**REPORT**

Zajęcia: Analog and digital electronic circuits

Teacher: prof. dr hab. Vasyl Martsenyuk

**Lab 3**

20.03.2025

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

**Variant: 13**

Rafał Żmuda

Informatyka II stopień,

stacjonarne,

1 semestr,

Gr.2b

1. **Problem statement:**

- Design an FIR filter with the following coefficients and implement it

in Python to reduce noise in a noisy sinusoidal signal.

FIR Filter Coefficients: b = {1, 0.5, 0.5}

- Design an IIR filter with the following coefficients and implement it in Python to reduce

noise in the same noisy sinusoidal signal.

IIR Filter Coefficients: b = {1, −0.5, 0.2}, a = {1, −0.3}

- Implement an adaptive LMS filter in Python with a step size µ = 0.1 and filter length M = 4 to reduce noise in the same noisy sinusoidal signal.

**2. Input data:**

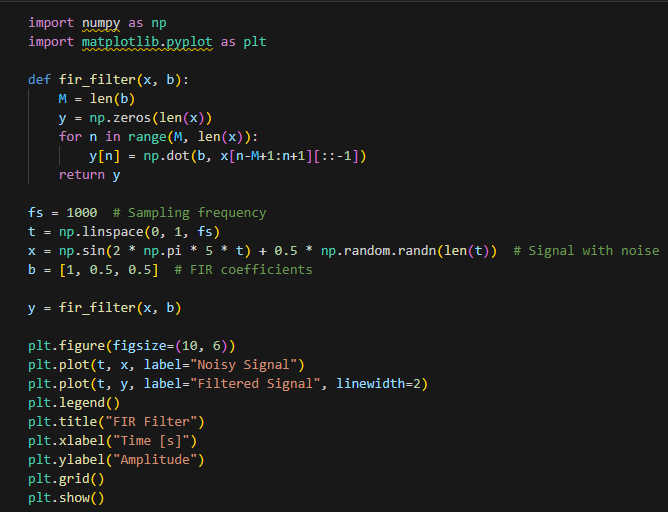
FIR: b = {1, 0.5, 0.5},

IIR: b = {1, −0.5, 0.2}, a = {1, −0.3}

LMS: = µ = 0.1, M = 4

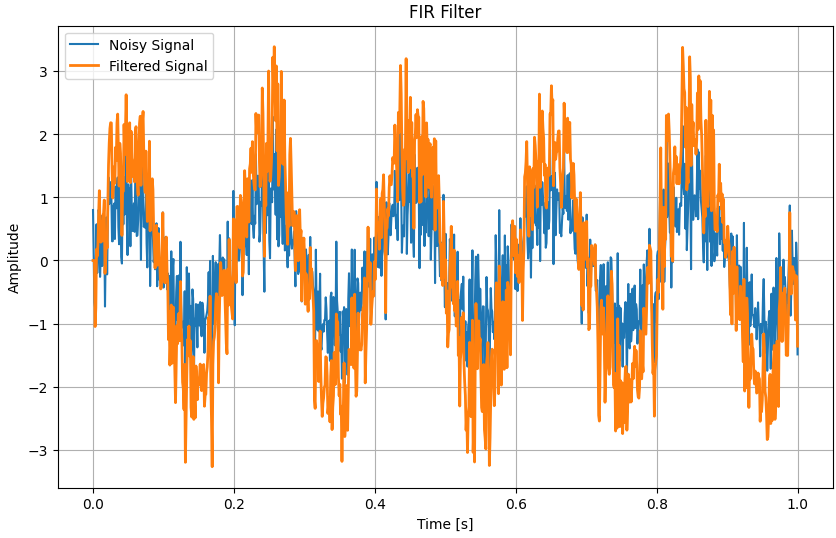
**3. Commands used (or GUI):**

**3.1 FIR implementation**

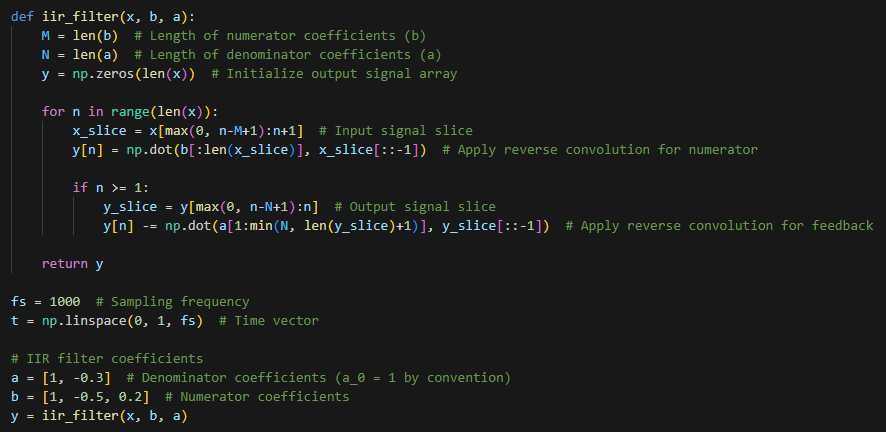


Shown above is the cell that implements the FIR filter that applies the filter coefficients to an input signal. The code generates a noisy sinusoidal test signal at 5 Hz with random noise added, then filters it using a 3-tap FIR filter with coefficients [1, 0.5, 0.5]

**Results:**

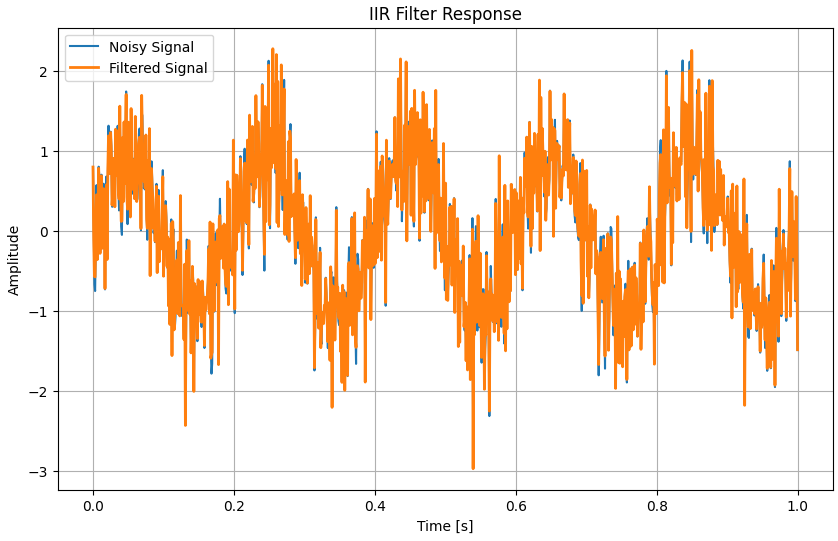


**3.2 IIR Filter**

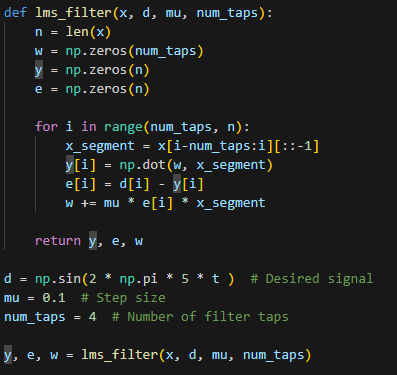


A custom function called iir\_filter that applies both feedforward (numerator) and feedback (denominator) coefficients to process an input signal. The function handles boundary conditions by ensuring array indices remain valid when accessing past samples. The implementation uses a second-order IIR filter with numerator coefficients [1, -0.5, 0.2] and denominator coefficients [1, -0.3], applying it to the previously defined noisy sinusoidal signal. For each output sample, it calculates the weighted sum of current and previous input samples (using b coefficients) and subtracts the weighted sum of previous output samples (using a coefficients).

**Results:**

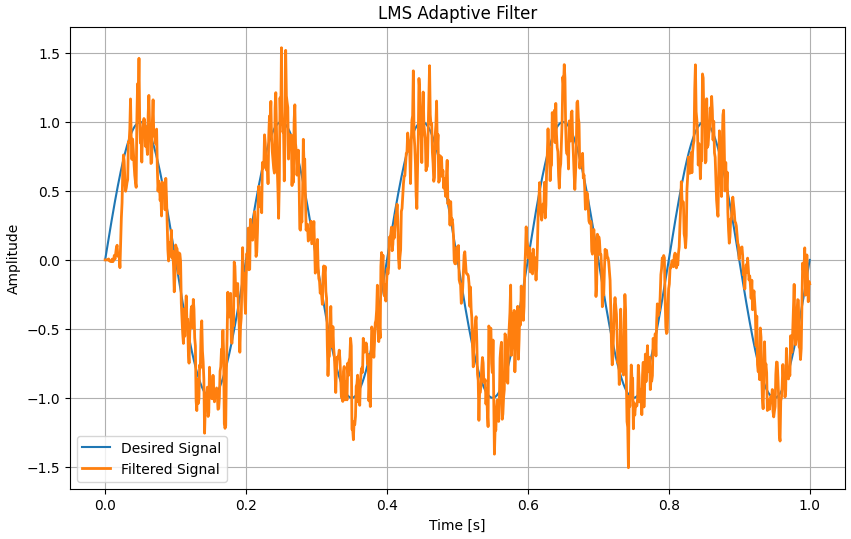
****

**3.3 Adaptive LMS Filter**



We define a function called lms\_filter that adaptively updates filter coefficients to minimize the error between the filtered output and a desired reference signal. The implementation initializes filter weights (w), output signal (y), and error signal (e) arrays, then processes the input sample by sample. For each iteration, it calculates the current output by applying the filter weights to a segment of the input signal, computes the error, and updates the weights proportionally to the error and input using the LMS algorithm.

**Results:**



Link to repository: https://github.com/RafalZmu/School/tree/main/Lab%203

**5. Conclusions:**

Each filter type suits different applications: FIR for precise phase requirements, IIR for efficient filtering with minimal coefficients, and adaptive filtering for unknown or changing signal environments.

The implementation complexity increases from FIR to IIR to adaptive methods, but this corresponds with greater flexibility and potentially improved performance in complex signal processing scenarios.