# Digital Signal Processing Lab (ECL333)

## Experiment 6: FIR Filter Design for Audio Denoising

Department of Electronics and Communication Engineering

Faculty: Jinu Jayachandran, Viji R

**Student Name:** ____________________ **Roll Number:** ________ **Date:** __________

## Instructions

1. Complete the Pre-lab work before the lab session.
2. Write all answers neatly within the space provided. Do not attach additional sheets.
3. Submit your lab sheet before leaving the lab. Late submissions receive zero marks.
4. Ensure to import Python libraries ‘numpy‘, ‘matplotlib.pyplot‘, ‘scipy.signal‘, ‘soundfile‘, and ‘time‘.
5. The post-lab questions are indicated by **PTQ**.
6. Make sure the ‘noisy_speech_8k.wav‘ file (uploaded in ETLAB) is in the same directory as your lab code.

## Learning Objectives

By the end of this lab, you will be able to:

- Design linear-phase FIR filters using common windows: Rectangular, Hanning (Hann) and Hamming.
- Estimate filter order from given passband/stopband specs and transition width.
- Implement and visualize magnitude/phase response and impulse response.
- Apply your designed filter to a real/synthetic signal audio and evaluate its effect.

## Background Theory

An ideal frequency response $H_d(e^{j\omega})$ corresponds to an infinite-duration impulse response $h_d[n]$. A realizable FIR filter is obtained by windowing the ideal impulse response:

$$h[n] = h_d[n] \cdot w[n], \quad n = 0, 1, \ldots, N-1$$

where $w[n]$ is a finite-length window of length $N$.

## Ideal Impulse Responses

Let $M = \frac{N-1}{2}$. For $n \neq M$,

$$\begin{aligned}
\text{LPF (cutoff } \omega_c): \quad & h_d[n] = \frac{\sin\big(\omega_c(n-M)\big)}{\pi(n-M)}, \\
\text{HPF (cutoff } \omega_c): \quad & h_d[n] = \delta[n-M] - \frac{\sin\big(\omega_c(n-M)\big)}{\pi(n-M)}, \\
\text{BPF } (\omega_1<\omega_2): \quad & h_d[n] = \frac{\sin\big(\omega_2(n-M)\big) - \sin\big(\omega_1(n-M)\big)}{\pi(n-M)}, \\
\text{BSF } (\omega_1<\omega_2): \quad & h_d[n] = \delta[n-M] - \frac{\sin\big(\omega_2(n-M)\big) - \sin\big(\omega_1(n-M)\big)}{\pi(n-M)}.
\end{aligned}$$

Handle $n = M$ by continuity: for LPF, $h_d[M] = \omega_c/\pi$; for BPF, $h_d[M] = (\omega_2 - \omega_1)/\pi$.

## Common Windows

For $0 \leq n \leq N-1$:

$$\begin{aligned}
\text{Rectangular: } & w[n] = 1. \\
\text{Hann: } & w[n] = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right). \\
\text{Hamming: } & w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right).
\end{aligned}$$

## Order & Transition Width (Rules of Thumb)

The transition width $\Delta\omega$ is defined as

$$\Delta\omega = \omega_{sb} - \omega_p$$

where $\omega_{sb}$ and $\omega_p$ are the digital stop band edge frequency and digital pass band edge frequency.

The digital angular frequencies are obtained from the analog frequencies by

$$\omega_p = \frac{2\pi f_p}{f_s}, \qquad \omega_{sb} = \frac{2\pi f_{sb}}{f_s}.$$

where $f_s$ is the sampling frequency.

In the window method, the frequency response of the designed FIR filter is the convolution of the ideal low-pass filter response with the Fourier transform of the chosen window. The **main-lobe width** of the window determines the **transition width** $\Delta\omega$ of the resulting filter.

For a Rectangular window of length $N$, the frequency response is given by

$$W(e^{j\omega}) \ = \ \frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}.$$

The first zero of $W(e^{j\omega})$ occurs at $\omega = \frac{2\pi}{N}$. Hence, the main-lobe width (between the first positive and first negative zero) is approximately $\Delta\omega_{\text{main}} \approx \frac{4\pi}{N}$.

The transition width of the filter, $\Delta\omega$, is approximately equal to the main-lobe width of the window: $\Delta\omega \approx \Delta\omega_{\text{main}} = \frac{4\pi}{N}$.

Rearranging the above gives the design formula:

$$N \approx \frac{4\pi}{\Delta\omega}.$$

The Rectangular window provides the narrowest main-lobe (sharper transition) for a given $N$, but it also has the largest side-lobes (poor stopband attenuation). Other windows (Hann, Hamming, Blackman, etc.) reduce side-lobes at the cost of wider main-lobes. For these windows, the required filter length $N$ is larger, e.g., $N \approx \frac{8\pi}{\Delta\omega}$ for Hann or Hamming windows.

- Rectangular: $N \approx \dfrac{4\pi}{\Delta\omega}$.
- Hann: $N \approx \dfrac{8\pi}{\Delta\omega}$.
- Hamming: $N \approx \dfrac{8\pi}{\Delta\omega}$.

# Pre-Lab Questions

1. A low-pass FIR filter is to be designed with $f_s = 10{,}000\,\text{Hz}$, passband edge $f_p = 1500\,\text{Hz}$, and stopband edge $f_{sb} = 2000\,\text{Hz}$. Using the Rectangular window, estimate the required filter length $N$. [2 Marks]

2. For an FIR filter designed with a Hann window, the passband and stopband edges are $f_p = 1200\,\text{Hz}$ and $f_{sb} = 1600\,\text{Hz}$ with $f_s = 8000\,\text{Hz}$.
   (a) Compute the transition width $\Delta\omega$.
   (b) Estimate the filter length $N$.
   (c) What is the expected group delay $\tau_g$ of this filter?

   [3 Marks]

3. Design an FIR low-pass filter using the Hamming window with $f_s = 4000\,\text{Hz}$, $f_p = 100\,\text{Hz}$, $f_{sb} = 2000\,\text{Hz}$. Calculate the transition width $\Delta\omega$ and estimate the filter length $N$. (Use $N \approx \left\lceil \frac{8\pi}{\Delta\omega} \right\rceil$.) [5 Marks]

# 1 Detailed Algorithm: FIR Low-Pass Filter Design (Window Method)

This algorithm provides a detailed step-by-step procedure for designing a Finite Impulse Response (FIR) low-pass filter using the windowing method.

## Input

- $f_s$: Sampling frequency in Hz.
- $f_p$: Passband edge frequency in Hz.
- $f_{sb}$: Stopband edge frequency in Hz.
- `window`: The type of window function to apply (`'rect'`, `'hann'`, or `'hamming'`).

## Output

- $h[n]$: The final FIR filter coefficients.
- $N$: The estimated filter length (order = length$-1$).
- Plots of the filter's magnitude response, phase response, and impulse response for each window.

## Main Algorithm Steps

1. **Initialization and Parameter Definition:**
   - Define the system parameters: $f_s = 8000\,\text{Hz}$, $f_p = 1000\,\text{Hz}$, and $f_{sb} = 1400\,\text{Hz}$.
   - Create a list of window types to be tested: `windows = ['rect', 'hann', 'hamming']`.
   - Initialize an empty list, `results`, to store the design output for each window type.
2. **Iterative Filter Design Loop:**
   - For each `wname` in the `windows` list:
     (a) Call `design_lpf` with ($f_s, f_p, f_{sb},$ `wname`) to get the filter coefficients $h$ and the length $N$.
     (b) Compute the filter's frequency response: call `freqz(h, worN=4096)` to obtain the frequency array $w$ and the complex frequency response $H$.
     (c) Store the results: append a tuple containing `(wname,` $N$`,` $w$`,` $H$`,` $h$`)` to the `results` list.
3. **Plotting the Filter Characteristics:**
   - Use `freqz` function to generate the frequency response *(Refer the previous labsheets to see how to use* ***freqz****)*
   - Create a single figure for the magnitude response, and impulse response.

- For each result tuple in the `results` list:
  (a) **Magnitude Response Plot:** Plot $20\log_{10}(\max(|H|, 10^{-10}))$ versus $w$ (use $10^{-10}$ to avoid $\log(0)$).
  (b) **Impulse Response Plot:** Plot the filter coefficients $h[n]$ versus the sample index $n$.
- Add appropriate titles, axis labels, legends, and grids to all plots.
- Use `plt.tight_layout()` and `plt.show()` to display all the figures.

## Sub-Algorithms (Function-Level Details)

**1.** `design_lpf(fs, fp, fsb, window)`

(a) Convert input frequencies to digital angular frequencies:

$$\omega_p = \frac{2\pi f_p}{f_s}, \qquad \omega_{sb} = \frac{2\pi f_{sb}}{f_s}.$$

(b) Compute the transition width: $\Delta\omega = \omega_{sb} - \omega_p$.

(c) Estimate the required filter length $N$ by calling `estimate_N(window,` $\Delta\omega$`)`.

(d) Generate the ideal impulse response $h_d[n]$ by calling `ideal_lp(`$\omega_p$`,` $N$`)`.

(e) Apply the window function by calling `apply_window(`$h_d$`, window)` to obtain the final filter coefficients $h[n]$.

(f) Return $h[n]$ and $N$.

**2.** `estimate_N(window, Dw)`

This function estimates the minimum filter length $N$ for a given transition bandwidth $\Delta\omega$.

(a) If `window` is `'rect'`, use the heuristic:

$$N \approx \left\lceil \frac{4\pi}{\Delta\omega} \right\rceil.$$

(b) If `window` is `'hann'` or `'hamming'`, use:

$$N \approx \left\lceil \frac{8\pi}{\Delta\omega} \right\rceil.$$

(c) Round up to the nearest integer using $\lceil \cdot \rceil$.

(d) Ensure $N$ is odd. If $N$ is even, set $N \leftarrow N + 1$ to meet Type-I linear-phase conditions.

(e) Return the final value of $N$.

**3.** `ideal_lp(wc, N)`

This function generates the ideal (sinc) impulse response for a low-pass filter with cutoff $\omega_c$.

(a)) Define the symmetry center $M = (N-1)/2$.

(b)) For each sample index $i = 0, 1, \ldots, N-1$, let $k = i - M$.

(c)) Use

$$h_d[i] = \begin{cases} \dfrac{\omega_c}{\pi}, & k = 0, \\ \dfrac{\sin(\omega_c k)}{\pi k}, & k \neq 0. \end{cases}$$

(d)) Return the array of ideal coefficients $h_d$.

**4.** `apply_window(hd, window)`

This function multiplies the ideal impulse response by the chosen window function.

(a) Let $N = \text{length}(h_d)$.

(b) Generate the window $w[n]$ based on the `window` type:

- If `'rect'`: $w[n] = 1$ for all $n$.
- If `'hann'`: $w[n] = 0.5\left(1 - \cos\left(\dfrac{2\pi n}{N-1}\right)\right)$.
- If `'hamming'`: $w[n] = 0.54 - 0.46\cos\left(\dfrac{2\pi n}{N-1}\right)$.

(c) Perform element-wise multiplication to get the final filter coefficients:

$$h[n] = h_d[n] \cdot w[n], \qquad n = 0, 1, \ldots, N-1.$$

(d) Return the final array $h$.

**PTQ1:** Run the code and plot the magnitude responses of the filters designed with Rectangular, Hann, and Hamming windows in a single figure.

**PTQ2:** Compare the transition width for the three cases. Which window gives the sharpest transition?

**PTQ3:** Plot the impulse responses $h[n]$ for all three windows. Verify whether the responses are symmetric about the midpoint.

**PTQ4:** The filter length $N$ for the Rectangular window is estimated using the formula ____________.

**PTQ5:** For Hann and Hamming windows, the filter length $N$ is approximately ____________.

**PTQ6:** The impulse response of the designed FIR filter is symmetric. This property ensures that the filter has ____________ phase.

**PTQ7:** Increasing the filter length $N$ will make the transition band ________ (narrower / wider).

**PTQ8:** Among Rectangular, Hann, and Hamming windows, the one with the highest stop-band attenuation is ____________.

**PTQ9:** Why is the filter length $N$ different for Rectangular, Hann, and Hamming windows even though the same $f_p$ and $f_{sb}$ are used?

**PTQ10:** From your plots, which window provides the best stopband attenuation? Which one provides the narrowest transition width?

**PTQ11:** Why is it necessary to use a window function instead of directly truncating the ideal sinc impulse response?

# 2. Audio Denoising with Hamming FIR Low-Pass Filter

This algorithm describes a process for audio signal denoising using a Finite Impulse Response (FIR) low-pass filter designed with a Hamming window. The core steps involve filter design, signal filtering, performance evaluation, and visualization.

## Input

- $f_s$: Sampling frequency of the audio signal in Hz.
- $f_p$: Desired passband edge frequency for the filter in Hz.
- $f_{sb}$: Desired stopband edge frequency for the filter in Hz.
- A noisy audio signal, $x[n]$.

## Output

- $y[n]$: The denoised audio signal.
- Plots comparing the spectrum and time-domain waveforms of the original and filtered signals.

## Main Algorithm Steps

1. **Initialization and Data Loading:**
   (a) Define the system's sampling frequency, $f_s = 8000\,\text{Hz}$.
   (b) Attempt to load a noisy audio file named `noisy_speech_8k.wav`. To load the `wav` file

   ```
   import soundfile as sf
   x, fs_file = sf.read("noisy_speech_8k.wav")
   ```

   (c) Use the data in variable `x` for further processing.

2. **FIR Filter Design:**
   (a) Define the passband and stopband frequencies: $f_p = 1000\,\text{Hz}$ and $f_{sb} = 1400\,\text{Hz}$.
   (b) Call the `design_hamming_lpf` function to design the filter. This function automatically selects the filter order $N$ and returns the coefficients $h[n]$.

3. **Signal Filtering:**
   (a) Apply the designed FIR filter to $x[n]$ using a linear filtering function (e.g., `scipy.signal.lfilter`). *Check how to use the function `lfilter` in python.*

   $$y[n] = \sum_{k=0}^{N-1} h[k] \cdot x[n-k]$$

   (b) The output is the denoised signal, $y[n]$.

4. **Spectral and Waveform Comparison (Plotting):**
   (a) **Spectrograms:**

   A spectrogram shows how a signal's spectrum changes over time. It's computed via the Short-Time Fourier Transform (STFT): split the signal into overlapping frames, window each frame (e.g., Hamming), take an FFT of each, then stack the magnitudes over time.

   - Compute the spectrogram for both $x[n]$ and $y[n]$. Use the function `spectrogram` to compute the spectrogram. A sample code for usage is

     ```
     f1, t1, Sx = spectrogram(x, fs=fs, nperseg=256, noverlap=128)
     ```

   - Plot the average power spectral density (PSD) of both signals on a semilogarithmic scale.

     ```
     plt.semilogy(f1, np.mean(Sx, axis=1)+1e-12, label="Before")
     ```

     *(Why are we taking the mean of `Sx` along the first axis direction?)*

   - This demonstrates suppression of high-frequency components by the low-pass filter.

(b) **Waveforms:**

- Create two subplots to show a short segment of the signals in the time domain.
- Plot the original noisy waveform in the top subplot and the filtered waveform in the bottom subplot.
- This comparison allows a visual assessment of noise reduction.

5. **Optional Audio File Saving:**

(a) Attempt to write the filtered signal $y[n]$ to a new WAV file named `denoised_hamming.wav`.

(b) Print a confirmation message upon success or an error message if the operation fails.

**PTQ12:** Plot the waveforms of $x[n]$ and $y[n]$ in the time domain for a short segment.

**PTQ13:** Plot the spectrograms of $x[n]$ and $y[n]$. Identify the regions where noise suppression is most visible.

**PTQ14:** The denoised signal $y[n]$ is obtained by convolving _______________ with _______________.

**PTQ15:** Higher side-lobe attenuation in the frequency response implies better suppression of _______________.

**PTQ16:** If the cutoff frequency $f_p$ is reduced from 1000 Hz to 500 Hz, how will this affect the filtered output signal?

**PTQ17:** Suppose you wanted a high-pass filter instead of a low-pass filter. How would you modify the algorithm?

## Work Done During Lab

(List what you have done in the lab.)

## Analysis and Inference

(Compare the different windows in terms of stop band attenuation, main lobe and side lobe widths, order of the filter. What are the trade-offs?)

## Code Attachment

(Attach your final Python code here.)

**Student Signature:** ______________________ **Date:** ______________