**Лекція 10**

**Тема 10. Фільтрація вихідного сигналу в ІКС для АУТП.**

*Основні причини виникнення шумів та спотворень у сигналах датчиків. Джерела завад у вихідних сигналах ІКС в АУТП. Види фільтрації сигналу в ІКС: аналогова, цифрова. Основні методи цифрової фільтрації. Приклади використання фільтрів у промислових датчиках. Вибір методу фільтрації залежно від вимог до точності, швидкодії та ресурсів обчислень.*

**Мета**: ознайомити студентів з основними джерелами шумів та завад у сигналах інформаційно-керуючих систем (ІКС) для автоматизованих систем управління технологічними процесами (АУТП); сформувати розуміння необхідності фільтрації вихідного сигналу датчиків; розкрити принципи роботи аналогової та цифрової фільтрації, ознайомити з методами цифрового згладжування та прикладами їх застосування у промислових умовах залежно від вимог до точності, швидкодії та ресурсів системи.

У реальних умовах експлуатації автоматизованих систем управління технологічними процесами (АУТП) вихідні сигнали від датчиків часто містять шуми, перешкоди та спотворення. Ці небажані компоненти погіршують якість управління, точність вимірювань, і можуть спричиняти помилкові реакції виконавчих пристроїв. Мета фільтрації: виділення корисної інформації із сигналу шляхом усунення або зменшення впливу завад без спотворення динаміки процесу.

## **Причини виникнення шумів та спотворень**

| **Джерело** | **Тип завади** |
| --- | --- |
| Електромагнітні впливи | Стрибки напруги, індукції |
| Механічні вібрації | Псевдосигнали від тензодатчиків |
| Температурні зміни | Дрейф нуля, шум в електричних колах |
| Поганий контакт | Імпульсні перешкоди |
| Внутрішній шум АЦП | Квантування, внутрішні флуктуації |
| Швидка зміна параметра | Стрибки, імпульси |

**Джерела завад у вихідних сигналах ІКС**

* Датчики та сенсори: неточності, паразитні сигнали.
* Аналого-цифрове перетворення (АЦП): шум квантування.
* Лінії передачі сигналу: електромагнітні перешкоди.
* Силові пристрої: індукції від частотників, пускових реле.
* Контролери (ПЛК/МК): цифрові коливання, фонові сигнали.

**Види фільтрації в ІКС**

**1. Аналогова фільтрація**

* Застосовується до перетворення сигналу в цифрову форму.
* Використовуються RC-фільтри, активні фільтри на ОП.
* Перевага: працює в режимі реального часу без затримки.
* Недолік: складно масштабувати, нестабільність при старінні компонентів.

**2. Цифрова фільтрація**

* Застосовується після АЦП в мікроконтролері або ПЛК.
* Переваги: гнучкість, точність, адаптивність.
* Недолік: потребує обчислювальних ресурсів.

## **Вибір методу фільтрації залежно від вимог**

| **Критерій** | **Рекомендований метод** |
| --- | --- |
| **Максимальна точність** | Калманівський фільтр / частотний FIR |
| **Низький рівень шуму** | Експоненційне згладжування / середнє |
| **Висока швидкодія** | Експоненційне згладжування (низьке α) |
| **Мінімум ресурсів МК** | Рухоме середнє / коеф. згладжування |
| **Імпульсні перешкоди** | Медіанний фільтр |
| **Висока частота даних** | Цифрові фільтри IIR |

**Фільтр рухомого середнього (Moving Average)**

**Середнє значення останніх N вимірювань. Добре згладжує шум, але знижує швидкодію.**

**Arduino (C-подібна мова):**

#define N 5

float buffer[N];

int index = 0;

float movingAverage(float newValue) {

buffer[index] = newValue;

index = (index + 1) % N;

float sum = 0;

for (int i = 0; i < N; i++) {

sum += buffer[i];

}

return sum / N;

}

**Експоненційне згладжування (Exponential Smoothing)**

**Згладжує сигнал, зменшуючи вплив раптових стрибків.**

def exp\_smoothing(x, y\_prev, alpha=0.2):

return alpha \* x + (1 - alpha) \* y\_prev

**Медіанний фільтр (Median Filter)**

**Вибирає "середнє" значення з вікна, стійкий до імпульсних перешкод.**

**Python (з бібліотекою):**

import statistics

def median\_filter(data\_window):

return statistics.median(data\_window)

**Калманівський фільтр (спрощена реалізація)**

Статистичний фільтр, що враховує похибку вимірювання та попередній стан.

**Python (спрощено):**

def kalman\_filter(z, x\_est, p\_est, q, r):

# z – нове вимірювання

# x\_est – попереднє оцінене значення

# p\_est – попередня дисперсія

# q – шум моделі, r – шум вимірювання

# Етап передбачення

x\_pred = x\_est

p\_pred = p\_est + q

# Оновлення

K = p\_pred / (p\_pred + r)

x\_new = x\_pred + K \* (z - x\_pred)

p\_new = (1 - K) \* p\_pred

return x\_new, p\_new

**Фільтр низьких частот FIR (Python + scipy)**

from scipy.signal import lfilter

# Фільтр із коефіцієнтами (FIR)

def fir\_filter(signal, coeffs):

return lfilter(coeffs, 1.0, signal)

# Приклад: згладжувальний фільтр з 5 коефіцієнтами

coeffs = [0.2, 0.2, 0.2, 0.2, 0.2]

filtered\_signal = fir\_filter(raw\_data, coeffs)