REPORT

Course: Analog and digital electronic circuits
Teacher: Prof. Dr. Hab. Vasyl Martsenyuk

Lab No. 5 and 6
Date: January 20, 2025
Topic: "Filtering"
Variant: 8

Name: Daria Kręcichwost
Computer Science (Second Degree)
Part-time studies, Semester 1
Group: B

5. Digital Filter Design and Analysis: Implementing FIR and IIR filters in Python. 6. Adaptive Filtering: Applying adaptive filtering algorithms to noise reduction.

Daria Kręcichwost

January 20, 2025

1 Problem Statement

The task is to design and implement three different types of filters to reduce noise in a noisy sinusoidal signal:

1. Design a Finite Impulse Response (FIR) filter with the given coefficients $b = [1, 0, 1]$.
2. Design an Infinite Impulse Response (IIR) filter with the given coefficients $b = [0.5, 0.5]$ and $a = [1, -0.3]$.
3. Implement an Adaptive Least Mean Squares (LMS) filter with a step size $\mu = 0.05$ and filter length $M = 5$.

2 Input Data

The input signal for the task is a noisy sinusoidal signal. The sinusoidal signal has a frequency of 5 Hz and is sampled at a frequency of 1000 Hz. Gaussian noise is added to this sinusoidal signal to simulate a noisy signal.

$$x(t) = \sin(2\pi f t) + \text{Noise}$$

where $f = 5$ Hz.

The desired signal is the clean sinusoidal signal without noise.

3 Commands Used

3.1 Source Code

The source code for the filter implementations is written in Python and uses the NumPy library for numerical computations and Matplotlib for plotting. Below is a summary of the code sections for the FIR, IIR, and LMS filters.

Link to remote repository on GitHub

3.1.1 FIR Filter

The FIR filter uses the coefficients $b = [1, 0, 1]$ and convolves the input signal with the filter coefficients.

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

def fir_filter(x, b):
    M = len(b)
    y = np.zeros(len(x))
    for n in range(M, len(x)):
        y[n] = np.dot(b, x[n-M+1:n+1][::-1])
    return y
```

3.1.2 IIR Filter

The IIR filter uses the coefficients $b = [0.5, 0.5]$ and $a = [1, -0.3]$. It applies both feedforward and feedback operations to calculate the output signal.

```
def iir_filter(x, b, a):
    M = len(b)  # Length of numerator coefficients (b)
    N = len(a)  # Length of denominator coefficients (a)
    y = np.zeros(len(x))  # Initialize output signal array

    for n in range(len(x)):
        x_slice = x[max(0, n-M+1):n+1]  # Input signal slice
        y[n] = np.dot(b[:len(x_slice)], x_slice[::-1])

        if n >= 1:
            y_slice = y[max(0, n-N+1):n]
            y[n] -= np.dot(a[1:], y_slice[::-1])

    return y
```

3.1.3 LMS Filter

The LMS filter is an adaptive filter that adjusts its weights based on the error between the desired and actual output.

```
def lms_filter(x, d, mu, num_taps):
    n = len(x)
    w = np.zeros(num_taps)
    y = np.zeros(n)
    e = np.zeros(n)

    for i in range(num_taps, n):
        x_segment = x[i-num_taps:i][::-1]
        y[i] = np.dot(w, x_segment)
        e[i] = d[i] - y[i]
        w += mu * e[i] * x_segment

    return y, e, w
```

3.2 Screenshots

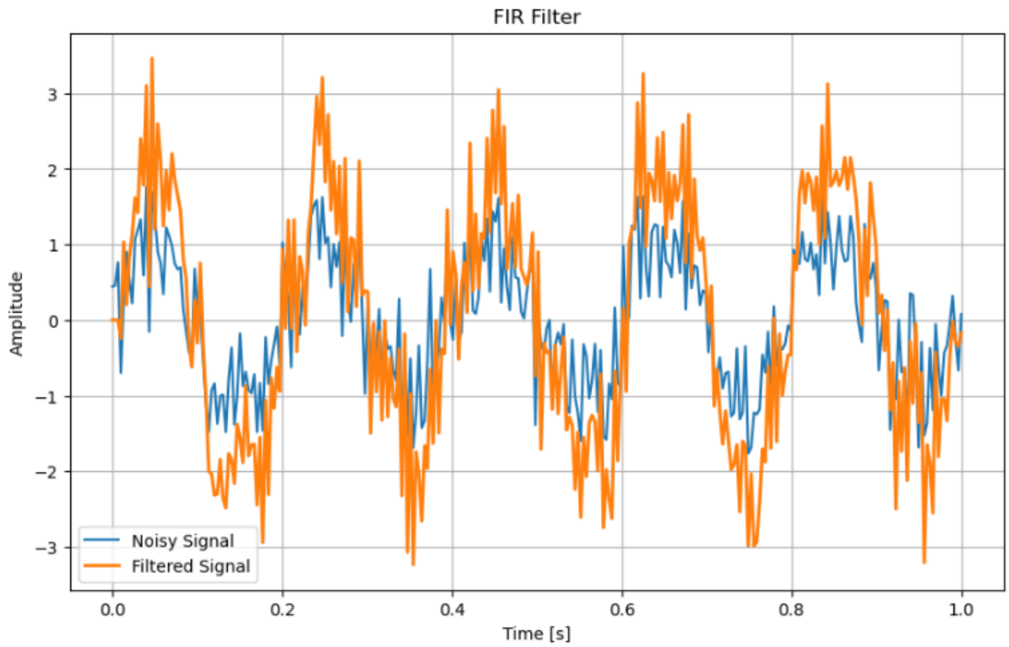

Figure 1: FIR Filter Response

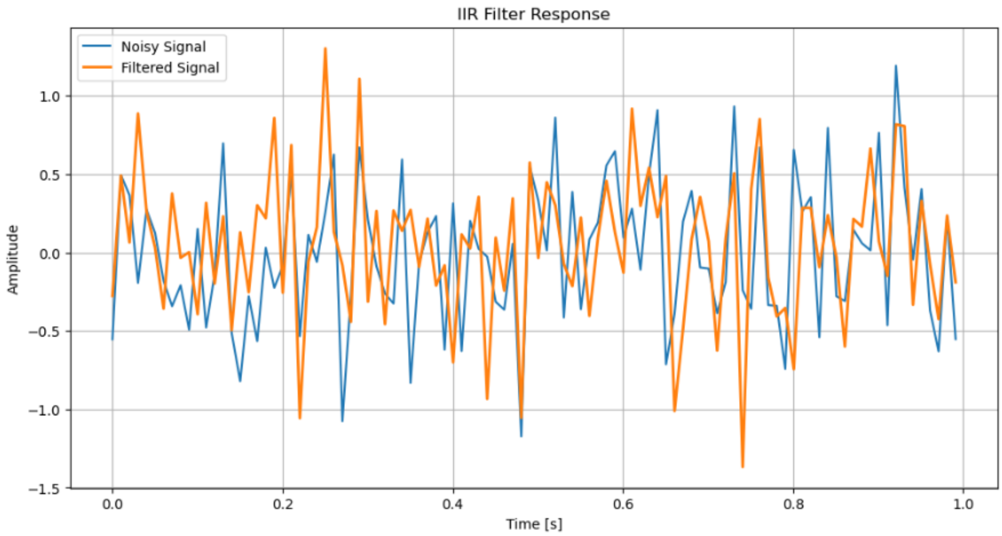

Figure 2: IIR Filter Response

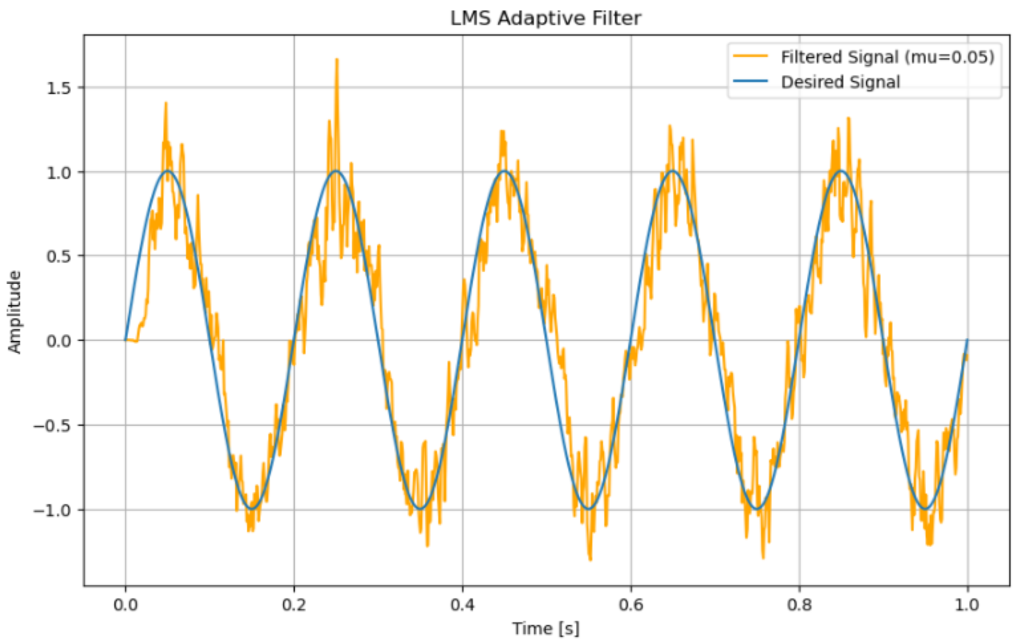

Figure 3: LMS Filter Response

4 Outcomes

The results obtained from the FIR, IIR, and LMS filters are as follows:

1. The FIR filter successfully reduces high-frequency noise, although some residual noise remains.
2. The IIR filter provides a smoother output, especially for higher frequencies.
3. The LMS adaptive filter is able to adjust dynamically to the noise, producing a clean signal by minimizing the error between the desired and actual signal.

5 Conclusions

1. The FIR filter is effective for reducing noise but may not perform as well as the IIR and LMS filters in all cases.

2. The IIR filter provides a more efficient solution for noise reduction with less residual noise compared to the FIR filter.

3. The LMS adaptive filter performs the best in terms of adaptability, adjusting the filter parameters to minimize the error and providing the cleanest output signal.