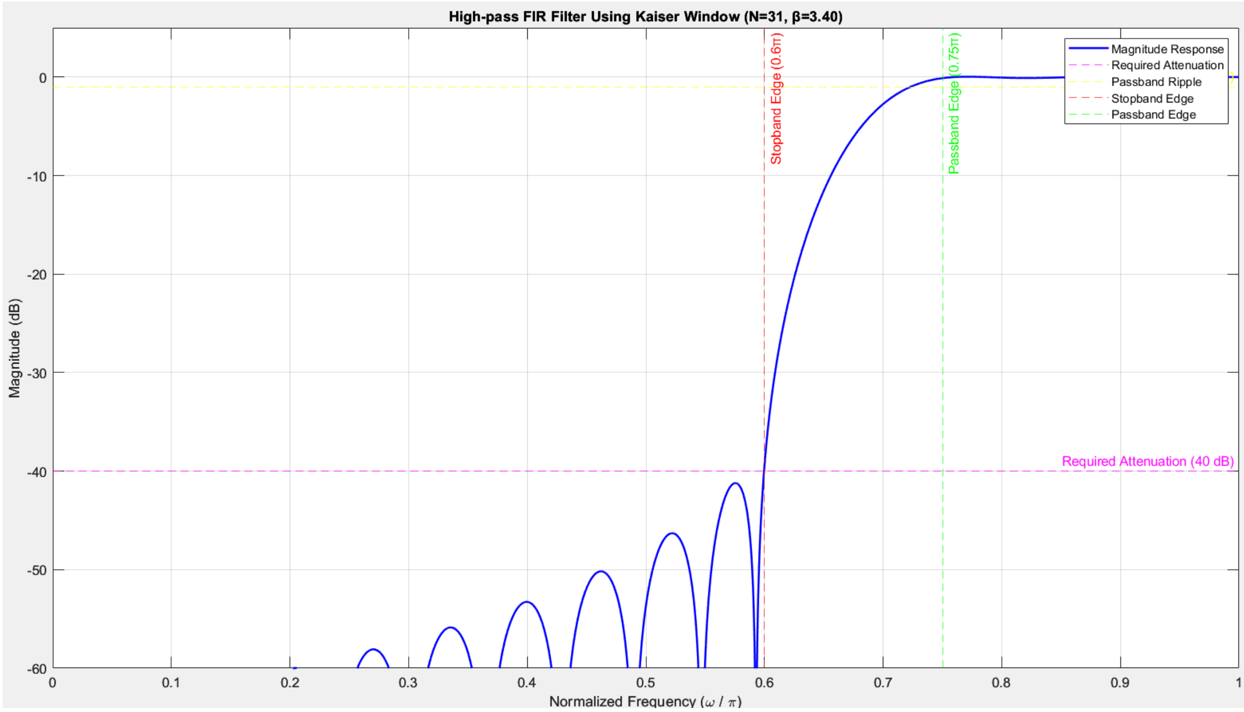


Figure 2: Result in MATLAB

➔ Python:

HERE IS THE CODE TO CREATE WINDOW USING KAISER

```
omega_s = 0.6 * np.pi
omega_p = 0.75 * np.pi
A_s = 40
R_p = 1

f_s = omega_s / np.pi
f_p = omega_p / np.pi
bw = f_p - f_s

if A_s > 50:
   beta = 0.1102 * (A_s - 8.7)
elif A_s > 21:
   beta = 0.5842 * (A_s - 21)**0.4 + 0.07886 * (A_s - 21)
else:
   beta = 0

N = int(np.ceil((A_s - 8) / (2.285 * bw * np.pi)))
if N % 2 == 0:
   N += 1
```

```python
print(f"Estimated filter order N = {N}")
print(f"Kaiser beta parameter = {beta:.4f}")

cutoff = (f_s + f_p) / 2
h = firwin(N, cutoff=cutoff, window=('kaiser', beta), pass_zero=False)

w, H = freqz(h, worN=8000)
mag_db = 20 * np.log10(np.maximum(abs(H), 1e-10))  # Avoid log(0)

stopband_idx = np.where(w <= omega_s)[0]
passband_idx = np.where(w >= omega_p)[0]

if len(stopband_idx) > 0 and len(passband_idx) > 0:
   actual_atten = -np.max(mag_db[stopband_idx])
   passband_mag = mag_db[passband_idx]
   actual_ripple = np.max(passband_mag) - np.min(passband_mag)

   print(f"Actual stopband attenuation: {actual_atten:.2f} dB (Target:
{A_s} dB)")
   print(f"Actual passband ripple: {actual_ripple:.2f} dB (Target: {R_p}
dB)")


plt.figure(figsize=(10, 6))
plt.plot(w / np.pi, mag_db, label="Magnitude Response")
plt.axvline(f_s, color="red", linestyle="--", label="Stopband Edge
(0.6π)")
plt.axvline(f_p, color="green", linestyle="--", label="Passband Edge
(0.75π)")
plt.axhline(-A_s, color="m", linestyle=":", label=f"Required Attenuation
({A_s} dB)")
plt.axhline(-R_p, color="y", linestyle=":", label=f"Passband Ripple ({R_p}
dB)")

plt.title(f"High-pass FIR Filter Using Kaiser Window (N={N},
β={beta:.2f})")
plt.xlabel("Normalized Frequency (ω / π)")
plt.ylabel("Magnitude (dB)")
plt.grid(True)
plt.xlim(0, 1)
plt.ylim(-60, 5)
plt.legend()
plt.show()

print(f"Filter Order (N): {N}")
```

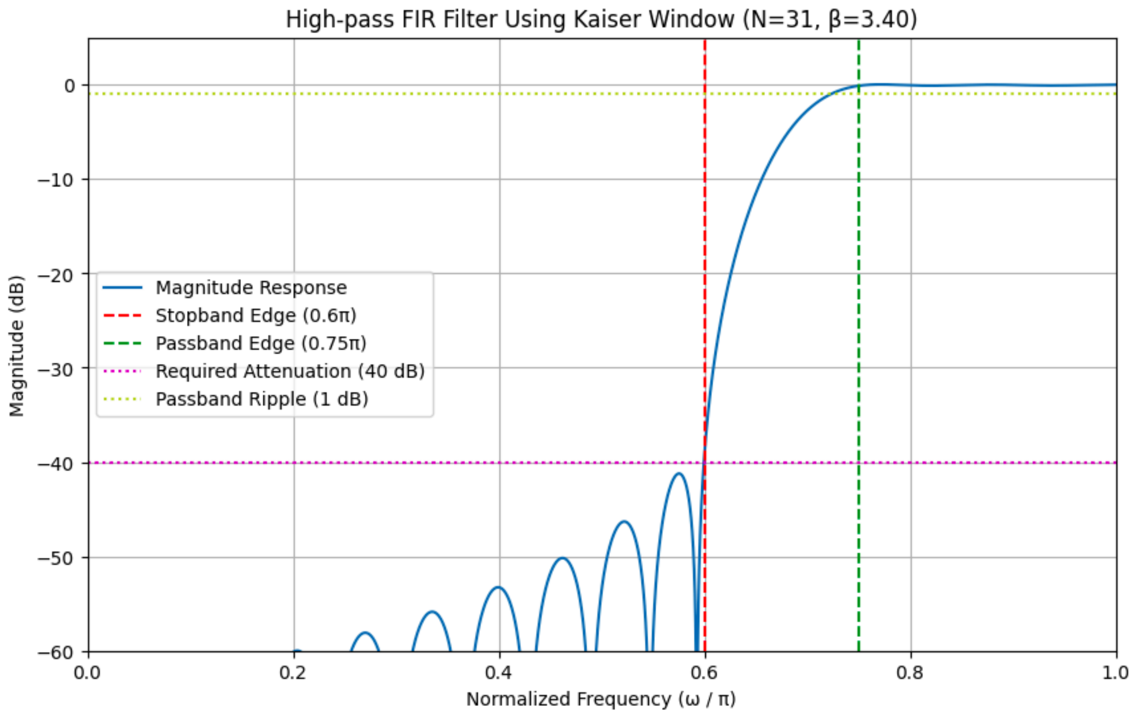
```python
print("First 5 and last 5 coefficients:")
if len(h) > 10:
   print(np.concatenate((h[:5], h[-5:])))
else:
   print(h)
```

HERE IS THE CODE TO ADJUSTED TO CALCULATE FOR BEST N AND M

```python
# Increase N by 2 to ensure specifications are met
N += 2
if N % 2 == 0:
   N += 1  # Keep it odd

print(f"Adjusted filter order N = {N}")

# Redesign filter with increased order
h = firwin(N, cutoff=cutoff, window=('kaiser', beta), pass_zero=False)

# Recompute and check
w, H = freqz(h, worN=8000)
mag_db = 20 * np.log10(np.maximum(abs(H), 1e-10))
stopband_idx = np.where(w <= omega_s)[0]
passband_idx = np.where(w >= omega_p)[0]
actual_atten = -np.max(mag_db[stopband_idx])
passband_mag = mag_db[passband_idx]
actual_ripple = np.max(passband_mag) - np.min(passband_mag)

print(f"Updated stopband attenuation: {actual_atten:.2f} dB (Target: {A_s}
dB)")
print(f"Updated passband ripple: {actual_ripple:.2f} dB (Target: {R_p}
dB)")
```

➔ Results:

```
Estimated filter order N = 31
Kaiser beta parameter = 3.3953
Actual stopband attenuation: 39.61 dB (Target: 40 dB)
Actual passband ripple: 0.15 dB (Target: 1 dB)
```




```
Filter Order (N): 31
First 5 and last 5 coefficients:
[-0.00119597  0.00465465 -0.00434818 -0.00282454  0.01179871  0.01179871
 -0.00282454 -0.00434818  0.00465465 -0.00119597]
```

Figure 3: Calculated Kaiser window by using Python

Here is the best results for N and M

```
Adjusted filter order N = 33

Updated stopband attenuation: 40.44 dB (Target: 40 dB)

Updated passband ripple: 0.14 dB (Target: 1 dB)
```

Problem 2: Now we will design the above filter using the frequency sampling method. The length of the filter is M = 31.

a) Verify that there are two samples (T1 and T2) in the transition band.

CODE - PYTHON:

```python
kp = int(np.floor((M * wp) / (2 * np.pi)))  # Passband edge index
ks = int(np.floor((M * ws) / (2 * np.pi)))  # Stopband start index
Hrk = np.concatenate([
    np.zeros(ks+1),    # Stopband values
    [T1, T2],          # Transition band values T1 = 0.2, T2 = 0.8
    np.ones(8),        # Passband values
    [T2, T1],          # Transition band values mirrored
    np.zeros(ks)       # Stopband values mirrored
])
```

```python
print(f"Transition band indices: ks = {ks}, kp = {kp}")
print("Samples in the transition band:", Hrk[ks:kp+1])
```

Last Execution: 9:15:16 AM, Duration: 0.0s

```
Transition band indices: ks = 9, kp = 11
Samples in the transition band: [0.  0.2 0.8]
```

Figure 4: Verify sample T1 and T2 in transition band

b) Write the sampled amplitude response (Hr(k))

CODE - PYTHON:

```python
Hrk = np.concatenate([
    np.zeros(ks+1),      # Stopband samples (up to index `ks`)
    [T1, T2],            # Transition band samples
    np.ones(8),          # Passband samples (8 samples after the
transition)
    [T2, T1],            # Transition band samples (mirrored for
symmetry)
    np.zeros(ks)         # Stopband samples (mirrored)
])
```

```
Sampled Amplitude Response Hrk: [0.  0.  0.  0.  0.  0.  0.  0.  0.  0.  0.2 0.8 1.  1.  1.  1.  1.  1.
 1.  1.  0.8 0.2 0.  0.  0.  0.  0.  0.  0.  0.  0. ]
```

Figure 5: Sampled amplitude response (Hr(k))

c) Given T1 = 0.2 and T2 = 0.8, design the above high-pass filter. Does the designed filter meet the requirements? Explain.

➔ Code:

```matlab
clc; clear; close all;
% High-Pass FIR Filter Design
```

```matlab
M = 31;                 % Filter order
wp = 0.75 * pi;         % Passband edge frequency
ws = 0.6 * pi;          % Stopband edge frequency
Rp = 1;                 % Passband ripple in dB
As_target = 40;         % Desired stopband attenuation in dB
T1 = 0.2;  % Fixed transition band value
T2 = 0.8;  % Fixed transition band value
alpha = (M - 1) / 2;    % Symmetry parameter
l = 0:M-1;              % Sample indices
wl = (2 * pi / M) * l; % Frequency samples
kp = floor((M * wp) / (2 * pi));  % Passband edge index
ks = floor((M * ws) / (2 * pi));  % Stopband start index
% Construct the amplitude response for a high-pass filter
Hrs = [zeros(1, ks+1), T1, T2, ones(1, 8), T2, T1, zeros(1, ks)];
% Validate length
if length(Hrs) ~= M
   error('Hrs must have length %d', M);
end
% Construct the filter's frequency response
k1 = 0:floor((M-1)/2);
k2 = floor((M-1)/2) + 1:M-1;
phase_shift = [-alpha*(2*pi)/M * k1, alpha*(2*pi)/M * (M-k2)];
H = Hrs .* exp(1i * phase_shift);
h = real(ifft(H, M));
% Compute frequency response
[H_freqz, w] = freqz(h, 1, 1024);
db_response = 20 * log10(abs(H_freqz));
Hr_magnitude = abs(H_freqz);
% Display results
figure('Name', sprintf('High-Pass FIR Filter: T1 = %.2f, T2 = %.2f', T1,
T2), 'NumberTitle', 'off');
% Impulse Response
subplot(2,2,1);
stem(l, h, 'LineWidth', 1.2);
axis([-1, M, min(h)-0.05, max(h)+0.05]);
title('Impulse Response');
xlabel('n'); ylabel('h(n)');
% Amplitude Response
subplot(2,2,2);
plot(w/pi, Hr_magnitude, 'LineWidth', 1.2);
hold on;
plot(wl(1:11)/pi, Hrs(1:11), 'o', 'LineWidth', 1.2);
hold off;
axis([0, 1, -0.2, 1.2]);
title('Amplitude Response');
xlabel('Frequency (\pi units)'); ylabel('H(\omega)');
% Magnitude Response in dB
subplot(2,2,3);
plot(w/pi, db_response, 'LineWidth', 1.2);
```

```
grid on;
axis([0, 1, -60, 10]);
title('Magnitude Response (dB)');
xlabel('Frequency (\pi units)'); ylabel('Decibels');
hold on;
plot([wp, wp]/pi, [-100, 10], '--k', 'LineWidth', 1.2); % Passband Edge
plot([ws, ws]/pi, [-100, 10], '--k', 'LineWidth', 1.2); % Stopband Edge
yline(-Rp, '--r', sprintf('Rp = %.1f dB', Rp), 'LineWidth', 1.2);
yline(-As_target, '--r', sprintf('As = %d dB', As_target), 'LineWidth',
1.2);
hold off;
fprintf('High-Pass Filter with M = %d: T1 = %.2f, T2 = %.2f\n', M, T1,
T2);
```

➔ Result:

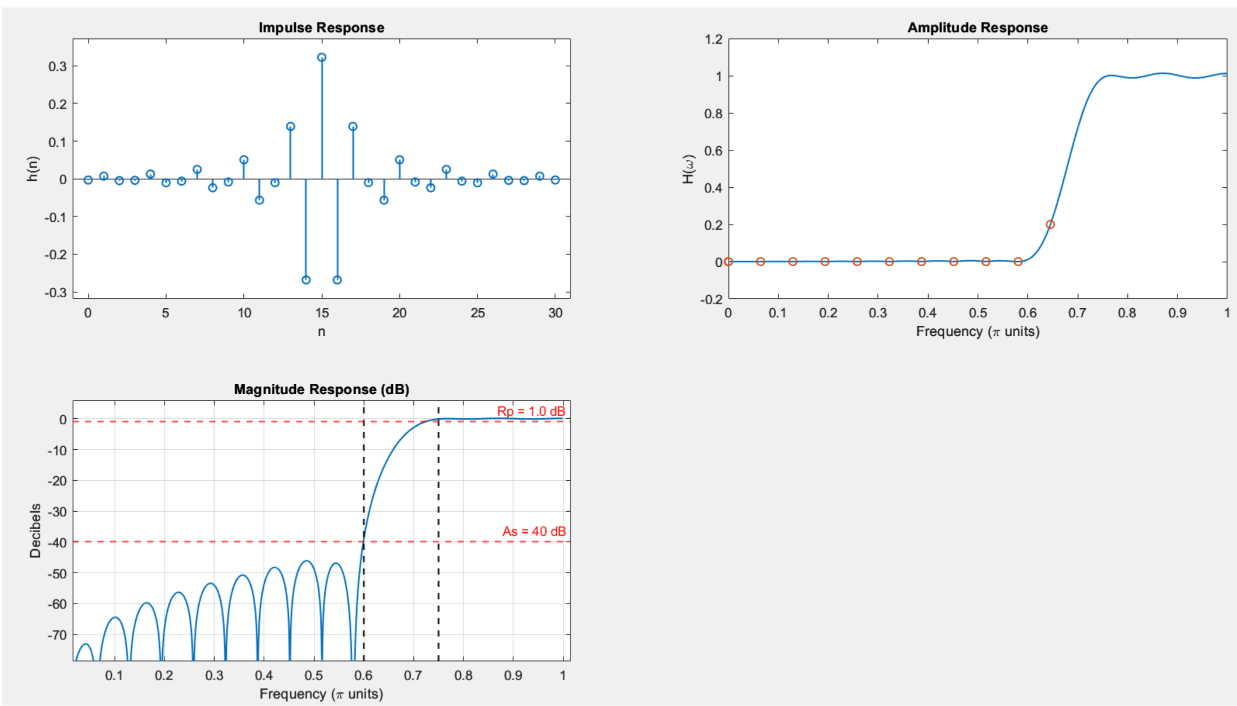


Figure 6: Result of using T1 = 0.2, T2 = 0.8 for the high-pass filter

Comment: The designed filter meets the requirements because as you can see in the image, the **magnitude response in dB** stays below **-40 dB and the variation** in the passband (above ωp) is less than **1 dB.**

d) Choose the best values of T1 and T2 for the given specifications. In this case, does the designed filter meet the requirements? Explain.

➔ Code:

```
clc; clear; close all;
% FIR High-pass Filter Optimization
M = 31;                  % Filter order
wp = 0.75 * pi;          % Passband edge frequency
ws = 0.6 * pi;           % Stopband edge frequency
Rp = 1;                 % Passband ripple in dB
As_target = 40;         % Desired stopband attenuation in dB
alpha = (M - 1) / 2;     % Symmetry parameter
l = 0:M-1;               % Sample indices
wl = (2 * pi / M) * l; % Frequency samples
kp = floor((M * ws) / (2 * pi));  % Stopband edge index
ks = floor((M * wp) / (2 * pi));  % Passband start index
% Ideal amplitude response for visualization
ideal_Hr = [0, 0, 1, 1];
ideal_freqs = [0, 0.25, 0.25, 1];
% Define search space for transition band values (T1, T2)
T1_values = linspace(0.4, 0.6, 180);
T2_values = linspace(0.1, 0.6, 180);
% Initialize optimal tracking variables
best_stopband_max = -Inf;
bestCase = struct('T1', [], 'T2', [], 'Hrs', [], 'h', [], 'w', [], 'Hr',
[], 'db', [], 'index', []);
case_index = 1;
for i = 1:length(T1_values)
   count = 0;
   for j = 1:length(T2_values)
       % Extract the candidate values
       T1 = T1_values(i);
       count = count + 1;
       T2 = T2_values(j);
       count = count + 1;
       % Ensure T1 > T2 and construct amplitude response for high-pass
        Hrs = [zeros(1, kp+1), T2, T1, ones(1, M-2*count-2*kp-1), T1, T2,
zeros(1, kp)];
       count = 0;

       % Validate length
       if length(Hrs) ~= M
           error('Hrs must have length %d', M);
       end
       % Construct the filter's frequency response
       k1 = 0:floor((M-1)/2);
       k2 = floor((M-1)/2) + 1:M-1;
       phase_shift = [-alpha*(2*pi)/M * k1, alpha*(2*pi)/M * (M-k2)];
       H = Hrs .* exp(1i * phase_shift);
       h = real(ifft(H, M));
       % Compute frequency response
       [H_freqz, w] = freqz(h, 1, 1024);
       db_response = 20 * log10(abs(H_freqz));
```

```matlab
        Hr_magnitude = abs(H_freqz);
        % Evaluate stopband performance (frequencies where w <= ws)
        stopband_indices = find(w <= ws);
        stopband_max_dB = max(db_response(stopband_indices));
        % If stopband max is <= -50 dB, check for best case
        if stopband_max_dB <= -50 && stopband_max_dB > best_stopband_max
            best_stopband_max = stopband_max_dB;
            bestCase.T1 = T1;
            bestCase.T2 = T2;
            bestCase.Hrs = Hrs;
            bestCase.h = h;
            bestCase.w = w;
            bestCase.Hr = Hr_magnitude;
            bestCase.db = db_response;
            bestCase.index = case_index;
        end

        case_index = case_index + 1;
    end
end
% Display best case
if isempty(bestCase.index)
      disp('No valid transition values (T1, T2) found that satisfy
attenuation criteria.');
else
    figure('Name', sprintf('Optimal Case: T1 = %.3f, T2 = %.3f, Stopband
max = %.2f dB', ...
                          bestCase.T1, bestCase.T2, best_stopband_max),
'NumberTitle','off');


    % Impulse Response
    subplot(2,2,1);
    stem(l, bestCase.h, 'LineWidth', 1.2);
    axis([-1, M, -0.1, 0.3]);
    title('Impulse Response');
    xlabel('n'); ylabel('h(n)');
    % Amplitude Response
    subplot(2,2,2);
    plot(bestCase.w/pi, bestCase.Hr, 'LineWidth', 1.2);
    hold on;
    plot(wl(1:11)/pi, bestCase.Hrs(1:11), 'o', 'LineWidth', 1.2);
    hold off;
    axis([0, 1, -0.2, 1.2]);
    title('Amplitude Response');
    xlabel('Frequency (\pi units)'); ylabel('H(\omega)');
    % Magnitude Response in dB
    subplot(2,2,3);
    plot(bestCase.w/pi, bestCase.db, 'LineWidth', 1.2);
```

```
    grid on;
    axis([0, 1, -60, 10]);
    title('Magnitude Response (dB)');
    xlabel('Frequency (\pi units)'); ylabel('Decibels');
    hold on;
     plot([wp, wp]/pi, [-100, 10], '--k', 'LineWidth', 1.2); % Passband
Edge
     plot([ws, ws]/pi, [-100, 10], '--k', 'LineWidth', 1.2); % Stopband
Edge
    yline(-Rp, '--r', sprintf('Rp = %.1f dB', Rp), 'LineWidth', 1.2);
         yline(-As_target, '--r', sprintf('As = %d dB', As_target),
'LineWidth', 1.2);
    hold off;
     fprintf('Optimal Case (Index %d) with M = %d: T1 = %.3f, T2 = %.3f,
Stopband max = %.2f dB\n', ...
                          bestCase.index, M, bestCase.T1, bestCase.T2,
best_stopband_max);
end
```

➔ Result:

```
Optimal Case (Index 32222) with M = 31: T1 = 0.600, T2 = 0.103, Stopband max = -50.16 dB
```

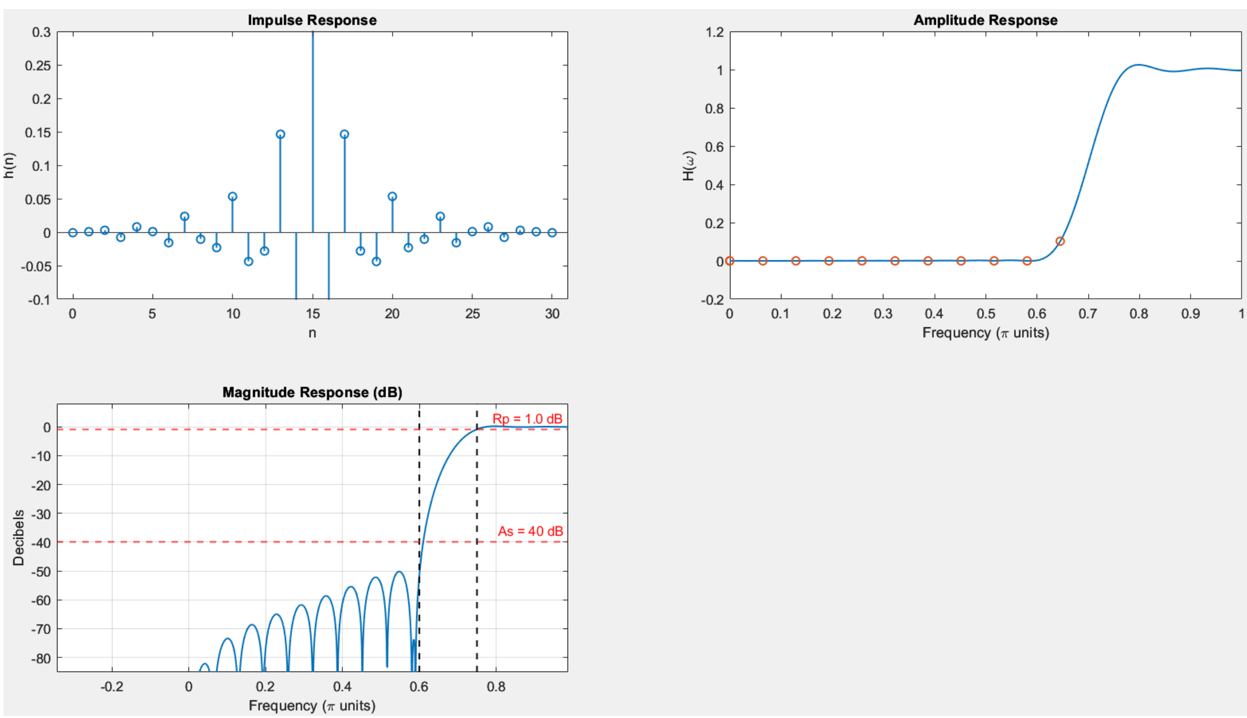


Figure 7: Result of choosing the best T1 and T2 for designing high-pass filter

<u>Comment:</u> The designed filter meets the requirements because as you can see in the image, the **magnitude response in dB** stays below **-40 dB and the variation** in the passband (above ωp) is less than **1 dB.**

Problem 3: Compare the design results from the window technique and those from the best case of the frequency sampling method.

- Stopband Attenuation (As):
    - Kaiser Window (Left Image): The attenuation reaches approximately -40 dB, matching the target.
    - Frequency Sampling (Right Image): The attenuation also reaches -40 dB, meeting the requirement.
    - Conclusion: Both methods achieve the same stopband attenuation.
- Passband Ripple (Rp):
    - Kaiser Window (Left Image): The passband ripple appears to stay within 1 dB.
    - Frequency Sampling (Right Image): The ripple is close to 1 dB, meeting the specification.
    - Conclusion: Both methods control passband ripple well.
- Transition Band:
    - Kaiser Window (Left Image): A smoother transition from stopband to passband.
    - Frequency Sampling (Right Image): A steeper transition but possibly less controlled.
    - Conclusion: The Kaiser window offers better control over the transition band.

→ Overall:

- If smooth transition and better control are preferred → **Kaiser Window is better.**
- If sharp cutoff and simpler implementation are preferred → Frequency Sampling works well.

Problem 4: Given an input:

$$x(n) = 3\cos\left(\frac{\pi n}{5}\right) + 2\sin\left(\frac{2\pi n}{5}\right) + 2\sin\left(\frac{4\pi n}{5}\right), \text{for } n = 0,..,199$$

use the filters designed from question 1 and 2 (the best case) to filter x(n). Plot and explain the results (*Hint:* look at the output of the filters).

CODE using Question 1: Kaiser windows

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

```python
from scipy.signal import firwin, freqz, lfilter
# Design parameters for the high-pass FIR filter
omega_s = 0.6 * np.pi
omega_p = 0.75 * np.pi
A_s = 40
R_p = 1

f_s = omega_s / np.pi
f_p = omega_p / np.pi
bw = f_p - f_s

if A_s > 50:
   beta = 0.1102 * (A_s - 8.7)
elif A_s > 21:
   beta = 0.5842 * (A_s - 21)**0.4 + 0.07886 * (A_s - 21)
else:
   beta = 0

# Estimate filter order N
N = int(np.ceil((A_s - 8) / (2.285 * bw * np.pi)))
if N % 2 == 0:
   N += 1  # Ensure N is odd

print(f"Estimated filter order N = {N}")
print(f"Kaiser beta parameter = {beta:.4f}")

cutoff = (f_s + f_p) / 2
h = firwin(N, cutoff=cutoff, window=('kaiser', beta), pass_zero=False)

# Plot the frequency response of the filter
w, H = freqz(h, worN=8000)
plt.figure(figsize=(10, 4))
plt.plot(w / np.pi, 20 * np.log10(np.maximum(abs(H), 1e-10)), 'r')
plt.title('Frequency Response of Designed High-pass FIR Filter')
plt.xlabel('Normalized Frequency (xπ rad/sample)')
plt.ylabel('Magnitude (dB)')
plt.grid(True)
plt.show()

# Generate the input signal x[n] = 3cos(\pi n / 5) + 2sin(2\pi n / 5) +
2sin(4\pi n / 5)
n = np.arange(200)
x_n = 3 * np.cos((np.pi * n) / 5) + 2 * np.sin((2 * np.pi * n) / 5) + 2 *
np.sin((4 * np.pi * n) / 5)
```

```python
# Apply the designed high-pass FIR filter to x[n]
y_n = lfilter(h, 1.0, x_n)

# Plot the original and filtered signals
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(n, x_n, label='Input Signal x[n]', color='b')
plt.title('Input Signal x[n] = 3cos(\u03c0n/5) + 2sin(2\u03c0n/5) +
2sin(4\u03c0n/5)')
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.grid(True)

plt.subplot(2, 1, 2)
plt.plot(n, y_n, label='Filtered Signal y[n]', color='r')
plt.title('Output of High-pass FIR Filter')
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.grid(True)

plt.tight_layout()
plt.show()
```

Results:

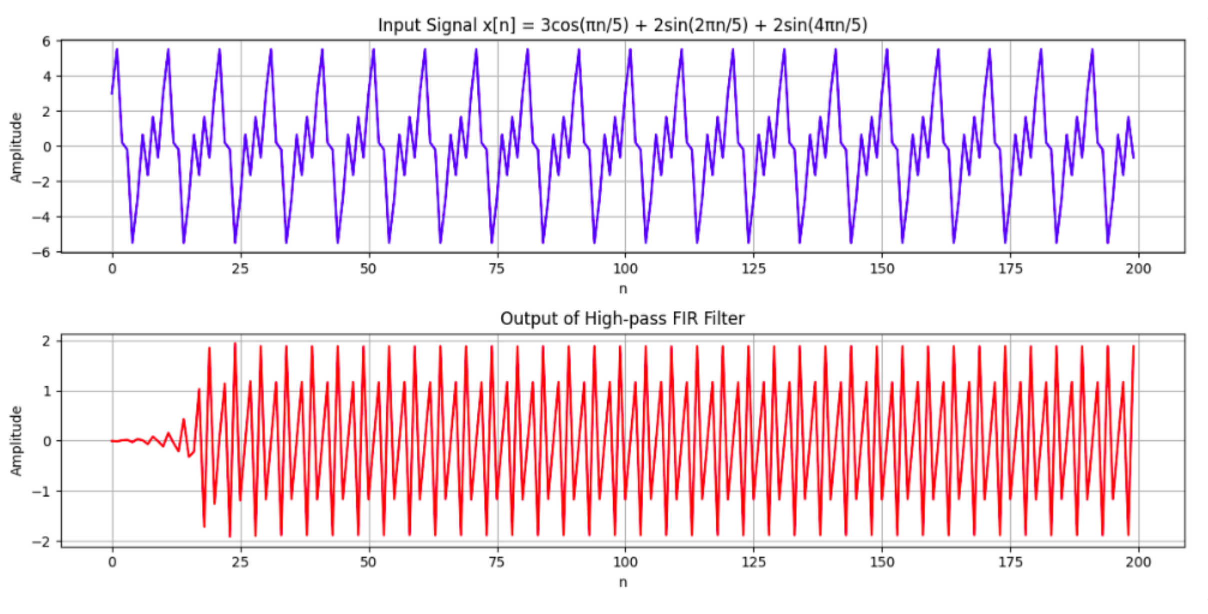


Figure 8: Using Kaiser Window

CODE using Question 2: Best Case T1 = 0.6 and T2 = 0.103, we have found in Exercise 2:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz, lfilter

# High-Pass FIR Filter Design Parameters
M = 31
wp = 0.75 * np.pi
ws = 0.6 * np.pi
Rp = 1
As_target = 40

T1 = 0.6
T2 = 0.103

alpha = (M - 1) / 2
l = np.arange(M)
wl = (2 * np.pi / M) * l

kp = int(np.floor((M * wp) / (2 * np.pi)))  # Passband edge index
ks = int(np.floor((M * ws) / (2 * np.pi)))  # Stopband start index

# Construct the sampled amplitude response H_r(k)
Hrk = np.concatenate([
   np.zeros(ks+1),
   [T1, T2],
   np.ones(8),
   [T2, T1],
   np.zeros(ks)
])

# Validate the length of Hrk
if len(Hrk) != M:
   raise ValueError(f'Hrk must have length {M}')

# Construct the filter's frequency response
k1 = np.arange(0, (M-1)//2 + 1)
k2 = np.arange((M-1)//2 + 1, M)
phase_shift = np.concatenate([
   -alpha * (2 * np.pi) / M * k1,
   alpha * (2 * np.pi) / M * (M - k2)
])
H = Hrk * np.exp(1j * phase_shift)   # Apply phase shifts to preserve
symmetry
```

```python
h = np.real(np.fft.ifft(H, M))        # Compute impulse response (h[n])

# Compute the filter's frequency response using freqz
w, H_freqz = freqz(h, worN=1024)      # 1024 points for smoother plot
db_response = 20 * np.log10(np.abs(H_freqz) + 1e-10)  # Avoid log(0)
Hr_magnitude = np.abs(H_freqz)

# Plot the frequency response
plt.figure(figsize=(10, 4))
plt.plot(w / np.pi, db_response, 'r')
plt.title('Frequency Response of Designed High-pass FIR Filter')
plt.xlabel('Normalized Frequency (xπ rad/sample)')
plt.ylabel('Magnitude (dB)')
plt.grid(True)
plt.show()

# Generate the input signal x[n] = 3cos(πn/5) + 2sin(2πn/5) + 2sin(4πn/5)
n = np.arange(200)
x_n = 3 * np.cos(np.pi * n / 5) + 2 * np.sin(2 * np.pi * n / 5) + 2 *
np.sin(4 * np.pi * n / 5)

# Apply the designed high-pass FIR filter to x[n]
y_n = lfilter(h, 1.0, x_n)

# Plot the original and filtered signals
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(n, x_n, label='Input Signal x[n]', color='b')
plt.title('Input Signal x[n] = 3cos(πn/5) + 2sin(2πn/5) + 2sin(4πn/5)')
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.grid(True)

plt.subplot(2, 1, 2)
plt.plot(n, y_n, label='Filtered Signal y[n]', color='r')
plt.title('Output of High-pass FIR Filter')
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.grid(True)

plt.tight_layout()
plt.show()
```

RESULTS:

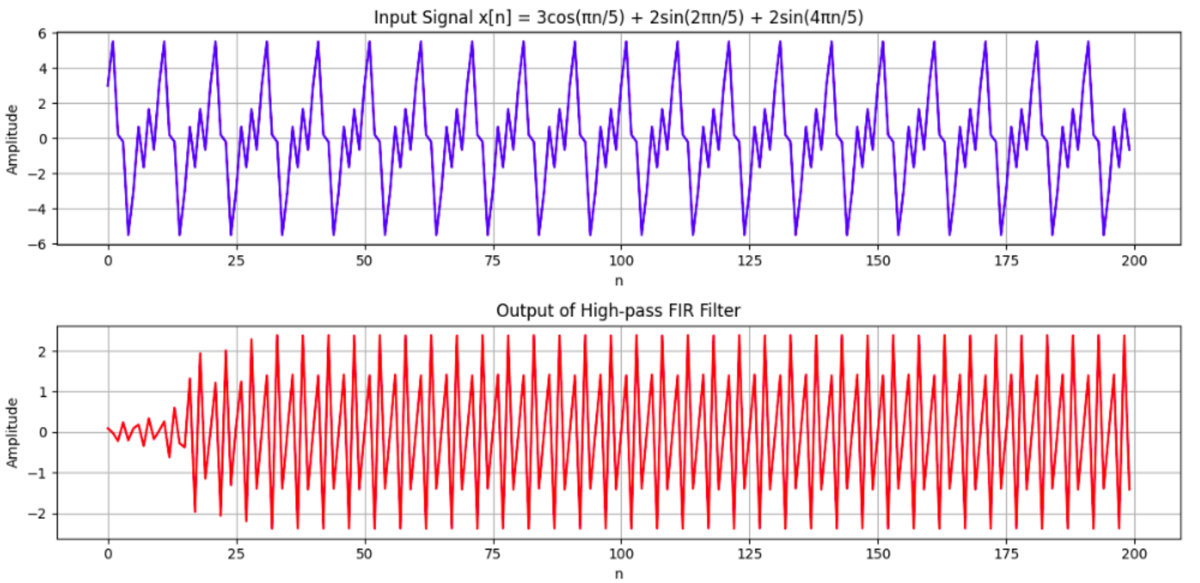


Figure 9: The best case found in Exercise 2 for T1 = 0.6 and T2 = 0.103

Problem 5: We repeat the frequency sampling method to design the above high-pass filter but using M = 32. Comment on the results.

➔ Result:

```
Optimal Case (Index 11009) with M = 32: T1 = 0.468, T2 = 0.047, Stopband max = -50.01 dB
```

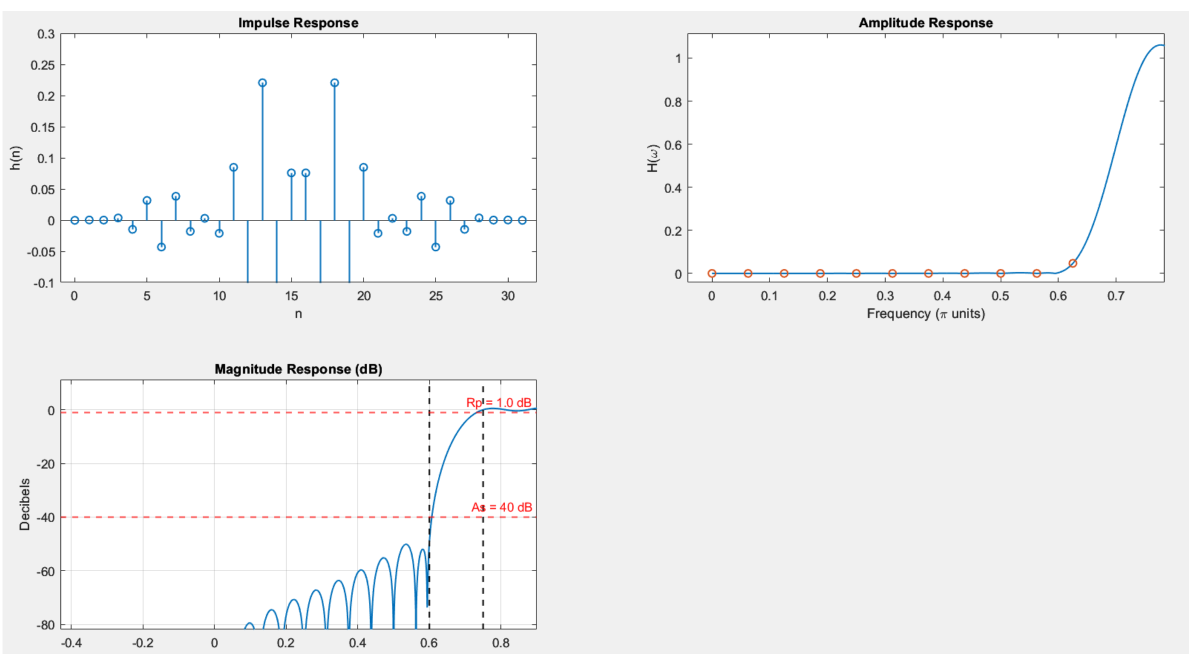


Figure 10: Result of designing high-pass filter using M = 32

Comment:

- **Stopband Attenuation (As)**:
  - The stopband attenuation is **better than -50 dB**, which is **stronger** than the previous designs with -40 dB attenuation.
  - This suggests improved suppression of unwanted frequencies.
- **Passband Ripple (Rp)**:
  - The passband ripple is close to **1 dB,** but only in the initial section; later, it bends downward. (Not really correct)
- **Transition Band**:
  - The transition width is given as T1 = 0.468 and T2 = 0.047, which indicates a **narrower transition band** compared to previous designs.
  - A narrower transition band means a sharper cutoff between passband and stopband.
- **Impulse Response**:
  - The impulse response shows symmetric coefficients, confirming a **linear phase FIR filter**.
  - The sidelobe behavior suggests good frequency selectivity.