Nathaniel Rogers
March 13, 2026

Final BME 308 Lab: Active vs Passive Low-Pass Filtered ECG

Objective

The objective of this lab was to compare the sharpness of a passive low-pass filter with an active Sallen-Key second-order low-pass filter when applied to an electrocardiogram (ECG) signal. High-frequency noise can degrade the quality and diagnostic usefulness of an ECG signal, so a sharper low-pass filter would be ideal for real-world applications. If a circuit can be built with outputs for both kinds of low-pass filter, it will allow the simultaneous data collection from both filters, which can then be compared using a fast Fourier transform (FFT) to quantify the relative intensity of high-frequency components.

Background Research

As with the previous ECG lab, the raw signal will be too small (in the millivolt range) to see in the presence of noise, so, in addition to filtering, two stages of amplification (yielding a total gain of ~10,000) are necessary. When simulating the frequency response behavior in LTSpice, it will also be valuable to simulate frequencies that will likely be found in an ECG signal, so A/C frequencies of 50 mHz (slow sensor drifts) to 100 Hz (noise) will be plotted. Finally, the Sallen-Key second-order low-pass filter uses the architecture laid out in [1], in which the cutoff frequency of the filter is:

$$f = \frac{1}{2\pi\sqrt{R_1 R_2 C_1 C_2}}$$

Block Diagram

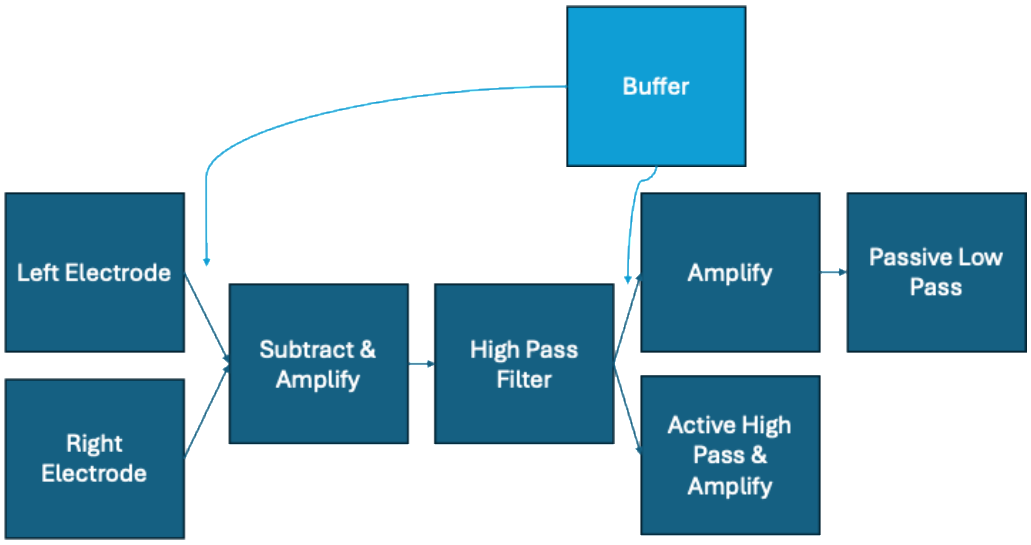

Figure 1: Block Diagram of Circuit

Circuit Diagram

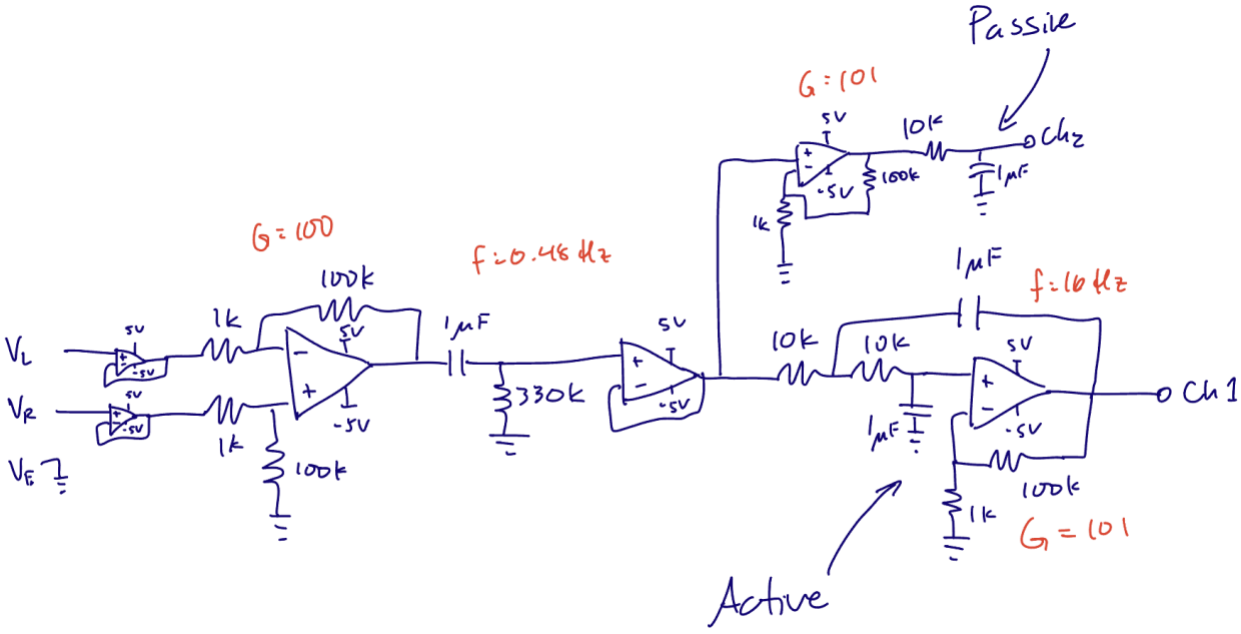

Figure 2: Complete Circuit Diagram

Simulated Frequency Response

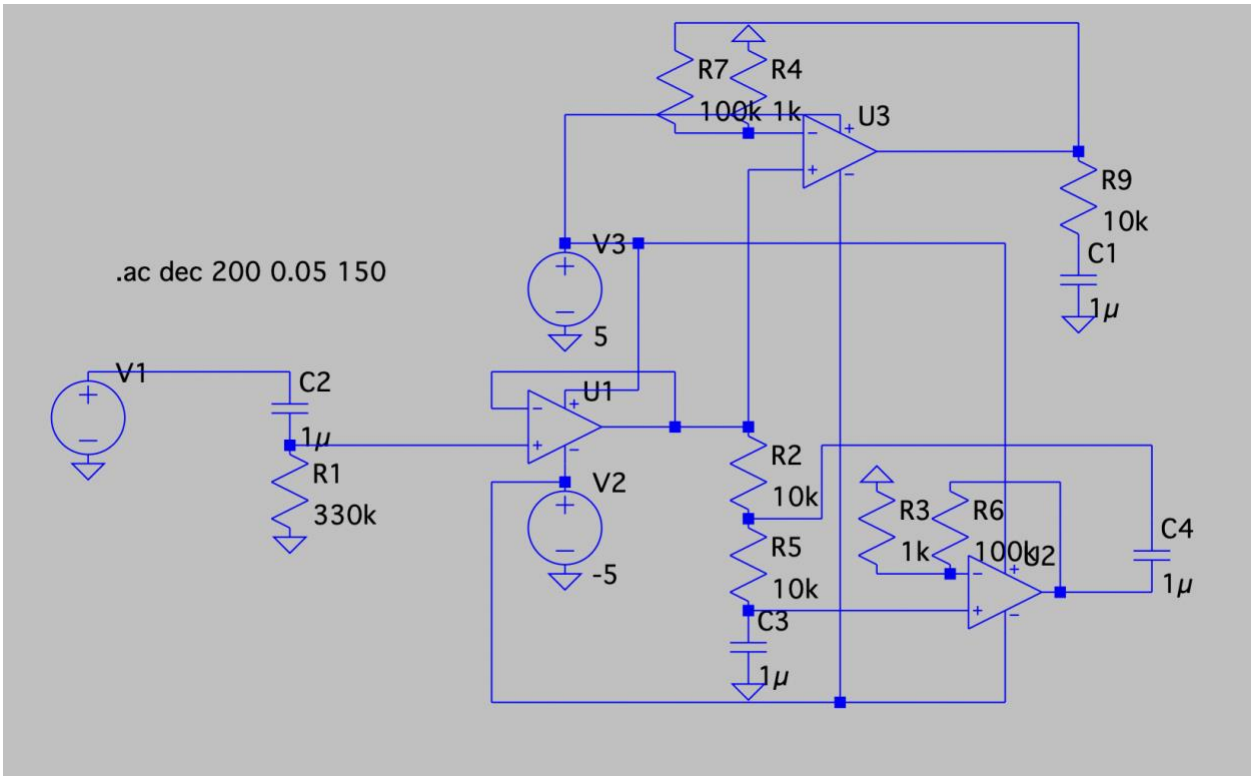

Figure 3: LTSpice Circuit Diagram (showing circuit after initial amplification)

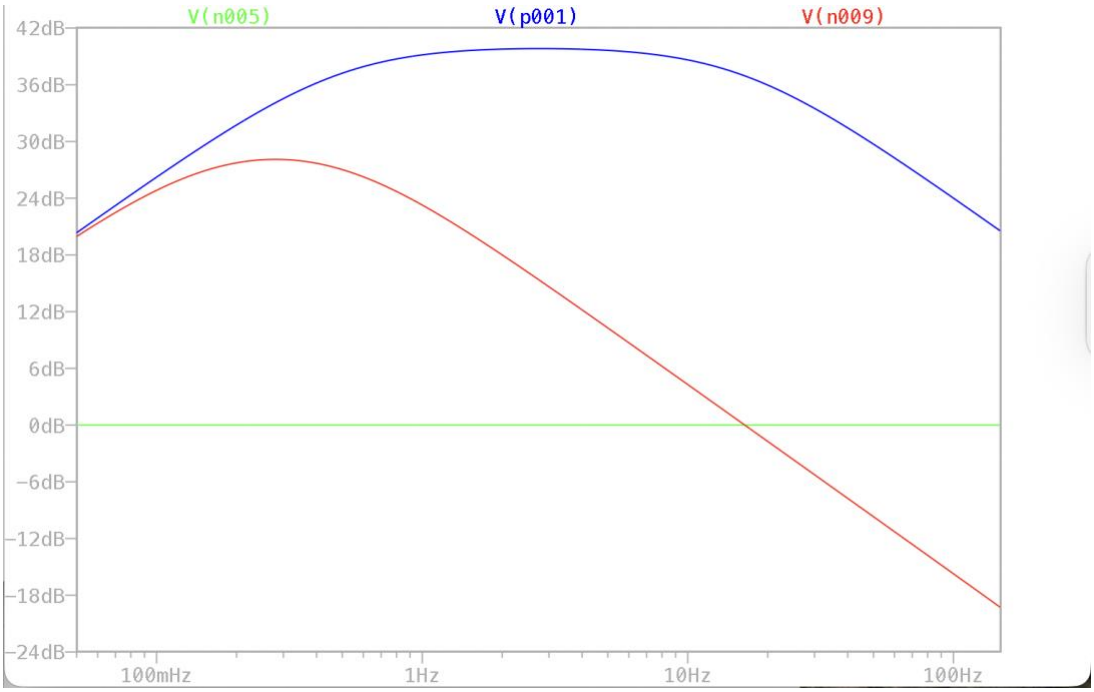

Figure 4: Frequency Response Curves (green ~ source, blue ~ passive low pass filter, red ~ active low pass filter)

Results

As expected, in the LTSpice frequency response shown in *Figure 4,* the active low-pass filter has a sharper drop-off after the cutoff frequency than the passive filter, suggesting higher-frequency ECG components will be more successfully filtered out using the active filter than the passive one. The active high-pass filter also has lower overall intensity, despite having the same mathematical gain. Both of these behaviors can be seen in the different ECG traces shown in *Figure 5,* in which the actively filtered trace appears noticeably smoother and less intense than the passively filtered trace. This difference in frequency response can also be qualitatively confirmed by the FFT plots shown in *Figure 6*, in which the peak intensities of high-frequency components are noticeably higher for the passive filter than those for the active filter.

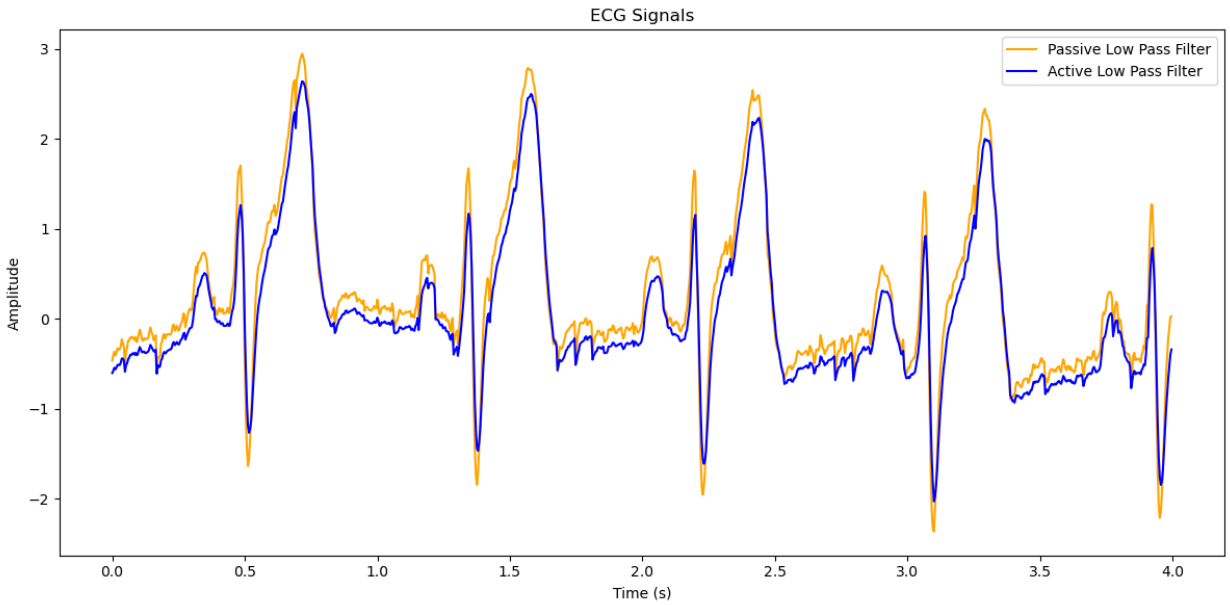

Figure 5: Filtered ECG Traces

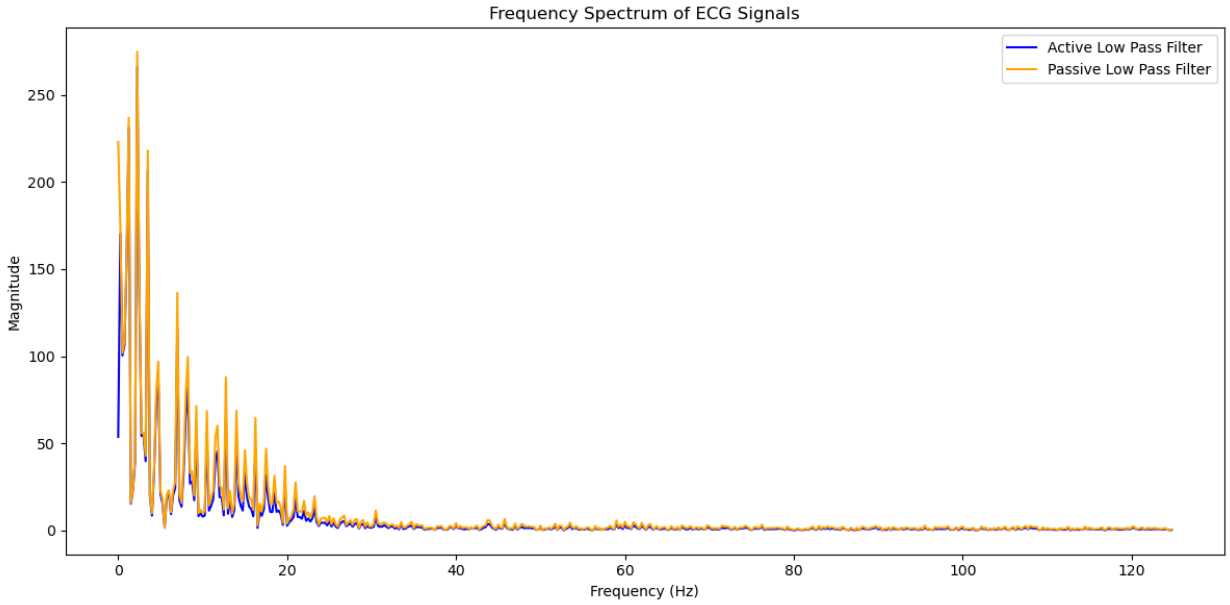

Figure 6: FFT Plot

References

[1] https://www.electronics-tutorials.ws/filter/second-order-filters.html\

Appendix: Python Code for ECG Data Collection and Processing

```
import matplotlib.pyplot as plt
import numpy as np
from nlabapi import LabBench, AnalogSignalPolarity
import time

SAMPLE_RATE = 250  # Hz

def get_ecg_data():
    nlab = LabBench.open_first_available()

    number_of_samples = SAMPLE_RATE * 4 #25 seconds of data

    print("Will record "+str(number_of_samples)+" samples at "+str(SAMPLE_RATE)+"Hz for
"+str(number_of_samples/SAMPLE_RATE)+" seconds")

    time.sleep(1)
    print("Recording starts in 3 seconds...")
    time.sleep(1)
    print("Recording starts in 2 seconds...")
    time.sleep(1)
    print("Recording starts in 1 seconds...")
    time.sleep(1)
    print("Recording!")

    data = nlab.read_all_channels(SAMPLE_RATE, number_of_samples)

    print("Recording complete")

    x1 = data[0]
    x2 = data[1]
    x3 = data[2]
    x4 = data[3]
    t = np.arange(number_of_samples)/SAMPLE_RATE

    return t, x1, x2, x3, x4
```

```
def plot_all(data):
    t, x1, x2, x3, x4 = data

    # Plotting the signals as lines on a single plot with stars on the peaks of
    plt.figure(figsize=(12, 6))

    plt.plot(t, x2, label='Passive Low Pass Filter', color='orange')
    plt.plot(t, x1, label='Active Low Pass Filter', color='blue')

    plt.xlabel('Time (s)')
    plt.ylabel('Amplitude')
    plt.title('ECG Signals')
    plt.legend()

    plt.tight_layout()
    plt.show()

def compare_high_frequency_noise(data):

    t, x1, x2, x3, x4 = data

    # calculate fft of x1 and x2
    fft_x1 = np.fft.fft(x1)
    fft_x2 = np.fft.fft(x2)

    # calculate frequency bins
    n = len(x1)
    freq = np.fft.fftfreq(n, d=1/SAMPLE_RATE)
    # Plot the frequency spectra
    plt.figure(figsize=(12, 6))
    plt.plot(freq[:n//2], np.abs(fft_x1)[:n//2], label='Active Low Pass Filter', color='blue')
    plt.plot(freq[:n//2], np.abs(fft_x2)[:n//2], label='Passive Low Pass Filter', color='orange')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Magnitude')
    plt.title('Frequency Spectrum of ECG Signals')
    plt.legend()
    plt.tight_layout()
```

```
    plt.show()

    #plot the difference in magnitude between to two signals vs frequency
    plt.figure(figsize=(12, 6))
    plt.plot(freq[:n//2], np.abs(fft_x1)[:n//2] - np.abs(fft_x2)[:n//2], label='Difference in Magnitude', color='green')
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Magnitude Difference')
    plt.title('Difference in Frequency Spectrum of ECG Signals')
    plt.legend()
    plt.tight_layout()
    plt.show()

def main():
    data = get_ecg_data()
    plot_all(data)
    compare_high_frequency_noise(data)

if __name__ == "__main__":
    main()
```