REPORT

Zajęcia: Analog and digital electronic circuits
Teacher: prof. dr hab. Vasyl Martsenyuk

Lab 5&6 - Design and analysis of digital filters: implementation of FIR and IIR filters in Matlab/Python. Adaptive filtering: application of adaptive filtering algorithms in noise reduction.
Date: 23.11.2024
Topic: Design and Analysis of Digital Filters: FIR, IIR, and Adaptive Filters
Variant DSP 7

Dawid Klimek
Informatyka II stopień,
niestacjonarne,
1 semestr,
Gr.A

1. Problem statement:

1. Problem statement:

The goal of this task is to design and implement digital filters in Python to reduce noise in a noisy sinusoidal signal. Three types of filters will be implemented:

1. A Finite Impulse Response (FIR) filter with coefficients b = {1, -1, 0.5}.
2. An Infinite Impulse Response (IIR) filter with coefficients b = {0.3, 0.4} and a = {1, -0.5, 0.2}.
3. An adaptive Least Mean Squares (LMS) filter with a step size mu = 0.05 and filter length M = 4.

Each filter will be applied to the same noisy sinusoidal signal, and their effectiveness in noise reduction will be analyzed.

2. Input data:

(FIR) filter with coefficients b = {1, -1, 0.5}.
(IIR) filter with coefficients b = {0.3, 0.4} and a = {1, -0.5, 0.2}.
LMS) filter with a step size mu = 0.05 and filter length M = 4.

3. Commands used (or GUI):

a) source code;

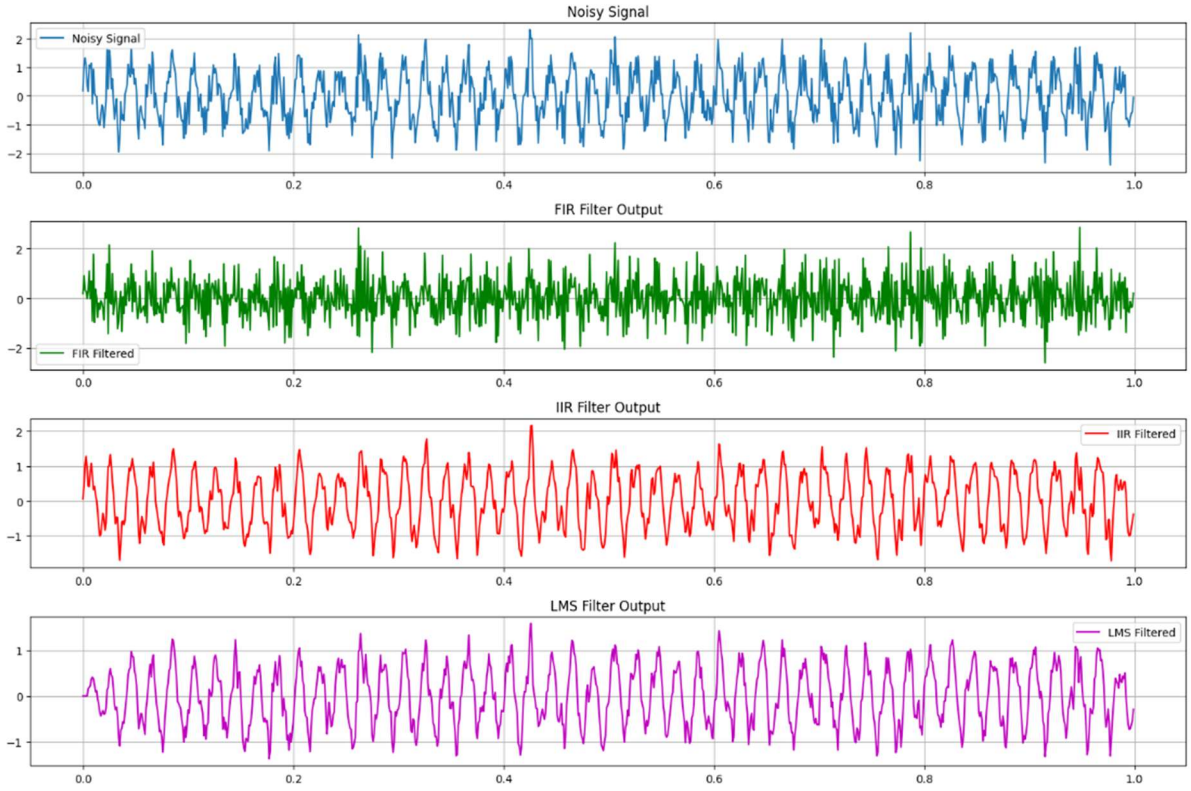
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import lfilter

# Generowanie zakłóconego sygnału sinusoidalnego
fs = 1000  # Częstotliwość próbkowania
t = np.linspace(0, 1, fs, endpoint=False)
f = 50  # Częstotliwość sygnału
signal = np.sin(2 * np.pi * f * t)  # Sygnał czysty
noise = np.random.normal(0, 0.5, signal.shape)  # Szum
noisy_signal = signal + noise  # Sygnał zakłócony

# FIR Filter
b_fir = [1, -1, 0.5]
filtered_fir = lfilter(b_fir, [1], noisy_signal)

# IIR Filter
b_iir = [0.3, 0.4]
a_iir = [1, -0.5, 0.2]
filtered_iir = lfilter(b_iir, a_iir, noisy_signal)
```

```
# LMS Adaptive Filter
def lms_filter(x, d, mu, M):
    N = len(x)
    w = np.zeros(M)
    y = np.zeros(N)
    e = np.zeros(N)

    for n in range(M, N):
        x_n = x[n:n-M:-1]  # Fragment sygnału wejściowego
        y[n] = np.dot(w, x_n)
        e[n] = d[n] - y[n]
        w += mu * e[n] * x_n

    return y, e

mu = 0.05
M = 4
desired_signal = signal  # Sygnał czysty jako odniesienie
filtered_lms, error = lms_filter(noisy_signal, desired_signal, mu, M)

# Wizualizacja wyników
plt.figure(figsize=(15, 10))

plt.subplot(4, 1, 1)
plt.plot(t, noisy_signal, label='Noisy Signal')
plt.title("Noisy Signal")
plt.grid()
plt.legend()

plt.subplot(4, 1, 2)
plt.plot(t, filtered_fir, label='FIR Filtered', color='g')
plt.title("FIR Filter Output")
plt.grid()
plt.legend()

plt.subplot(4, 1, 3)
plt.plot(t, filtered_iir, label='IIR Filtered', color='r')
plt.title("IIR Filter Output")
plt.grid()
plt.legend()

plt.subplot(4, 1, 4)
plt.plot(t, filtered_lms, label='LMS Filtered', color='m')
plt.title("LMS Filter Output")
```

```
plt.grid()
plt.legend()

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

https://github.com/skruty/DSP_UBB

4. Outcomes:



5. Conclusions:

The implementation and analysis of FIR, IIR, and adaptive LMS filters demonstrate their effectiveness in reducing noise in a noisy sinusoidal signal. Each filter has distinct characteristics and use cases:

1. **FIR Filter**:
   The FIR filter effectively reduces noise without introducing phase distortion, making it ideal for applications where maintaining the signal's phase integrity is crucial. However, its performance depends heavily on the choice of coefficients and may require a higher filter order to achieve sharp cutoffs.

2. **IIR Filter**:
   The IIR filter offers better frequency selectivity and requires fewer coefficients to achieve similar results compared to FIR filters. However, it introduces phase distortion and requires careful design to ensure stability, particularly for higher filter orders.
3. **LMS Adaptive Filter**:
   The LMS adaptive filter is highly effective in dynamic environments where noise characteristics change over time. By continuously adapting to the signal, it can reduce noise efficiently, although its performance depends on the step size (mu) and filter length (M). Selecting these parameters appropriately is critical to balance convergence speed and steady-state error.

In conclusion, the choice of filter depends on the specific requirements of the application, such as noise characteristics, computational efficiency, and the need for phase preservation. This study highlights the importance of understanding the trade-offs between different filter types to optimize performance in digital signal processing tasks.