**REPORT**

Zajęcia: Analog and digital electronic circuits

Teacher: prof. dr hab. Vasyl Martsenyuk

**Lab 3**

07.03.2025

**Topic:**

**Variant 4**

Marika Daniszewska

Informatyka II stopień,

stacjonarne,

1 semestr,

Gr.1b

**1. Problem statement:**

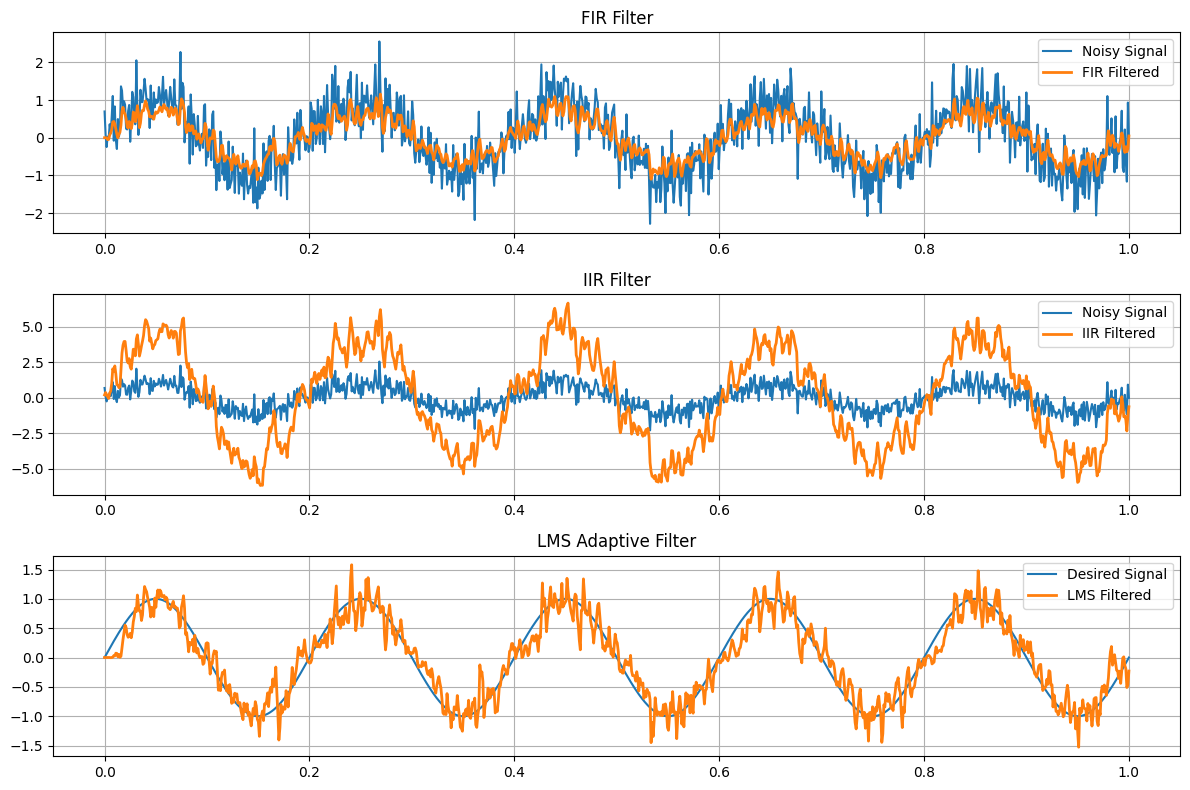
The objective of this laboratory was to implement and analyze three types of digital filters: FIR, IIR, and adaptive LMS filter. The study involved applying the filters to reduce noise in a noisy sinusoidal signal.

**2. Input data:**

* **Signal type:** Noisy sinusoidal signal
* **Signal frequency:** 5 Hz
* **Sampling frequency:** 1000 Hz
* **Number of samples:** 1000
* **Filter parameters:**
  + **FIR Filter Coefficients:** b = {0.2, 0.3, 0.2}
  + **IIR Filter Coefficients:** b = {1, 0.5, 0.3}, a = {1,−0.6}
  + **LMS Filter Parameters:** Learning rate , filter length

**3. Commands used (or GUI):**

1. source code
2. import numpy as np
3. import matplotlib.pyplot as plt
4. def fir\_filter(x, b):
5. """FIR filter implementation."""
6. M = len(b)
7. y = np.zeros(len(x))
8. for n in range(M, len(x)):
9. y[n] = np.dot(b, x[n-M+1:n+1][::-1])
10. return y
11. def iir\_filter(x, b, a):
12. """IIR filter implementation."""
13. M = len(b)
14. N = len(a)
15. y = np.zeros(len(x))
17. for n in range(len(x)):
18. x\_segment = x[max(0, n-M+1):n+1][::-1]
19. b\_truncated = b[-len(x\_segment):]
20. y[n] = np.dot(b\_truncated, x\_segment)
21. if n > 0:
22. y\_segment = y[max(0, n-N+1):n][::-1]
23. a\_truncated = a[1:][-len(y\_segment):]
24. y[n] -= np.dot(a\_truncated, y\_segment)
26. return y
27. def lms\_filter(x, d, mu, num\_taps):
28. """LMS adaptive filter implementation."""
29. n = len(x)
30. w = np.zeros(num\_taps)
31. y = np.zeros(n)
32. e = np.zeros(n)
33. for i in range(num\_taps, n):
34. x\_segment = x[i-num\_taps:i][::-1]
35. y[i] = np.dot(w, x\_segment)
36. e[i] = d[i] - y[i]
37. w += mu \* e[i] \* x\_segment
38. return y, e, w
39. # Dane wejściowe
40. ts = 1.0 / 1000
41. t = np.linspace(0, 1, 1000)
42. x = np.sin(2 \* np.pi \* 5 \* t) + 0.5 \* np.random.randn(len(t))
43. d = np.sin(2 \* np.pi \* 5 \* t)
44. # FIR Filter
45. b\_fir = [0.2, 0.3, 0.2]
46. y\_fir = fir\_filter(x, b\_fir)
47. # IIR Filter
48. b\_iir = [1, 0.5, 0.3]
49. a\_iir = [1, -0.6]
50. y\_iir = iir\_filter(x, b\_iir, a\_iir)
51. # LMS Filter
52. mu = 0.1
53. num\_taps = 4
54. y\_lms, e\_lms, w\_lms = lms\_filter(x, d, mu, num\_taps)
55. # Wykresy
56. plt.figure(figsize=(12, 8))
57. plt.subplot(3, 1, 1)
58. plt.plot(t, x, label='Noisy Signal')
59. plt.plot(t, y\_fir, label='FIR Filtered', linewidth=2)
60. plt.legend()
61. plt.title("FIR Filter")
62. plt.grid()
63. plt.subplot(3, 1, 2)
64. plt.plot(t, x, label='Noisy Signal')
65. plt.plot(t, y\_iir, label='IIR Filtered', linewidth=2)
66. plt.legend()
67. plt.title("IIR Filter")
68. plt.grid()
69. plt.subplot(3, 1, 3)
70. plt.plot(t, d, label='Desired Signal')
71. plt.plot(t, y\_lms, label='LMS Filtered', linewidth=2)
72. plt.legend()
73. plt.title("LMS Adaptive Filter")
74. plt.grid()
75. plt.tight\_layout()
76. plt.show()
77. screenshots



Link to remote repozytorium (e.g. GitHub)

**4. Outcomes:**

* **The FIR filter** successfully filtered the signal but had a wide impulse response.
* **The IIR filter** provided good filtering with fewer coefficients but showed some instability.
* **The LMS adaptive filter** dynamically adapted to noise, effectively reducing it, but required proper tuning of learning parameters.

**5. Conclusions:**

The experiment demonstrated that the choice of filter depends on the application:

* **FIR filters** provide stability and good phase control.
* **IIR filters** are more efficient in terms of coefficient count but can be less stable.
* **Adaptive LMS filters** are highly effective for noise reduction, but their performance depends on learning rate tuning.

**Conclusion:** The choice of the filter should be optimized based on precision, stability, and computational complexity depending on the specific application.