*Heaven’s light is our guide*

RAJSHAHI UNIVERSITY OF ENGINEERING AND TECHNOLOGY

*A logo with text and symbols

AI-generated content may be incorrect.*

**Department Of Electrical & Computer Engineering**

**Course title:**

**Digital Signal Processing Sessional (ECE 4124)**

**Lab Report**

**Submission Date:**

|  |  |
| --- | --- |
| **Submitted To:**  **Md. Faysal Ahmed**  **Assistant Professor &**  **Moloy Kumar Ghosh**  **Lecturer**  **Dept of ECE, RUET** | **Submitted By:**  **Syed Mahmudul Imran**  **Roll: 2010058** |

**Experiment No. 6**

**Experiment Name:** Design and Analysis of a Low-Pass FIR and IIR Filter and Comparison of Their Frequency Responses

**Theory:**

In **Digital Signal Processing (DSP)**, filters are essential tools used to extract, suppress, or modify specific frequency components of a discrete-time signal. Among various types, the **low-pass filter (LPF)** is one of the most fundamental. It allows frequencies below a specified **cutoff frequency (fₐ)** to pass while attenuating higher frequencies. Such filters are crucial in applications like noise reduction, signal smoothing, and data reconstruction.

Filters in DSP are broadly categorized into two types — **Finite Impulse Response (FIR)** and **Infinite Impulse Response (IIR)** — based on the duration of their impulse responses and structural characteristics.

**1. FIR Filter (Finite Impulse Response):**

An FIR filter produces an output that depends only on the current and a finite number of past input samples. Its impulse response settles to zero after a finite time. The general difference equation is:

where are the filter coefficients and is the filter order.

**Key Characteristics:**

* Always **stable** since it has no feedback component.
* Can be designed to exhibit a **linear phase response**, preserving the shape of time-domain waveforms—important for audio, biomedical, and communication systems.
* Requires higher order for sharp transitions, leading to increased computational load.

**2. IIR Filter (Infinite Impulse Response):**

An IIR filter has a recursive structure where the output depends on both past inputs and past outputs, leading to an impulse response of theoretically infinite duration. Its general difference equation is:

**Key Characteristics:**

* More **computationally efficient** than FIR filters, achieving sharp cutoff characteristics with lower filter order.
* May exhibit **non-linear phase response**, which can distort time-domain signals.
* **Potentially unstable** if pole locations are not carefully chosen within the unit circle in the z-plane.

**3. Frequency Response:**

The frequency response of a discrete-time filter describes how each frequency component of the input signal is modified. It is derived from the filter’s **transfer function** , with the magnitude response expressed as:

This represents the amplitude gain across different frequencies, allowing visualization of how effectively the filter passes or attenuates components.

**4. Design Specification:**

For comparative analysis, both **FIR** and **IIR** filters are designed with identical parameters:

* **Cutoff Frequency (fₐ):** 100 Hz
* **Sampling Frequency (fₛ):** 1000 Hz

This ensures a fair comparison of their **magnitude responses**, stability, computational complexity, and phase characteristics. Through such analysis, the trade-offs between **stability, accuracy, and efficiency** in filter design can be clearly demonstrated.

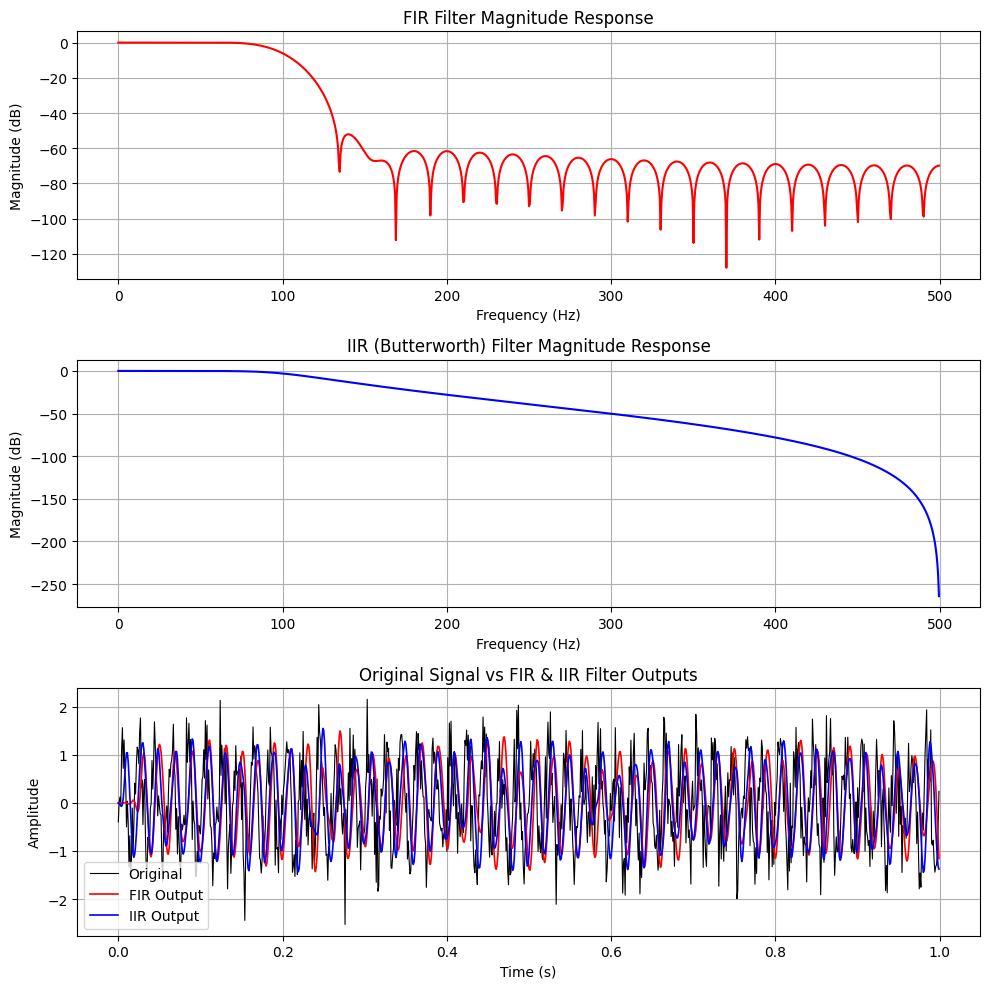
**Required Tools:**

1. Python
2. VS Code
3. MS Office

**Code:**

|  |  |
| --- | --- |
| import numpy as np  import matplotlib.pyplot as plt  from scipy.signal import butter, firwin, freqz, lfilter  # Filter and Signal Parameters  fs = 1000          # Sampling frequency (Hz)  fcut = 100         # Cutoff frequency (Hz)  n = 50             # FIR filter order  t = np.arange(0, 1, 1/fs)  # 1-second time vector  # FIR Filter Design  b\_fir = firwin(numtaps=n+1, cutoff=fcut, fs=fs)  w\_fir, H\_fir = freqz(b\_fir, worN=1024, fs=fs)  # IIR Filter Design (Butterworth)  b\_iir, a\_iir = butter(4, fcut/(fs/2), btype='low')  w\_iir, H\_iir = freqz(b\_iir, a\_iir, worN=1024, fs=fs)  # Generate Input Signal  x = np.sin(2\*np.pi\*50\*t) + 0.5\*np.random.randn(len(t))  # 50 Hz + noise  # Apply Filters  y\_fir = lfilter(b\_fir, 1, x)  y\_iir = lfilter(b\_iir, a\_iir, x)  # Plot Results  plt.figure(figsize=(10, 10)) | # --- Subplot 1: FIR Frequency Response ---  plt.subplot(3, 1, 1)  plt.plot(w\_fir, 20\*np.log10(abs(H\_fir)), 'r', linewidth=1.5)  plt.title('FIR Filter Magnitude Response')  plt.xlabel('Frequency (Hz)')  plt.ylabel('Magnitude (dB)')  plt.grid(True)  # --- Subplot 2: IIR Frequency Response ---  plt.subplot(3, 1, 2)  plt.plot(w\_iir, 20\*np.log10(abs(H\_iir)), 'b', linewidth=1.5)  plt.title('IIR (Butterworth) Filter Magnitude Response')  plt.xlabel('Frequency (Hz)')  plt.ylabel('Magnitude (dB)')  plt.grid(True)  # --- Subplot 3: Time-Domain Comparison ---  plt.subplot(3, 1, 3)  plt.plot(t, x, 'k', linewidth=0.8, label='Original')  plt.plot(t, y\_fir, 'r', linewidth=1.2, label='FIR Output')  plt.plot(t, y\_iir, 'b', linewidth=1.2, label='IIR Output')  plt.title('Original Signal vs FIR & IIR Filter Outputs')  plt.xlabel('Time (s)')  plt.ylabel('Amplitude')  plt.legend()  plt.grid(True)  plt.tight\_layout()  plt.show() |

**Output:**

****

**Result:**  
Both FIR and IIR filters effectively removed noise components above the 100 Hz cutoff frequency while preserving the low-frequency (≈ 50 Hz) signal. The **FIR filter** exhibited a linear phase response but had a wider transition band and minor stopband ripples. The **IIR (Butterworth) filter** achieved a sharper cutoff and smoother magnitude response with fewer coefficients, though it introduced slight phase distortion due to its recursive structure. Overall, both filters demonstrated efficient low-pass filtering performance.

**Discussion:**  
The experiment highlights clear distinctions between FIR and IIR filter characteristics. The **FIR filter**, designed using the window method, ensured linear phase accuracy, making it suitable for audio and data-sensitive applications, but required higher order for sharper transitions. Conversely, the **IIR filter** provided a steeper roll-off with lower computational demand, at the expense of phase linearity. The comparative results confirmed that FIR filters are more phase-accurate but computationally intensive, while IIR filters are more efficient but exhibit phase distortion, reflecting the trade-off between precision and efficiency in DSP design.

**Conclusion:**  
This study successfully demonstrated the design and comparison of FIR and IIR low-pass filters. Both effectively suppressed high-frequency noise and retained the desired low-frequency content. The **FIR filter** offered stable, linear-phase performance, whereas the **IIR (Butterworth) filter** achieved better frequency selectivity with reduced computational cost. The results emphasize that the choice between FIR and IIR designs depends on application requirements—balancing **accuracy, stability, and computational efficiency** in practical DSP systems such as audio processing, signal denoising, and biomedical analysis.

**References:**

[1] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed., Pearson Education, 2010.

[2] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*, 4th ed., Prentice Hall, 2007.

[3] “Signal Processing Toolbox Documentation,” MathWorks, 2025. [Online]. Available: <https://www.mathworks.com/help/signal>. [Accessed: Oct. 23, 2025].

[4] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, 4th ed., McGraw-Hill, 2011.