

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Приладобудівний факультет**

**Кафедра інформаційно-вимірjuвальних технологій**

До захисту допущено:

Завідувач кафедри

\_\_\_\_\_ Володимир ЄРЕМЕНКО

«\_\_» \_\_\_\_\_ 20\_\_ р.

**Дипломна робота**

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою «Метрологія та вимірjuвальна  
техніка»**

**спеціальності 152 «Метрологія та інформаційно-вимірjuвальна техніка»  
на тему: «Моделювання алгоритмів цифрової фільтрації»**

Виконав:

студент IV курсу, групи ВВ-71

Матюк Дмитро Іванович \_\_\_\_\_

Керівник:

Доцент, к.т.н., доцент

Синиця Валентин Іванович \_\_\_\_\_

Консультант з нормоконтролю:

Ст. викладач, к.т.н.

Щербань Анастасія Павлівна \_\_\_\_\_

Рецензент:

Ст.викладач, к.т.н.

Лисенко Юлія Юріївна \_\_\_\_\_

Засвідчую, що у цьому дипломному  
проєкті немає запозичень з праць інших  
авторів без відповідних посилань.

Студент \_\_\_\_\_

Київ – 2021

**Національний технічний університет України**  
**«Київський політехнічний інститут імені Ігоря Сікорського»**  
**Приладобудівний факультет**  
**Інформаційно-вимірювальних технологій**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 152 «Метрологія та інформаційно-вимірювальна техніка»

Освітньо-професійна програма «Метрологія та вимірювальна техніка»

ЗАТВЕРДЖУЮ  
Завідувач кафедри  
Володимир ЄРЕМЕНКО  
«\_\_\_» червня 2021р.

**ЗАВДАННЯ**  
**на дипломну роботу студенту**  
**Матюку Дмитру Івановичу**

1. Тема роботи «Моделювання алгоритмів цифрової фільтрації», керівник роботи к.т.н., доц. Синиця Валентин Іванович, затверджені наказом по університету від «\_\_\_»\_\_\_\_\_ 20\_\_ р. №\_\_\_\_\_

2. Термін подання студентом роботи 09.06.2021 р. \_\_\_\_\_

3. Вихідні дані до роботи графічний інтерфейс програми, який призначений для моделювання алгоритмів цифрової фільтрації і повинен забезпечити: а) моделювання вхідного сигналу; б) генерація вимірювальних сигналів: прямий сигнал(функція Хевісайда); синусоїдального сигналу за похибкою та за дисперсією; в) моделювання та відтворення на графіках алгоритмів фільтрації; г) сумісне та візуальне подання сигналів для порівняння.

4. Зміст роботи ТЗ. Вступ. Призначення та галузь застосування. Основні характеристики. Огляд і аналіз існуючих технічних рішень. Розробка структури та програмного компоненту. Проект методичних вказівок до виконання лабораторної роботи: «МОДЕЛЮВАННЯ АЛГОРИТМІВ ЦИФРОВОЇ ФІЛЬТРАЦІЇ». Обробка результатів експериментального дослідження.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) Блок-схема (1л). Панель інтерфейсу користувача (1л).

Експериментальне дослідження алгоритму (1л).

6. Дата видачі завдання 20.02.2021 р. \_\_\_\_\_

Календарний план

№	Назва етапу виконання роботи	Термін виконання
1	Розробка та узгодження технічного завдання	15.03.2021
2	Огляд і аналіз існуючих технічних рішень	21.03.2021
3	Вибір фільтрів для моделювання та порівняння їх ефективності	29.03.2021
4	Розробка практичної частинної в середовищі Microsoft Visual Studio, код програми C++	05.04.2021
5	Дослідження ефективності вибраних фільтрів	26.04.2021
6	Розробка лабораторної роботи та методичних вказівок до її виконання	02.05.2021
7	Оформлення графічних матеріалів	10.05.2021
8	Оформлення пояснювальної записки	21.05.2021
9	Попередній захист дипломного проекту	09.06.2021
10	Рецензування дипломного проекту	14.06.2021
11	Захист дипломного проекту	16.06.2021

Студент

Дмитро МАТЮК

Керівник

Валентин СИНИЦЯ

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО»**

**Приладобудівний факультет Інформаційно-  
вимірювальних технологій**

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри**

**Володимир ЄРЕМЕНКО**

**” \_\_\_\_ ” червня 2021 р.**

**ТЕХНІЧНЕ ЗАВДАННЯ**

**на дипломну роботу**

**«Моделювання алгоритмів цифрової фільтрації»**

**ВВ71.120004.001ТЗ**

**УЗГОДЖЕНО:**

**Керівник дипломної роботи:**

**к.т.н., доц. Синиця В. І.**

**” \_\_\_\_ ” \_\_\_\_\_ 2021 р.**

**Дипломник:**

**студент групи ВВ-71**

**Матюк Д.І.**

**” \_\_\_\_ ” \_\_\_\_\_ 2021 р.**

**Залікова книжка\_\_\_\_\_**

## 1. ВСТУП

1.1 Найменування роботи: моделювання алгоритмів цифрової фільтрації

1.2 Галузь використання: дослідження в приладобудуванні при створенні приладів для автоматизації виробничих процесів; дослідження в вимірювальній техніці; освіта.

## 2. ОСНОВА ДЛЯ РОЗРОБКИ

2.1 Данна робота виконується на основі завдання на дипломну роботу на тему «Моделювання алгоритмів цифрової фільтрації» затвердженого кафедрою інформаційно-вимірювальної техніки від 2 лютого 2021 р.

## 3. ПРИЗНАЧЕННЯ РОБОТИ

3.1 Метою роботи є створення графічного інтерфейсу програми та розгляд алгоритмів цифрової фільтрації для FIR, LMS, Wiener фільтру використовуючи мову програмування C++, інтегроване середовище розробки Visual Studio.

3.2 Розроблена графічна модель допоможе визначити ефективність алгоритмів цифрової фільтрації.

## 4. ВИМОГИ ДО ПРОГРАМНОГО КОМПЛЕКСУ

4.1 Вимоги до функціональних характеристик.

4.1.1 Архітектура програмного комплексу повинна забезпечувати такі функції:

- моделювання вхідного сигналу;
- генерація вимірювальних сигналів: прямий сигнал(функція Хевісайда); синусоїдального сигналу за похибкою та за дисперсією;
- моделювання та відтворення на графіках алгоритми фільтрації
- сумісне візуальне подання сигналів для порівняння.

4.2 Вимоги до показників призначення.

4.2.1 Вимоги до формування вимірювальних сигналів:

- сигнали:

○ функція Хевісайда з амплітудою = 1В

О синусоїдального сигналу зі зміною за похибкою та за дисперсією з амплітудою в діапазоні -1В до +1В;

- шум: похибка АЦП, що дорівнює  $\gamma_{\text{АЦП}} = \frac{1}{4} \text{LSB}$ 
  - о LSB змінюється в діапазоні від -0,5 до +0,5;
  - о Розрядність АЦП: 8

4.2.2 Вимоги до формування візуального відображення сигналів графічної моделі в часовій області та частотній області встановлюються узгоджено з параметрами моделі та параметрами модельного часу.

4.2.3 Вимоги до візуалізації зовнішніх специфікацій.

4.2.3.1 Зовнішні специфікації повинні мати функціональне групування на екрані відео монітора визначене кольором або засобами графіки, рівномірно займати його площину та використовувати вікно, яке не перевищує межі 720 pixel по горизонталі та 600 pixel по вертикалі.

4.3 Вимоги надійності не встановлюються.

4.4 Вимоги до складу та параметрів технічних засобів: процесор Intel I5-7xxx; оперативний запам'ятовуючий пристрій 16 GB; жорсткий диск 1000 GB; SSD 256 GB; монітор 24'' з роздільною здатністю 1920x1080; клавіатура 104 клавіші; маніпулятор миша 1000 dpi; принтер лазерний 1200 dpi (опція); блок резервного живлення 1000 W (опція).

4.5 Вимоги до інформаційної та програмної сумісності.

4.5.1 Вимоги до програмного забезпечення: операційна система Windows 10 32..64 bit.

4.5.2 Вимоги до інструментальних програмних засобів: створення програмного комплексу повинно виконуватись в інтерактивному середовищі візуального програмування.

4.6 Вимоги до маркування та упаковки не встановлюються.

4.7 Вимоги до транспортування та зберігання не встановлюються.

## 5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 В роботі повинні розроблятися програмні документи: опис програми; керівництво користувача.

## 6 СТАДІЇ ТА ЕТАПИ РОБОТИ

6.1 Етапи роботи подані в таблиці 6.1.

Таблиця 6.1

Етапи роботи		
№	Назва етапу виконання роботи	Термін виконання
1	Розробка та узгодження технічного завдання	15.03.2021
2	Огляд і аналіз існуючих технічних рішень	21.03.2021
3	Вибір фільтрів для моделювання та порівняння їх ефективності	29.03.2021
4	Розробка практичної частини в середовищі Microsoft Visual Studio, код програми C++	05.04.2021
5	Дослідження ефективності вибраних фільтрів.	26.04.2021
6	Оформлення графічних матеріалів	02.05.2021
7	Оформлення пояснювальної записки	10.05.2021
8	Попередній захист дипломного проекту	09.06.2021
9	Рецензування дипломного проекту	14.06.2021
10	Захист дипломного проекту	16.06.2021

## 7 ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

7.1 Приймання дипломної роботи виконується комісією затвердженою кафедрою інформаційно-вимірювальної техніки.

7.2 Склад документації, яка подається до приймання:

- технічне завдання – 1 прим.;
- пояснювальна записка з додатками – 1 прим.;
- блок-схема програмного комплексу – 1 лист формату A1;
- панель інтерфейсу користувача – 1 лист формату A1;
- експериментальне дослідження алгоритму – 1 лист формату A1.

7.3 Вимоги технічного завдання можуть бути змінені за бажанням сторін. Всі зміни повинні бути оформлені письмово і затверджені сторонами.

## 8 ВИМОГИ ЩОДО ТЕХНІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ З ОБМЕЖЕНИМ ДОСТУПОМ

8.1 Вимоги щодо технічного захисту інформації з обмеженим доступом не встановлюються.

Студент  
Керівник

Дмитро МАТЮК  
Валентин СИНИЦЯ



## АНОТАЦІЯ

Дипломна робота складається зі вступу, трьох розділів, висновків. Загальний об'єм пояснювальної записки становить 53 сторінки, 39 рисунків та 1 таблиця.

В ході виконання дипломного проекту було розроблено програмне забезпечення для дослідження найефективнішого оптимального фільтру.

Для розробки програмного забезпечення для дослідження найефективнішого з оптимальних фільтрів було використано інтегроване середовище розробки Microsoft Visual Studio 2019, реалізація програмного коду була відтворена через програмний код C++.

Для створення програмно-методичного комплексу були вирішені наступні завдання:

1. Розроблено програмне забезпечення.
2. Розроблено методичне забезпечення створеного програмного забезпечення.
3. Проведено перевірку адекватності та працездатності розробленого програмного інструменту шляхом моделювання обраних алгоритмів фільтрації.

## ABSTRACT

Thesis consists of an introduction, three sections, conclusions. The total length of the explanatory note is 53 pages, 39 figures and 1 table.

During the implementation of the diploma project, software was developed to study the most effective optimal filter.

An integrated Microsoft Visual Studio 2019 development environment was used to develop software to study the most effective of the optimal filters, and the Implementation of the software code was reproduced through the C ++ software code.

The following tasks were solved to create a program-methodical complex:

1. Developed software.
2. Developed methodological support for the created Software.
3. The adequacy and efficiency of the developed software tool was checked by modeling the selected filtering algorithms.

Пояснювальна записка  
до дипломної роботи  
на тему:  
«Моделювання алгоритмів цифрової фільтрації»  
**ВВ71.120004.001ТЗ**

УЗГОДЖЕНО:

Керівник дипломної роботи:

к.т.н., доц. Синиця В. І.

" \_\_\_\_ " \_\_\_\_\_ 2021 р.

Дипломник:

студент групи ВВ-71

Матюк Д.І.

" \_\_\_\_ " \_\_\_\_\_ 2021 р.

Залікова книжка \_\_\_\_\_

Київ – 2021 року

## ЗМІСТ

ВСТУП.....	4
РОЗДІЛ 1 .....	5
ОГЛЯД АНАЛОГІЧНИХ РІШЕНЬ .....	5
1.1 Нестационарний процес.....	5
1.1.1 Класифікація нестационарних процесів .....	5
1.1.2 Класифікація характеристик нестационарних процесів .....	8
1.2 Оптимальна фільтрація сигналів.....	9
1.2.1 Застосування фільтрів.....	11
РОЗДІЛ 2 .....	15
РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ТА МАТЕМАТИЧНОЇ МОДЕЛІ ФІЛЬТРІВ .....	15
2.1 КІХ фільтр .....	15
2.1.1 Математична модель FIR фільтру .....	16
2.1.2 Параметри FIR фільтру .....	18
2.1.3 Проектування КІХ фільтрів .....	19
2.1.3 Розробка програмної моделі FIR фільтра.....	21
2.2 LMS фільтр(реалізований на базі адаптивного алгоритму LMS).....	23
2.2.1 Експлуатація та огляд структури LMS алгоритму .....	23
2.2.2 Алгоритм найменших квадратів у коді (LMS адаптивний алгоритм).....	25
2.3 Фільтр Вінера .....	28
2.3.1 Оптимальний фільтр Вінера .....	29
2.3.2 Фільтр Вінера, реалізація в коді: .....	31
2.4 Побудова графічної моделі.....	32
ВИСНОВКИ ДО РОЗДІЛУ 2 .....	36
РОЗДІЛ 3 .....	37
ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МОДЕЛІ.....	37
3.1 Дослідження за одиничним (окремим ) сигналом.....	37
3.2 Синусоїдальний сигнал зі зміною за дисперсією .....	43

					ВВ71.120004.001 ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Моделювання алгоритмів цифрової фільтрації Пояснювальна записка	Літ.	Аркуш	Аркушів
Розроб.	Матюк Д.І.						1	---
Перевір.	Синиця В.І.							
Реценз.	---							
Н. Контр.	Щербань А.П.							
Затверд.	Ефременко В.С.					НТУУ «КПІ», ПБФ		

3.3 Синусоїдальний сигнал зі зміною за похибкою .....	47
ВИСНОВКИ ДО РОЗДІЛУ 3 .....	53
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	54
ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ.....	55

					BB71.120004.001	Арк.
						3
Зм.	Арк.	№ докум.	Підп.	Дата		

## ВСТУП

Сучасні технології займають високий пріоритет в нашому житті. Для розширення цієї сфери потрібно ставити ще більше високі вимоги в процесі отримання даних. Щоб покращити результат вимірювання ми можемо використовувати різні види фільтрів та їх алгоритм реалізації. Звичайно, що при використанні цих алгоритмів ми все ще маємо похибку, як пристрою вимірювання, так і вхідних/вихідних даних.

Щоб знайти оптимальний алгоритм обробки сигналу потрібно схилитись на деякі шуми та статичні моделі сигналів. В більшість випадків, при формуванні цих моделей використовуються концепції лінійності, стаціонарності та нормалізації(за Гауссом). Але перераховані принципи не завжди виконуються на практиці, що не скажучи про адекватність обраної моделі в певній мірі залежить від якості вхідного сигналу. Кращим рішенням проблеми виявляється використання адаптивних фільтрів, які в свою чергу налаштовують систему так, що вона підлаштовується під статистичні параметри вхідного сигналу, яка не потребує при цьому задання певних моделей.

Аналіз та обробка сигналів лежить в основі вирішення прикладних задач у багатьох науково-технічних областях. Важливою складовою аналізу сигналів є їх класифікація, що базується на поділі сигналів за характером зміни та інформативним змістом.

					BB71.120004.001	Арк.
						4
Зм.	Арк.	№ докум.	Підп.	Дата		

# РОЗДІЛ 1

## ОГЛЯД АНАЛОГІЧНИХ РІШЕНЬ

### 1.1 Нестационарний процес

Нестационарності, як правило, супроводжуються нелінійностями. Це обумовлює необхідність розвитку та використання нетипових підходів до моделювання та прогнозування процесів, які аналізуються.

В залежності від того змінюються чи не змінюються параметри процесів, а саме: швидкість руху потоку, температура, тиск, їх розподіляють на стаціонарні та нестационарні процеси. Якщо ми зобразимо сукупність параметрів які впливають  $\frac{\partial U}{\partial \tau} \neq 0$ , на процес через параметр  $U$ , то при нестационарному процесі, тобто параметри які впливають на процес змінюються не тільки в просторі, але й в часі також.

Нестационарний процес виконує, наприклад, в період запуску та зміни режиму роботи установок неперервної дії. Загалом, в при порівнянні цих двох процесів стаціонарний стан являється менш продуктивним та ефективним, ніж нестационарний.[1]

#### 1.1.1 Класифікація нестационарних процесів

З яскравих прикладів нестационарного процесу — це політ високошвидкісного літака, так як він залежить від висоти та швидкості повітряного потоку, а саме ці параметри можуть змінюватись відносно одного повітряного польоту. Проте, для того щоб побачити результат ми можемо відтворити політ при певно-заданих умовах та забезпечити стаціонарність різних варіацій тисків. Після цього наші параметри з завдання ми можемо простим чином змінювати за певних умовах від експерименту до експерименту отримуючи щоразу стаціонарний запис. В результаті реалізації ми отримаємо детальний опис всіляких умов польоту який створюється в

					BB71.120004.001	Арк.
						5
Зм.	Арк.	№ докум.	Підп.	Дата		

вигляді набору кусково-стаціонарних реалізацій. Такий спосіб відображення в цілому нестаціонарних процесів за допомогою стаціонарних процесів використовується досить часто та наполегливо рекомендується, оскільки вони допомагають запобігти застосування методів, які являються необхідними при аналізі нестаціонарних процесів.

Існують певні ситуації, коли описуючий метод аналізу даних недоцільний, або нереальний, що потрібно аналізувати індивідуальні реалізацію, таким чином, як реалізації нестаціонарного процесу. З обчислювальної точки зору більш успішна ситуація, коли експеримент дає реалізацію аналізуючого нестаціонарного процесу, також допускає повторення при статистично- ідентичних умовах. Таким чином, можна отримати “ансамбль” реалізації в єдиній системі відліку, як показано на рисунку 1.1. Частіше бувають випадки, що нестаціонарне явище, яке ми розглядаємо, являється унікальним і його неможливо повторити при статистично схожих умовах. Наприклад це може бути нестаціонарні хвилі в океані, атмосферна турбулентність, різні часові ряди в економіці. Основні сили, які створюють процес, настільки тяжкі, що неможливо виконати повторний процес при схожих умовах. У всіх таких випадках аналіз можливо здійснити лише за однією реалізацією.

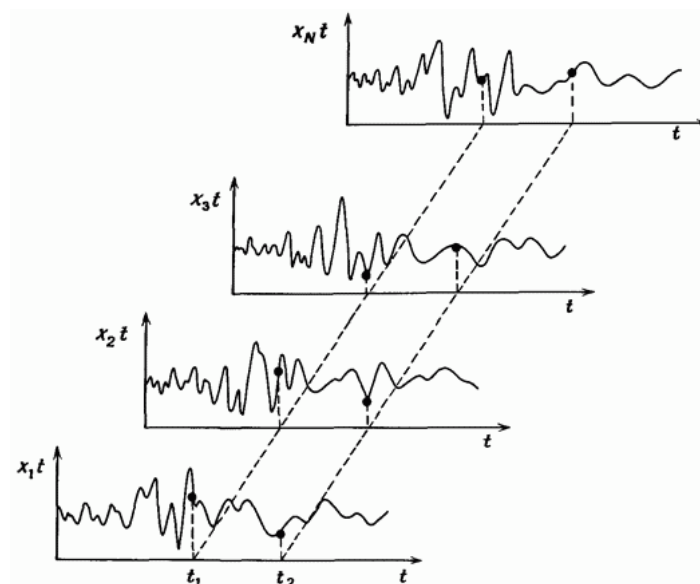


Рисунок 1.1 — Реалізації нестаціонарного випадкового процесу.



Користуючись індивідуальністю реалізації нестационарного процесу, не існує певної методології, у межах якої ми можемо робити аналіз властивостей будь-яких типів цього процесу. В житті це зв'язано з тим, що висновок нестационарності вивчаемого процесу має негативний характер та означає просто відсутність властивості стаціонарності. Коректного визначення природи нестационарності такий висновок не має. Тому для аналізу нестационарних процесів потрібно розробляти спеціальні методи, які можуть застосовуватись тільки до певних класів нестационарних процесів. Зазвичай підхід до реалізації полягає в застосуванні гіпотези про специфічність характеру нестационарності розглядаючого процесу, який розглядається як результат впливу певних детермінованих факторів на стаціонарний випадковий процес. На таблиці 1.1 показано приклади, які представляють собою реалізацію нестационарних процесів виду:

$\varphi(t)$	1	$x(t) = \varphi(t) + \eta x(t)$	Нестационарний за математичним очікуванням
0	$\eta(t)$	$x(t) = \eta(t)x(t)$	Нестационарний за дисперсією
$\varphi(t)$	$\eta(t)$	$x(t) = \varphi(t) + \eta(t) + x(t)$	Нестационарний за математичним очікуванням та дисперсією

Таблиця 1.1 — Часткові випадки нестационарних процесів

де  $u(t)$  — реалізації стаціонарного випадкового процесу  $\{ u(t) \}$ , а  $a(t)$  — задана функція, яка в точності така сама для кожної реалізації. Ці прості нестационарні моделі ми можемо узагальнювати чи комбінувати, домагаючись кращої відповідності реальним фізичним умовам.[2]

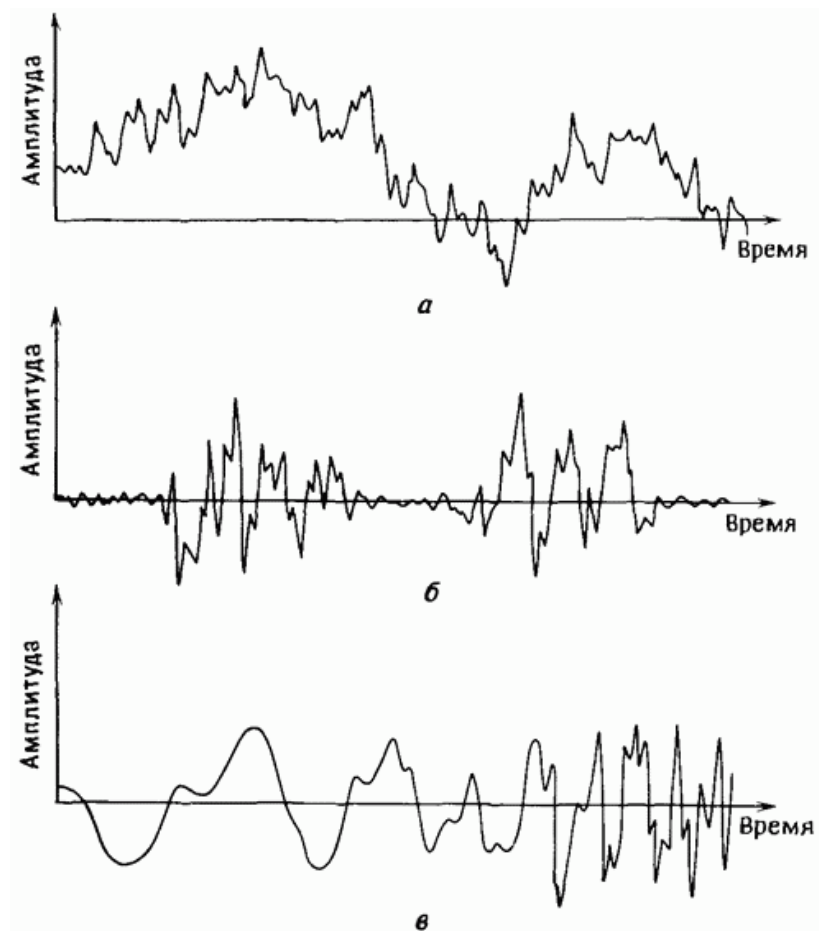


Рисунок 1.2. — Приклади нестационарних процесів:

- а — зі змінним в часі середнім значенням;
- б — зі змінним в часі середнім квадратом;
- в — зі змінним в часі частотною структурою.

### 1.1.2 Класифікація характеристик нестационарних процесів

Більшість випадкових процесів які трапляються на практиці, мають в загальному випадку нестационарний характер. Характеристики нестационарного процесу в загальному випадку являють собою певні функції часу, які можливо визначити тільки “осередненням” по ансамблю реалізації, які утворюють цей процес. В задачах дуже часто отримати велике число реалізації для знаходження характеристик

процесу з необхідною достовірністю являється неможливим. Ця умова перешкоджає розвитку практичних методів оцінювання та аналізу нестационарних випадкових процесів.

В класі нестационарних процесів відповідних справжнім фізичним явищам, в більшість випадків, можемо відокремити особливі типи нестационарності, для яких задача оцінювання й аналізу спрощується. Наприклад, деякі випадкові явища описуються нестационарним випадковим процесом  $\{ Y(t) \}$ , кожна реалізація якого має вид  $Y(t) = A(t) \cdot X(t)$ , де

$X(t)$  – реалізація випадкового стаціонарного процесу  $\{ X(t) \}$

$A(t)$  – детермінований(визначений) множник.

Процеси такого типу мають загальний детермінований тренд. Якщо нестационарний процес відповідає певній моделі такого типу, то для його опису немає необхідності реалізовувати осереднення по ансамблю. Будь-які необхідні характеристики можемо оцінити за однією реалізацією, як і для ергодичних процесів.

## 1.2 Оптимальна фільтрація сигналів

Будь-яку систему необхідно спроектувати так, щоб вона мала хорошу завадостійкість, але завадостійкий зв'язок наразі лишається проблемою.

Часто на практиці точну форму корисного сигналу нам невідомо. Тому реальний сигнал, який приходить на вхід в радіоканал від мікрофону телевізійної камери і тд., можемо приблизно розглядати, як типову реалізацію стаціонарного ергодичного ансамблю. Якщо щільність ймовірності такого випадкового процесу відома(частіше її називають гаусовою), то єдина інформація про всю сукупність можливих сигналів укладена в спектрі потужності або в функції кореляції.[3]

					BB71.120004.001	Арк.
						9
Зм.	Арк.	№ докум.	Підп.	Дата		

В радіоканалі, спектрі потужності, або в функції кореляції крім випадкових корисних сигналів також присутні перешкоди(шуми). Як правило, спектри потужності корисних сигналів і перешкод в тій чи іншій мірі розрізняються, перш за все, своїм розташуванням на частотній осі. Це дозволяє знайти стаціонарний лінійний фільтр, який виділяє випадковий корисний сигнал найкращим чином.

Для теорії сигналів та кіл особливе захоплення являє собою можливість ослаблення шкідливого впливу перешкод за допомогою лінійної фільтрації, заснованих на використанні лінійних частотних фільтрів. Протягом тривалого часу до частотних фільтрів ставилася умова: більш рівномірного пропускання спектру сигналу та більш рівномірного поглинання частот поза зоною цього спектру. Тому ідеальною вважалася П-образна АЧХ у фільтра.

Пізніше стало відомо, що обране вище трактування має наступні недоліки:

1. не враховувалась форма сигналу (може бути різною при однаковій ширині спектра сигналу)
2. не враховувались статистичні властивості завад.

В залежності від задачі – виявлення сигналу, дослідження його параметрів чи розрізнення сигналів – критерії оптимальності можуть бути різними. Для задачі виявлення сигналів в шумах найбільш поширеним являється критерій максимуму відношення сигналу / шум на виході фільтра.

Узгоджений фільтр – це лінійний фільтр, на виході якого максимально-можливе значення сигнал / шум при прийомі сигналу відомої форми на фоні білого шуму.

Оптимальний фільтр – це стаціонарна лінійна частотно-виборча система, яка виконує обробку зашумленого сигналу найкращим способом, тобто  $(с/ш) = \max$ .

					BB71.120004.001	Арк.
						10
Зм.	Арк.	№ докум.	Підп.	Дата		

### 1.2.1 Застосування фільтрів

В загальному випадку фільтри застосовуються:

- для подавлення шумів та перешкод в аналогових сигналах обмеженням їх спектрів знизу та зверху, або усуненням вузьких смуг;
- для обробки сигналів в процесі відтворення та запису звуку для покращення вірності відтворення.
- для ущільнення передачі інформації в каналах в радіозв'язку;
- для звуження спектрів сигналів з допустимою втратою інформації за- для зменшення обсягів файлів при наступних перетворювань в цифрову фо- рму;

Фільтри можуть бути класифіковані за низкою ознак

1. за видом амплітудно-частотної характеристики (АЧХ):

- ФВЧ (фільтр верхніх частот) на рисунку 1.3

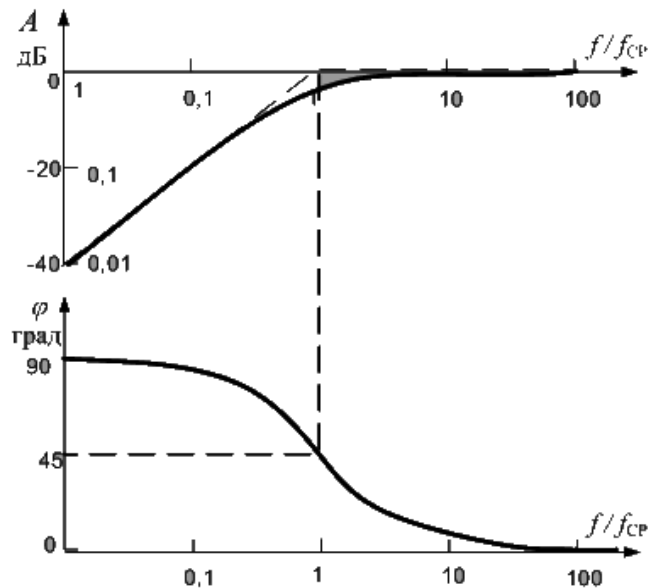


Рисунок 1.3 – ФВЧ(фільтр верхніх частот)

- ФНЧ (фільтр нижніх частот) на рисунку 1.4

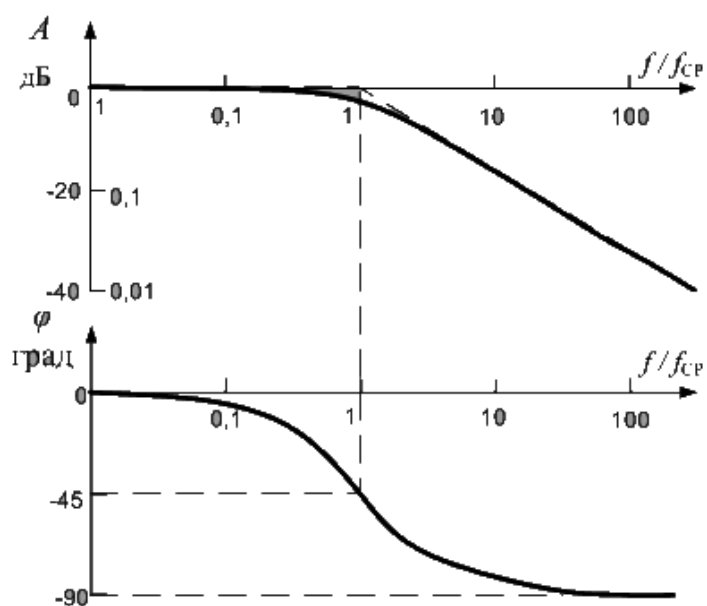


Рисунок 1.4 – ФНЧ (фільтр нижніх частот)

- смугові фільтри (СФ), рисунок 1.5

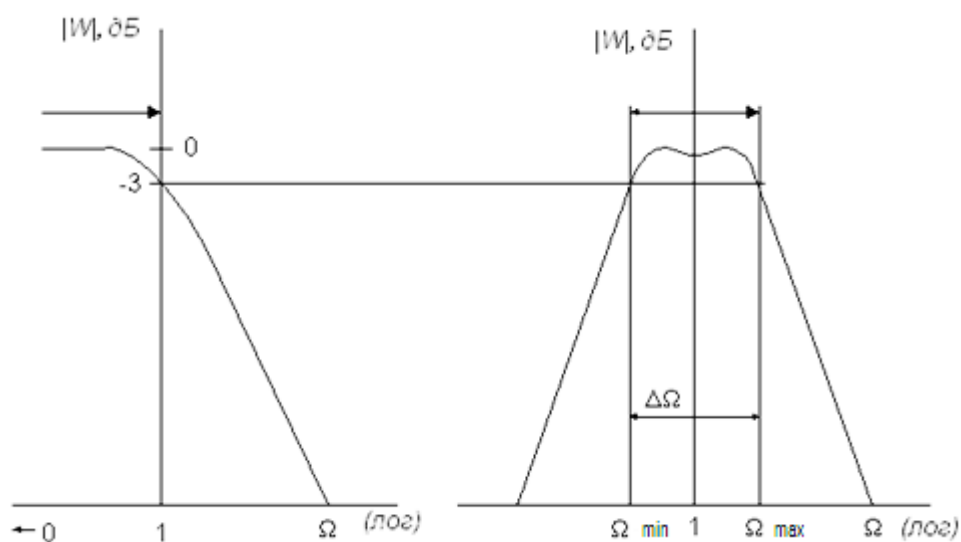


Рисунок 1.5 – Смуговий фільтр (СФ)

- режекторні фільтри (РЖ), рисунок 1.6

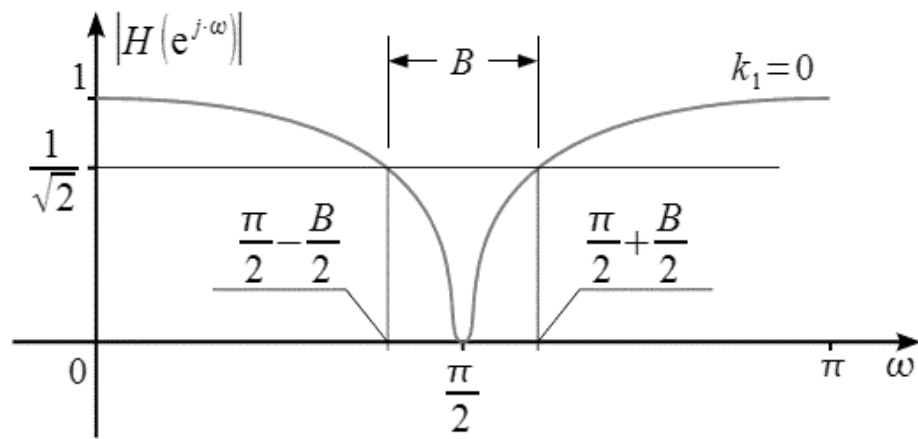


Рисунок 1.6 — Режекторний фільтр (РЖ)

Фазові фільтри (ФФ) можемо відокремити в іншу групу.

- По відношенню до поліномів, які застосовуються при апроксимації перехідної функції розрізняють такі фільтри: Бесселя, Баттерворда, Чебишева, та критичного затухання (АЧХ фільтрів зображена на рисунку 1.7)

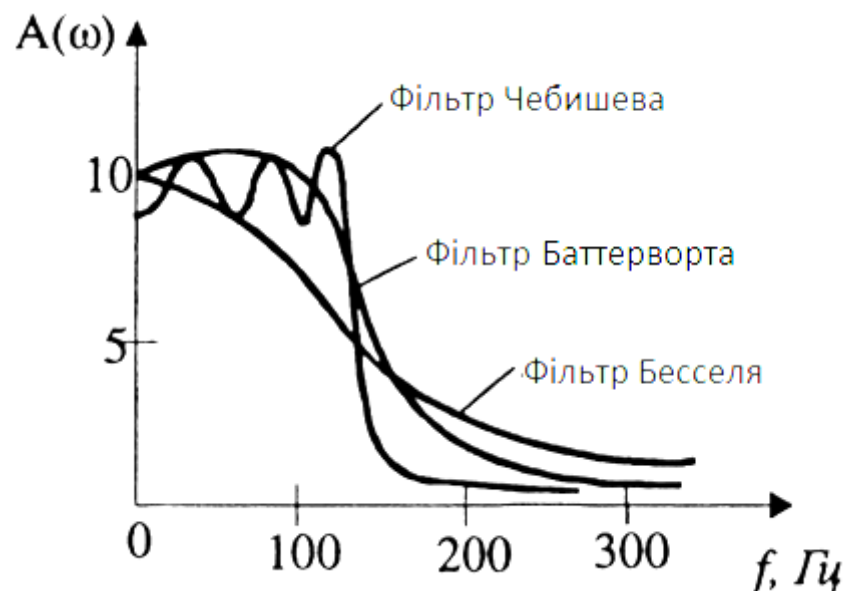


Рисунок 1.7 — Амплітудно частотна характеристика фільтрів Чебишева, Баттерворта, та Бесселя

3. За структурною базою фільтри можна розділити на активні та пасивні. Активні можуть включати в схему RLC- фільтр активного елементу, в якості яких дуже часто використовуються ОП(операційні підсилювачі).

					BB71.120004.001	Арк.
						14
Зм.	Арк.	№ докум.	Підп.	Дата		



## РОЗДІЛ 2

### РОЗРОБКА СТРУКТУРНОЇ СХЕМИ ТА МАТЕМАТИЧНОЇ МОДЕЛІ ФІЛЬТРІВ

В структуру робочої моделі було покладено аналітичне дослідження трьох фільтрів: фільтр Вінера, FIR фільтр (фільтр із скінченною імпульсною характеристикою, та LMS фільтр (фільтр найменших середніх квадратів). Робоча модель реалізована для визначення оптимальності фільтру порівнюючи з іншими. Програмна реалізація обраних видів фільтрів була розроблена на мові високого рівня C++ в середі розробки Microsoft Visual Studio.

#### 2.1 КІХ фільтр

Як відомо, існують такі класи фільтрів як БІХ фільтр — з нескінченною імпульсною характеристикою, та КІХ фільтр — з кінцевою імпульсною характеристикою. КІХ фільтр(з англ. FIR — «finite impulse response») представляє собою цифровий фільтр, важливою особливістю якого являється обмеженість у часі імпульсної характеристики, інакше кажучи вона стає рівною нулю з певного моменту часу. Як правило, більшість FIR фільтрів виконуються без зворотнього зв'язку, тому практично всі КІХ фільтри — нерекурсивні.

Кінцева форма з кількістю ланок  $N$  FIR-фільтру зображена на рисунку 2.1. Як було сказано в попередньому розділі, FIR-фільтр повинен працювати згідно до рівняння, що надає згортку:

$$Y(n) = h(k) * x(n) = \sum_{k=0}^{N-1} h(k)x(n - k), \quad [2.1]$$

де  $h(k)$  – масив з коефіцієнтами фільтру, а  $x(n - k)$  – масив вхідних даних фільтру. Число  $N$  в рівнянні 2.1 це кількість ланок яка визначає ефективність фільтру. КІХ - фільтр з числом ланок  $N$  потребує  $N$  кількість циклів (операцій) множення з збільшенням[6].

					BB71.120004.001	Арк.
						15
Зм.	Арк.	№ докум.	Підп.	Дата		

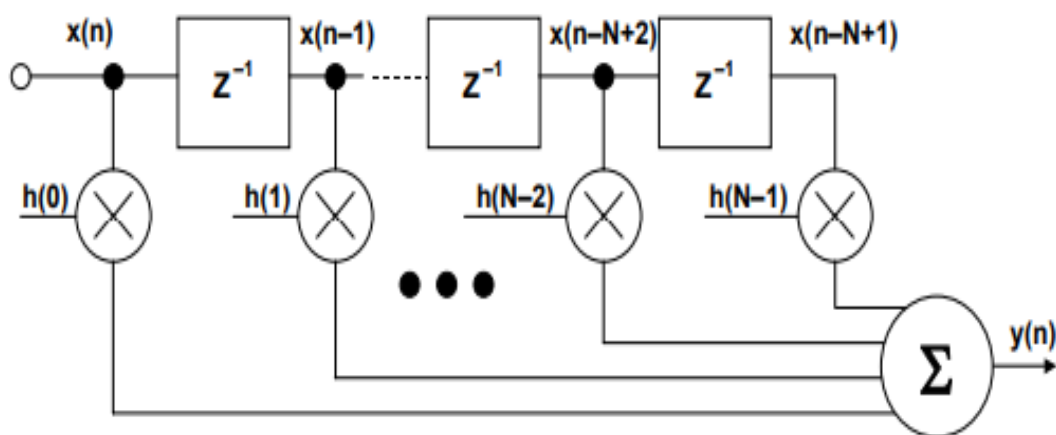


Рисунок 2.1 – FIR фільтр

Моделювання КІХ фільтру полягає в тому, що коефіцієнти фільтру та частотна характеристика визначаються імпульсною характеристикою.

Одиничний імпульс надходить на вхід фільтру і в порядку проходження цього імпульсу через деякі елементи затримки, по черзі, на виході, утворюється коефіцієнти фільтру. Отже, самий процес моделювання фільтру полягає у знаходженні імпульсної характеристики по заданій частотній характеристиці з майбутнім квантуванням імпульсної характеристики в ході створення коефіцієнтів цього фільтру.

### 2.1.1 Математична модель FIR фільтру

FIR фільтр описується наступним рівнянням:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k), \quad [2.2]$$

$$y(n) = \sum_{k=0}^{N-1} h(k)z^{-k}, \quad [2.3]$$

де  $y(n)$  – вихідний сигнал(функція минулих і поточних значень на вході схеми);

$h(k)$  – коефіцієнти імпульсної характеристики сигналу;

$x(n)$  – вхідний вплив сигналу;

$H(z)$  – передавальна характеристика;

$N$  – довжина фільтру (число коефіцієнтів);

Можливість отримання точної лінійної фазової характеристики є важливою особливістю КІХ фільтрів. При проходженні через фільтр сигнал може піддаватися всьляким перетворенням. Зокрема, змінюється амплітуда і фаза сигналу в залежності від частотної характеристики фільтру (фазової, фазо-частотної, амплітудної, амплітудно-частотної характеристики). Для таких сигналів як багаточастотних неприпустимо, щоб фаза сигналу несла спотворення при проходженні через блок обробки. Якщо ж з амплітудно-частотною характеристикою, яка знаходиться в полосі пропускання, зробити практично постійною не займе труднощів, то з фазо-частотною характеристикою можуть виникнути проблеми різного характеру. Для цього вводиться поняття фазової та групової затримок, щоб оцінити спотворення фази.

Отже фазова затримка – значення затримки для кожних з частотних компонент сигналу. Вона знаходиться як кут зсуву фази, розділений на частоту. Групова затримка – середня коротка затримка всього багато-частотного сигналу. Визначається як похідна фази по частоті. Згідно математичної точки зору групова та фазова затримка записуються наступним чином:

$$\tau_{ph} = -\theta(\omega)/\omega \quad [2.4]$$

$$\tau_g = -d\theta(\omega)/d\omega \quad [2.5]$$

З формули 2.4 (для групової затримки) випливає умова лінійності ФЧХ фільтру, а саме, якщо фазово-частотна характеристика – лінійна, отже групова затримка після знаходження похідної дорівнює константі, іншими словами вона являється постійною для частотних компонентів. Логічно зробити висновок, що фільтр з нелінійної фазо-частотною характеристикою буде вносити певні спотворення в фазу сигналу.

					BB71.120004.001	Арк.
						17
Зм.	Арк.	№ докум.	Підп.	Дата		

Отож, лінійність фазової характеристики являється однією з найважливіших особливостей КІХ-фільтрів. [5]

### 2.1.2 Параметри FIR фільтру

Всім відомо, що потрібні унікальні коефіцієнти для різних типів фільтрів тому потрібно визначитися саме з параметрами фільтра, спочатку це зазвичай робиться теоретично, а лише після теоретичного процесу вивчається амплітудно-частотна характеристика дійсних вимірів.

За цією амплітудно-частотною характеристикою ми визначаємося з ідеальною частотною характеристикою (ті частоти які проходять вільно, і як сильно ми їх усуваємо), для цього потрібна імпульсна характеристика ідеального стану, ми можемо вирахувати як Фур'є-образ:

$$h_D(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_D(w) * e^{iwn} dw \quad [2.5]$$

де  $H_D(w)$  – ідеальна характеристика.

Ми отримали ідеальні імпульсні характеристики одразу ж після закінчення обчислень:

$$h_D(n) = 2 \frac{f_c * \sin(nw_c)}{nw_c} \quad n \neq 0 \quad [2.6]$$

$$h_D(n) = 2f_c \quad n = 0,$$

де  $w_c$  та  $f_c$  – частота зрізу сигналу

Коли ми отримали ідеальні імпульсні характеристики потрібно знайти «реальну» імпульсну характеристику. Щоб це реалізувати потрібно визначитися з ваговою функцією  $w(n)$ . Залежно від вимог до фільтру (Хеммінга, Хеннінга, Блекмена,

					BB71.120004.001	Арк.
						18
Зм.	Арк.	№ докум.	Підп.	Дата		

Кайзера), є декілька різновидів функцій в даному випадку я обрав функцію Хеннінга:

$$w(n) = 0.5 * \left( 1 - \cos \left( \frac{2\pi n}{N-1} \right) \right) \quad [2.7]$$

Де коефіцієнт N – кількість коефіцієнтів, тобто довжина фільтра.

Потрібно перемножити вагову функцію з ідеальною імпульсною характеристикою:

$$h(n) = h_D(n) * w(n) \quad [2.8]$$

### 2.1.3 Проектування КІХ фільтрів

Під «розрахунком FIR фільтру» здебільшого здійснюють пошук за значенням його частотної характеристики коефіцієнтів.

При створенні нового цифрового КІХ фільтру потрібно пройти через певні стадії розробки:

- **Специфікація фільтру.** Задається тип фільтру (фільтр верхніх, нижніх частот(ФВЧ, ФНЧ), режекторний, смуговий), число коефіцієнтів N, потрібна частотна характеристика з допусками на нелінійність в полосі пропускання та згорання.
- **Аналіз наслідків розрядності.** На даному кроці відбувається оцінка впливу ефектів квантування на вихідні, проміжні та на коефіцієнти фільтра.
- **Реалізація.** Здійснюється реалізація фільтра через створення ІР-ядер, або програмна розробка шляхом використання мови програмування.
  - дані кроки в на шляху до реалізації можуть відрізнятися, але суть не змінюється.

					BB71.120004.001	Арк.
						19
Зм.	Арк.	№ докум.	Підп.	Дата		

Блок схема алгоритму FIR фільтру приведена на рисунку 2.2.



Рисунок 2.2 —Блок схема алгоритму фільтра з кінцевою імпульсною характеристикою.

### 2.1.3 Розробка програмної моделі FIR фільтра

Для початку потрібно визначитися з основними параметрами, а саме з довжиною фільтра, з частотою дискретизації вхідних даних, з частотою смуги пропускання та затухання:

```
const int N = 4; //довжина фільтра

double Fd = 200; //Частота дискретизації вхідних даних

double Fs = 1; //Частота смуги пропускання

double Fx = 2; //Частота смуги затухання
```

Також додатково виділяємо пам'ять для параметрів знаходження фільтра:

```
long double H[N] = { 0 };

//імпульсна характеристика фільтра

long double H_id[N] = { 0 };

//ідеальна імпульсна характеристика

long double W[N] = { 0 };

//вагома функція
```

Розрахунок імпульсної характеристики:

```
double Fc = (Fs + Fx) / (2 * Fd);

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

    if (I == 0) H_id[i] = 2 * M_PI*Fc;

    else H_id[i] = sinl(2 * M_PI*Fc*i) / (M_PI*i);

    W[i] = 0.5 - 0.5 * cos((2 * M_PI * i)/(N - 1));
```

					BB71.120004.001	Арк.
						21
Зм.	Арк.	№ докум.	Підп.	Дата		

$$H[i] = H\_id[i] * W[i];$$

де  $W[i]$  – вагома функція (вікно Ханна).

Одним з останніх кроків є нормування імпульсної характеристики. Після нормування ми маємо переконатися, що сума її відліків повинна дорівнювати одиниці:

```
double SUM = 0;

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

    SUM += H[i];

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

    H[i] /= SUM;           // сума коефіцієнтів дорівнює 1
```

Щоб отримати правильне відтворення профільтованих даних потрібно якраз і проводити нормування імпульсної характеристики.

Завершальна частина – це програмний алгоритм фільтрації вхідних даних:

```
for (int I = 0; I < sizeIn + 1; i++) {

    out[i] = 0.;

    for (int j = 0; j < N - 1; j++)

        // моделювання формули КІХ-фільтру

        if (I - j > 0)

            out[i] += H[j] * in[I - j];
```



## 2.2 LMS фільтр(реалізований на базі адаптивного алгоритму LMS)

Цей LMS алгоритм став найбільш популярним серед адаптивних алгоритмів через свою простоту. Знаходження не потребує обчислення матричних інверсій або кореляційної матриці. Саме так, цей алгоритм дійсно простий в роботі, що і створило його стандартом, та займає високе місце в порівнянні з іншими алгоритмами фільтрації [7].

LMS алгоритм проявив свою вартість в адаптивній обробці сигналів як один з ключових функціональних блоків. Він запропонував декілька дуже бажаних характеристик:

- Простий у впровадженні, будь то програмне або апаратне забезпечення.
- Незалежність роботи(будь то у відомому чи невідомому середовищі).
- Можливість стежити за тимчасовими варіаціями вихідної статистики.

### 2.2.1 Експлуатація та огляд структури LMS алгоритму

LMS алгоритм (алгоритм найменших середніх квадратів) – адаптивний лінійний алгоритм фільтрації, який включає в себе два основні процеси, а саме:

- фільтрування, яке містить:
  - А) генерацію похибки оцінювання через порівняння отриманого виходу з жаданим результатом;
  - Б) розрахунок виходу фільтру, створеного через суму вхідних сигналів;
- Адаптивний процес, який включає в себе автоматичне регулювання важелів крана фільтра згідно з оцінкою помилки.

					BB71.120004.001	Арк.
						23
Зм.	Арк.	№ докум.	Підп.	Дата		

Структура LMS алгоритму може використовуватись для будь-яких додатків в фільтрації, як:

- 1 Скасування шуму
- 2 Ідентифікація системи
- 3 Адаптивного передбачення
- 4 Вирівнювання
- 5 Зворотне моделювання
- 6 Скасування втручання

#### Ідентифікація системи

На рисунку 2.2 ми можемо розглянути адаптивну структуру LMS фільтру, яка використовується для моделювання або ідентифікації системи. Вхід  $u(n)$  знаходиться в невідомій системі паралельно до адаптивного фільтру. Сигнал похибки  $e(n)$  – це результат різниці реакції адаптивного фільтру  $y(n)$  між результатом системи  $d(n)$ . Сигнал помилки повертається в адаптивний фільтр який здійснює покращення коефіцієнтів адаптивного фільтру, до тих пір поки загальний результат не буде рівним  $d(n) = y(n)$ . Процес адаптації починає закінчуватися, коли це відбувається, а  $e(n)$  приближається до 0[8].

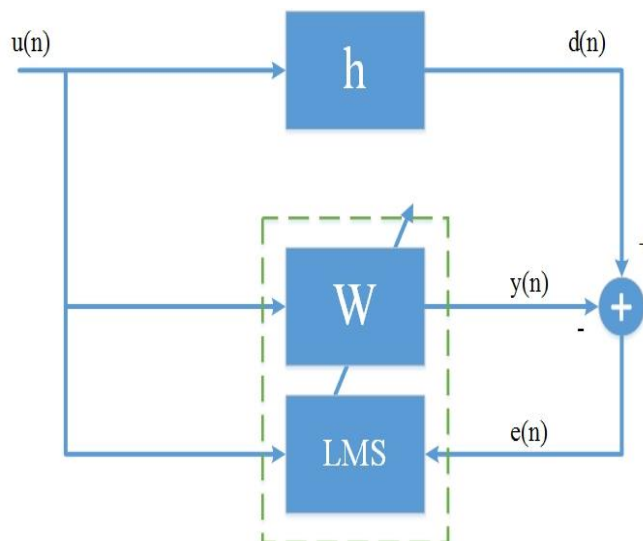


Рисунок 2.3 — Адаптивна структура фільтру

Як зображено на рисунку 2.3,  $y(n)$  – вихід адаптивного фільтра:

$$y(n) = \sum_{k=0}^{M-1} w_k(n)u(n-k) \quad [2.9]$$

де  $w_k(n)$  показує  $M$  кількість коефіцієнтів за певний період часу  $n$ .

Щоб дізнатися наскільки фільтр є оптимальним, слід провести дослідження його продуктивності. Обраний метод базується на похибці сигналу, яка дорівнює різниці між сигналом на виході адаптивного фільтра  $y(n)$  та бажаним сигналом  $d(n)$ . Саме коефіцієнти  $w_k(n)$  редагуються так, щоб функція середньої квадратичної похибки була максимально зменшена.

$$e(n) = d(n) - y(n) \quad [2.10]$$

Середня квадратична функція похибки  $e(n)$   $E = e(n)^2$ , де  $E$  показує очікуваний результат. Так як присутні  $k$  коефіцієнти, то ми можемо знайти градієнт середньої квадратичної функції похибки. Використаємо градієнт  $e(n)$ :

$$w_k(n+1) = w_k(n) + \mu e(n)x(n-k) \quad k = 0, 1, \dots, N-1 \quad [2.11]$$

### 2.2.2 Алгоритм найменших квадратів у кодї (LMS адаптивний алгоритм)

Для реалізації алгоритму найменших квадратів використаємо адаптивний алгоритм

Для початку потрібно встановити порядок фільтра та коефіцієнт збіжності:

```
#define mu 0.09

// встановлення коефіцієнту збіжності

#define M 2

//встановлення порядку фільтра
```

Встановимо, оцінку системи:

					BB71.120004.001	Арк.
						25
Зм.	Арк.	№ докум.	Підп.	Дата		

```
double H[M] = {0.75, 0.25, 0, 0.125, 0.0625}
```

Потрібно встановити інверсію системи згортки (обираються лише два останні елементи, тому що порядок фільтру дорівнює 2):

```
double H[M] = { /* 0.0625, 0.125, 0, */ 0.25, 0.75 };
```

Завдяки згортанню сформованих випадкових даних і Н створюємо потрібний нам сигнал:

```
double D, Y, E;
```

```
double W[M] = { 0.0 };
```

```
double X[M] = { 0.0 };
```

```
double* Desired = new double[size];
```

```
for (int i = 0; i < size; i++)
```

```
    Desired[i] = { 0.0 };
```

```
for (int I = 0; I < size; i++) {
```

```
    for (int j = 0; j < M; j++) {
```

```
        if (I - j >= 0)
```

```
            Desired[i] += input[I - j] * H[j];
```

```
    }
```

```
}
```

```
// обчислення виходу адаптивного фільтру y(n)
```

```
for (int T = 0; T < size; T++) {
```

					BB71.120004.001	Арк.
						26
Зм.	Арк.	№ докум.	Підп.	Дата		

```

for (int m = T; m > T - M; m--) {

    if (m >= 0)

        X[M + (m - T) - 1] = input[m];

    else break;

}

D = Desired[T];

//очікуваний сигнал

Y = 0;

//встановлення на визід фільтру 0

for (int I = 0; I < M; i++)

    Y += (W[i] * X[i]);

```

Обчислення похибки сигналу:

```
E = D - Y;
```

Оновлення всіх коефіцієнтів фільтру найменших квадратів:

```

for (int I = 0; I < M; i++)

    W[i] = W[i] + (mu * E * X[i]);

output[T] = Y;

```

Для кожної вихідної точки вибірки процес буде повторюватись.

## 2.3 Фільтр Вінера

При обробці сигналів фільтр Вінера - це фільтр, що використовується для отримання оцінки бажаного або цільового випадкового процесу за допомогою лінійної інваріантної до часу фільтрації (LTI) фільтрації спостережуваного шумного процесу, припускаючи відомі стаціонарні спектри сигналу та шуму та додатковий шум. Фільтр Вінера мінімізує середньоквадратичну помилку між передбачуваним випадковим процесом та бажаним процесом.

Метою фільтра Вінера є обчислення статистичної оцінки невідомого сигналу з використанням відповідного сигналу як вхідного сигналу та фільтрування цього відомого сигналу для отримання оцінки як вихідного. Наприклад, відомий сигнал може складатися з невідомого сигналу, який був пошкоджений адитивним додатковим шумом. Фільтр Вінера може бути використаний для фільтрації шуму та для отримання оцінки базового сигналу, який нас цікавить.

Типові детерміновані фільтри розроблені для отримання бажаної частотної характеристики. Однак алгоритм фільтра Вінера використовує інший підхід. Передбачається, що ми маємо початкові дані про спектральні властивості вихідного сигналу та шуму, а інший підхід шукає лінійний інваріантний до часу фільтр, вихід якого буде максимально наближений до вихідного сигналу. Фільтр Вінера характеризується наступним:

1. Припущення: сигнал та (адитивний) шум - це стаціонарні лінійні випадкові процеси з відомими спектральними характеристиками або з відомою автокореляцією та взаємокореляцією
2. Вимога: фільтр повинен бути фізично реалізованим
3. Критерій ефективності: мінімальна середньоквадратична помилка (MMSE)

Цей фільтр часто використовується в процесі деконволюції.

					BB71.120004.001	Арк.
						28
Зм.	Арк.	№ докум.	Підп.	Дата		

### 2.3.1 Оптимальний фільтр Вінера

Перш ніж власне розглядати алгоритми адаптації, необхідно встановити оптимальні параметри фільтра, до яких ці алгоритми повинні наближуватись. Підхід до задачі оптимальної фільтрації може бути як статистичним, так і детермінованим. Спочатку розглянемо статистичний варіант.

Якщо випадковий дискретний сигнал  $x(k)$  оброблюється нерекурсивним дискретним фільтром  $N$ -го порядку, коефіцієнти якого можуть бути представлені векторним стовбцем  $\mathbf{w} = [w_0, w_1, \dots, w_N]^T$ . Вихідний сигнал фільтра дорівнює:

$$y(k) = \mathbf{u}^T(k)\mathbf{w} \quad [2.12]$$

$$\mathbf{u}(k) = [x(k), x(k-1), \dots, x(k-N)]^T \text{ — де} \quad [2.13]$$

вектор стовбець вмісті лінії затримки фільтра на  $k$ -му кроці.

Передусім, є сигнал  $d(k)$  який являється також випадковим. Помилка результату такого сигналу дорівнює:

$$e(k) = d(k) - y(k) = d(k) - \mathbf{u}^T(k)\mathbf{w} \quad [2.14]$$

Потрібно визначити коефіцієнти фільтру  $\mathbf{w}$ , які будуть забезпечувати максимальну ідентичність вихідного сигналу до зразкового фільтру, тобто які максимально зменшать похибку  $e(k)$ . Оскільки  $e(k)$  також є випадковим процесом, в якості запобіжної величини розумно прийняти середній квадрат. Таким чином, оптимізований функціонал виглядає так:

$$J(\mathbf{w}) = \overline{e^2(k)} \rightarrow \min \quad [2.15]$$

Квадрат похибки дорівнює:

$$e^2(k) = (d(k) - \mathbf{u}^T(k)\mathbf{w})^2 = d^2(k) - 2d(k)\mathbf{u}^T(k)\mathbf{w} + \mathbf{w}^T\mathbf{u}(k)\mathbf{u}^T(k)\mathbf{w}. \quad [2.16]$$

Статистично усереднити даний вираз, отримаємо:

$$J(\mathbf{w}) = \overline{e^2(k)} = \overline{d^2(k)} - \overline{2d(k)\mathbf{u}^T(k)\mathbf{w}} + \mathbf{w}^T \overline{\mathbf{u}(k)\mathbf{u}^T(k)\mathbf{w}}. \quad [2.17]$$

Вхідні величини в отриману формулу несуть такий сенс:

- $\overline{d^2(k)} = \sigma_d^2$  – середній квадрат зразкового сигналу;
- $\overline{d(k)\mathbf{u}^T(k)} = \mathbf{p}^T$  – транспонований векторстовбець  $\mathbf{p}$  взаємо-кореляцій між  $k$ -м відліком зразкового сигналу та вмістом лінії затримки фільтру. Якщо розглянуті випадкові процеси  $x(t)$  та  $d(k)$  є спільно стаціонарними, то вектор взаємних кореляцій не залежить від номеру кроку  $k$ ;
- $\overline{\mathbf{u}(k)\mathbf{u}^T(k)} = \mathbf{R}$  – кореляційна матриця з розміром  $(N+1) \times (N+1)$ .

$$\mathbf{R} = \begin{bmatrix} R_x(0) & R_x(1) & R_x(2) & \dots & R_x(N) \\ R_x(1) & R_x(0) & R_x(1) & \dots & R_x(N-1) \\ R_x(2) & R_x(1) & R_x(0) & \dots & R_x(N-2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ R_x(N) & R_x(N-1) & R_x(N-2) & \dots & R_x(0) \end{bmatrix},$$

тут  $R_x(m) = \overline{x(k)x(k-m)}$  — кореляційна функція (КФ) випадкового процесу  $\{x(k)\}$ .

З урахуванням введених позначень рівняння приймає наступний вид:

$$J(\mathbf{w}) = \sigma_d^2 - 2\mathbf{p}^T\mathbf{w} + \mathbf{w}^T\mathbf{R}\mathbf{w}. \quad [2.18]$$

Дане рівняння представляє форму щодо  $\mathbf{w}$ , тому матриця  $\mathbf{R}$  має єдиний мінімум. Щоб знайти розв'язок цього рівняння потрібно прирівняти цей єдиний мінімум до нуля вектор градієнту:

$$\mathbf{grad} J(\mathbf{w}) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w} = 0. \quad [2.19]$$



Звідси отримаємо рішення для оптимальних коефіцієнтів фільтру:

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{p}. \quad [2.20]$$

Такий фільтр називається фільтром Вінера. Підстановка рівняння дає мінімально досяжну дисперсію сигналу похибки.[9]



Рисунок 2.4 — Структурна схема роботи фільтра Вінера

### 2.3.2 Фільтр Вінера, реалізація в коді:

```
WienerFilter(
    int parM, //розмірність отримуючого фільтру
    int parN, //розмірність отримуючого фільтру
    double parWienerKcoef, //коефіцієнт в формулі Вінера
    (відношення сигнал/шум)
    double parSigma) //- значення сігми в формулі Гауса
{
    GaussFilter gaussFilter(parN,parM,parSigma);

    vector <vector<complex<double> > > gaussFilterFourie;

    dft(gaussFilter.filter,                                gaussFilterFourie,
    DFT_COMPLEX_OUTPUT);
```

```
filter.resize(parM);
```

Приведено алгоритм фільтрації даних:

```
for (int i = 0; i < parM; i++)
{
    filter[i].resize(parN);
    for (int j = 0; j < parN; j++)
    {
        complex<double>          gaussFilterFourieAbs          =
gaussFilterFourie[i][j]
                                * conj(gaussFilterFourie[i][j]);
        filter[i][j] = gaussFilterFourieAbs /
                                (gaussFilterFourie[i][j]
                                *
(gaussFilterFourieAbs + parWienerKoef));
    }
}
}
```

## 2.4 Побудова графічної моделі

Лістинг реалізація даної програмної моделі на мові приведена у Додатку А.

Робочий інтерфейс реалізації фільтрації зображено на рисунку 2.5:

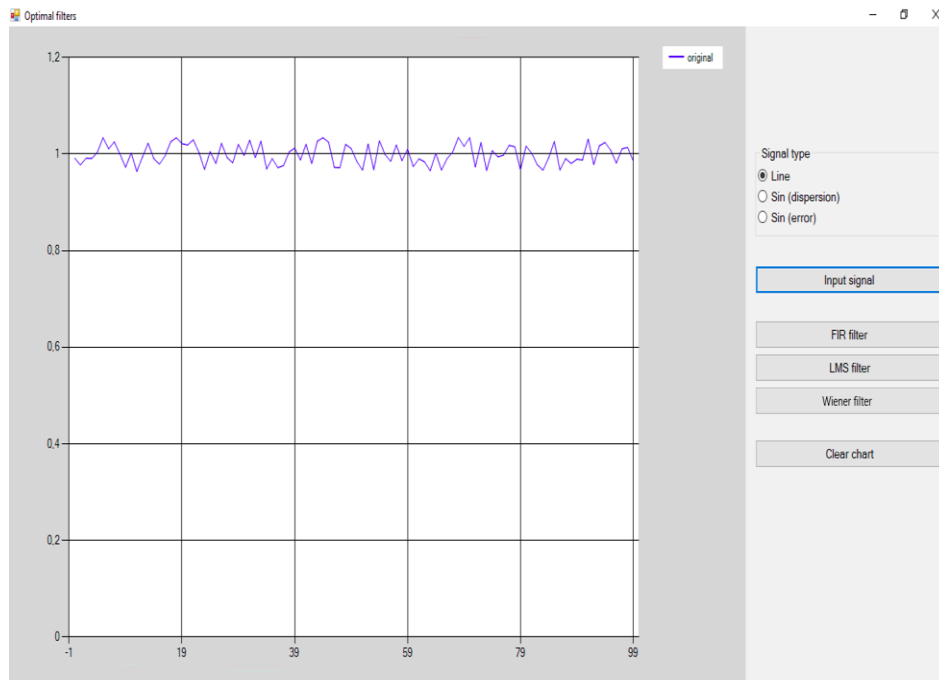


Рисунок 2.5 — Вікно графічної моделі оптимальної фільтрації (за замовчування обрано одиничний сигнал з похибкою Line)

Модель складається з:

- графічного вікна, де можна побачити сигнал (рисунок 2.6)

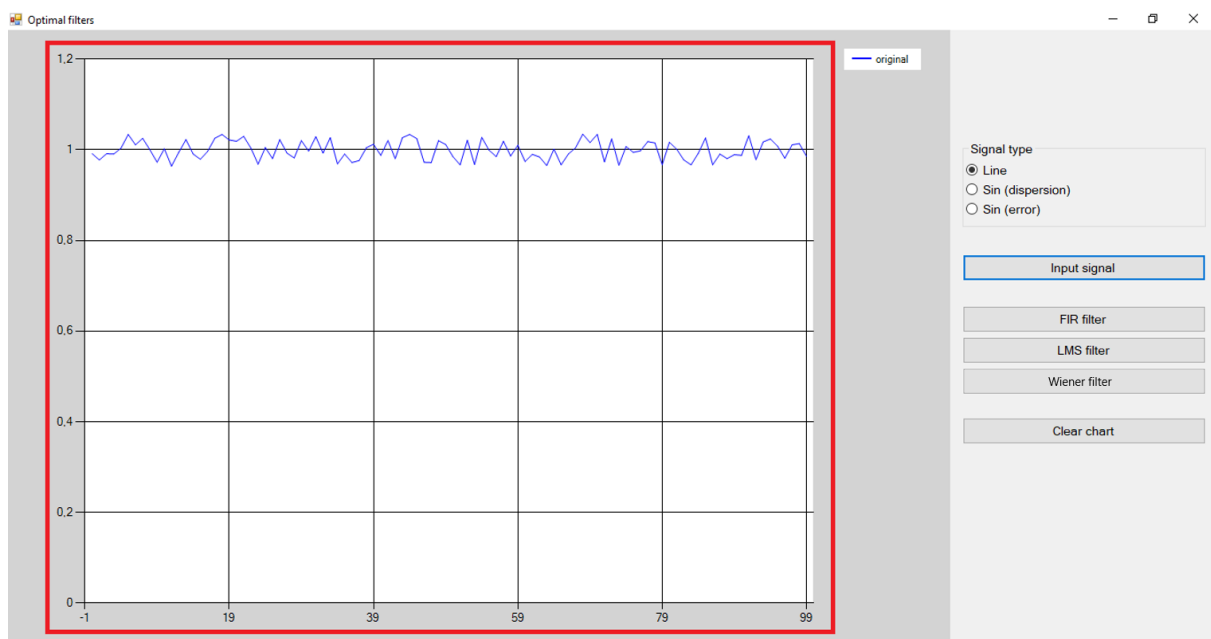
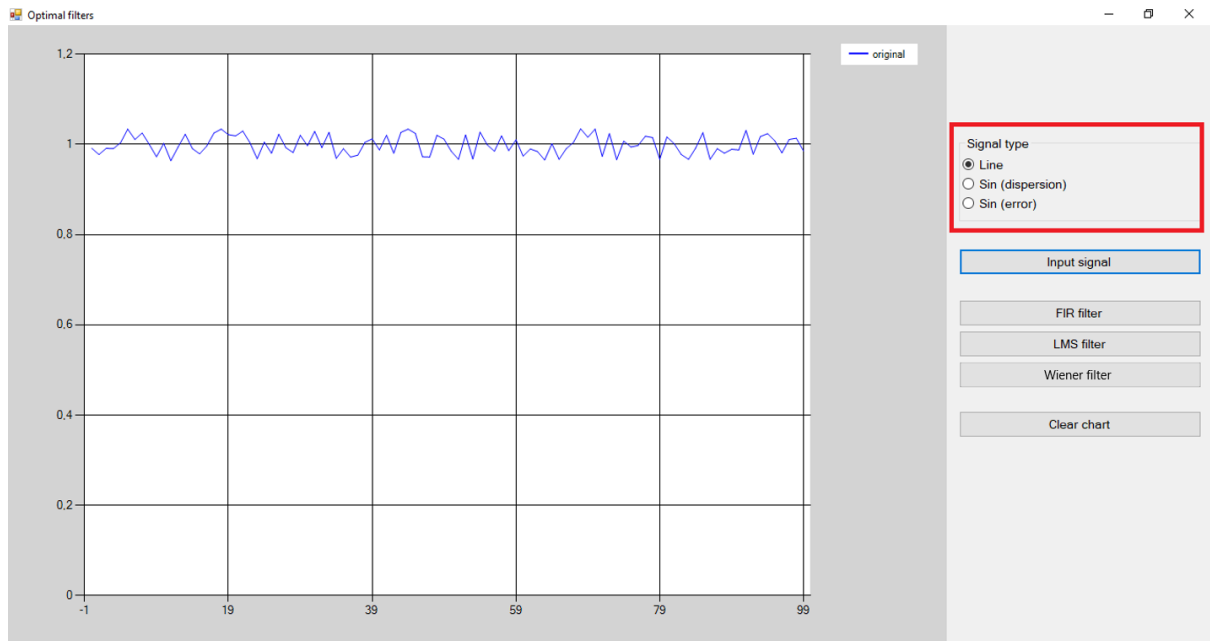


Рисунок 2.6 – Графічне зображення моделі, блок де можемо побачити сигнал (за замовчування обрано одиничний сигнал з похибкою Line)

- меню з кнопками вибору типів сигналу яке зображене на рисунку 2.6: одиничний (окремий) сигнал з похибкою; синусоїдальний(окремий) сигнал, зі зміною дисперсії; синусоїдальний окремий сигнал, з похибкою:



Ри-

сунок 2.6 – Кнопки вибору сигналу у графічній моделі

- меню інтерфейсу кнопок з генерацією типу функцій (рисунок 2.7): генерація вхідного сигналу (кнопка Input signal); FIR filter(фільтр з скінченною імпульсною характеристикою); LMS фільтр (фільтр найменших середніх квадратів; Wiener filter (фільтр Вінера):

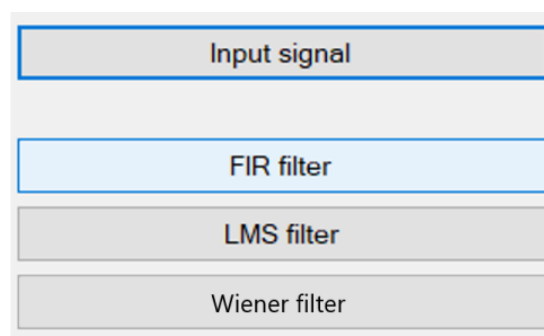


Рисунок 2.7 – Кнопки виклику функцій у графічній моделі

- кнопка очищення графіку на рисунку 2.9

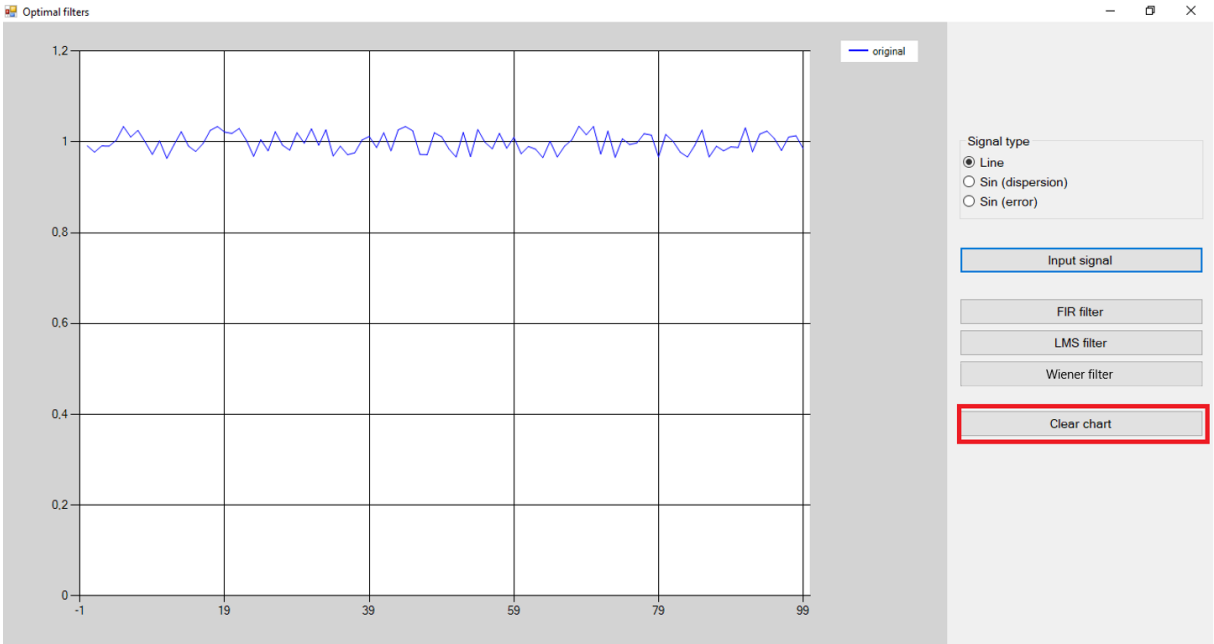


Рисунок 2.8 – Кнопки виклику функцій у графічній моделі

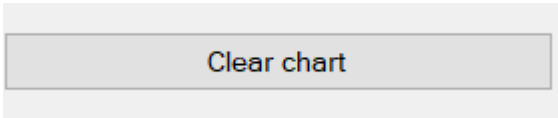


Рисунок 2.9 – Кнопка очищення обраної функції у графічній моделі  
(детальна версія)

## ВИСНОВКИ ДО РОЗДІЛУ 2

Основної задачею цього розділу було змодельовати досліджувану модель для порівняння фільтрів та їх алгоритмів і визначення найоптимальнішого та більш ефективного з них.

Були розглянуто три програмні алгоритми фільтрації, та було розроблено графічний інтерфейс для моделювання цифрових фільтрів, яка може відтворювати на графіках розглянуті алгоритми. Для реалізації була обрана мова високого рівня C++. Код для генерації прямого(окремого) сигналу(функція Хевісайда), синусоїдального сигналу за дисперсією та за похибкою.

Характеристики програмної реалізації моделі:

- сигнали:
  - прямий сигнал(функція Хевісайда) з амплітудою = 1В
  - синусоїдальний сигналу зі зміною за дисперсією та за похибкою з амплітудою в радіусі від -1В до +1В;
- шум: похибка АЦП, що дорівнює  $\gamma_{\text{АЦП}} = \frac{1}{4} \text{LSB}$ 
  - max LSB регулюється в радіусі від -0,5 до +0,5;
  - Розрядність АЦП: 8.
  - СКО:  $\sigma = \frac{\Delta}{2\sqrt{3}}$

Ступінь квантування:  $\Delta = \frac{2}{2^{\sigma-1}}$

					BB71.120004.001	Арк.
						36
Зм.	Арк.	№ докум.	Підп.	Дата		

## РОЗДІЛ 3

### ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МОДЕЛІ

#### 3.1 Дослідження за одиничним (окремим) сигналом

Для того щоб провести дослідження за одиничним сигналом потрібно згенерувати окремий сигнал з похибкою АЦП, яка дорівнює  $\gamma_{\text{АЦП}} = \frac{1}{4} \text{LSB}$ , де LSB – молодший значущий розряд, в радіусі від -0,5 до +0,5:

Код моделі генерації окремого сигналу наведений в Додатку Б.

Для того, щоб згенерувати сигнал у графічній програмі слід вибрати (Рисунок 3.1):

- ✓ Тип сигналу – Line;
- ✓ Натиснути кнопку для зображення сигналу Input Signal;

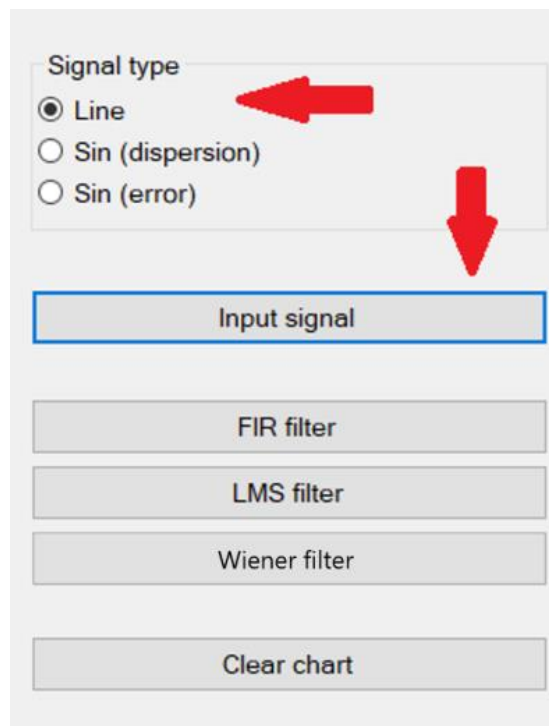


Рисунок 3.1 – Меню інтерфейсу програми

У вікні графіку можемо розглянути згенерований сигнал(рисунок 3.2):

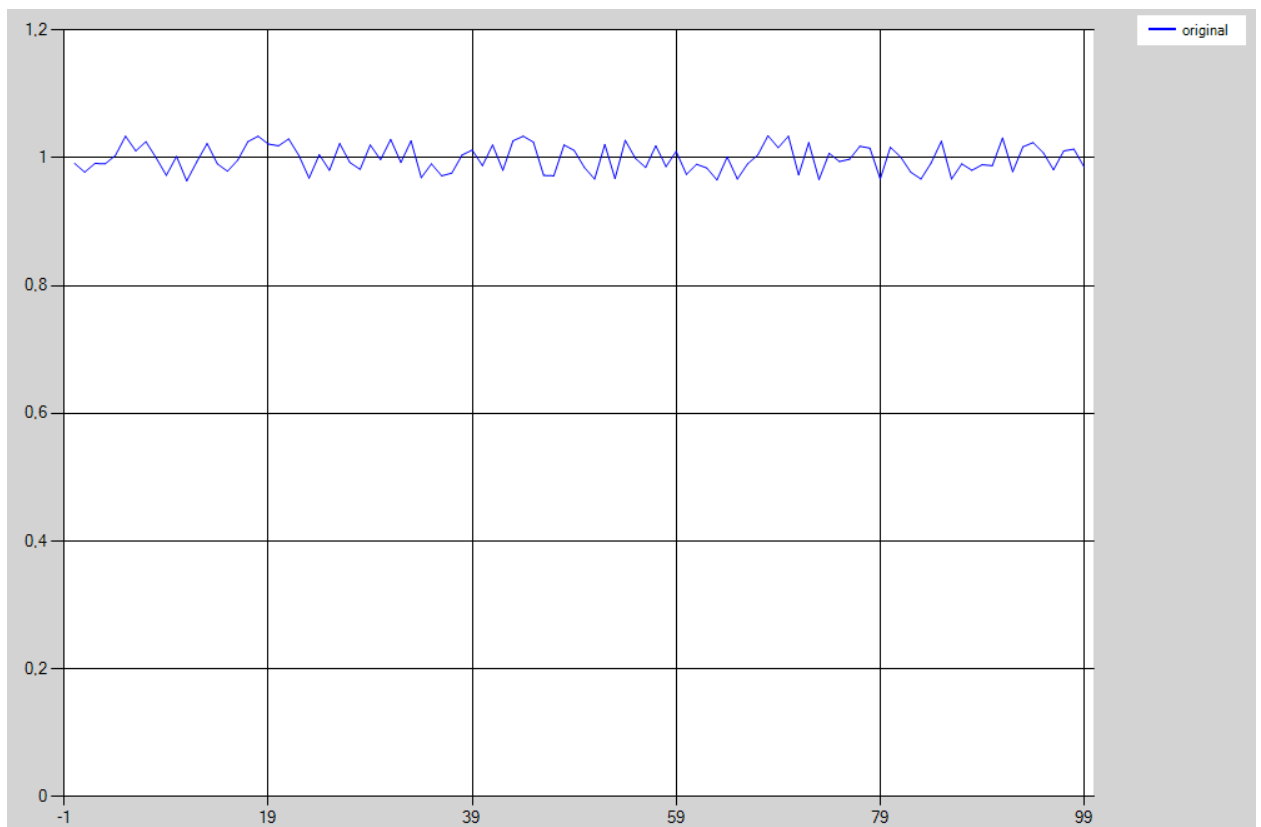


Рисунок 3.2 Згенерований окремий сигнал

Модуляція фільтру зі скінченною імпульсною характеристикою. В панелі меню кнопок обираємо FIR filter (рисунок 3.3):

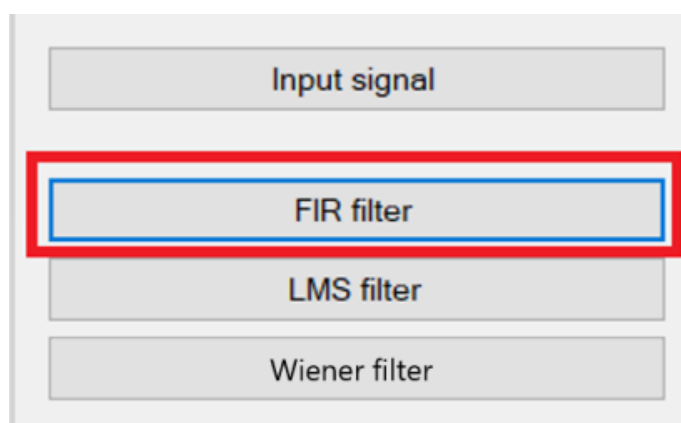


Рисунок 3.3 – Вибір алгоритму фільтрації в меню інтерфейсу програми

У вікні робочої моделі можна побачити відфільтрований сигнал за допомогою FIR фільтру (рисунок 3.4):



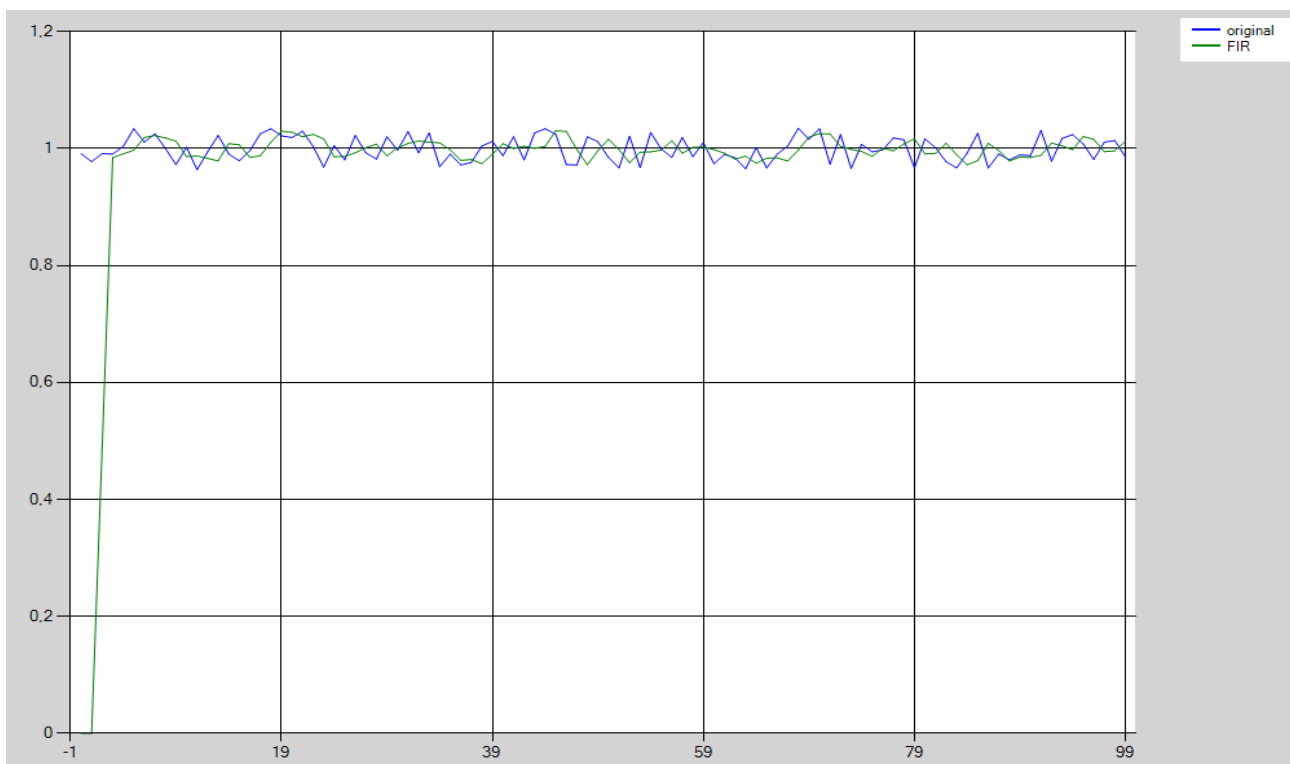


Рисунок – 3.4 Експериментальне дослідження сигналу за допомогою FIR фільтру

При натиску на кнопку LMS filter генеруємо фільтрований сигнал LMS фільтр (рисунок 3.5, рисунок 3.6):

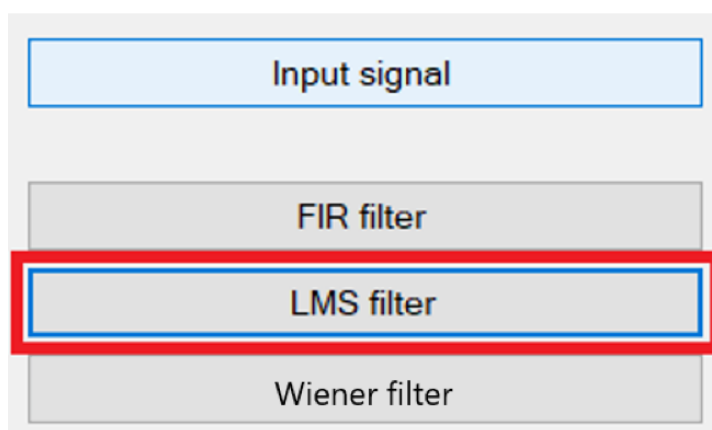


Рисунок 3.5 – Вибір LMS фільтру в меню програми

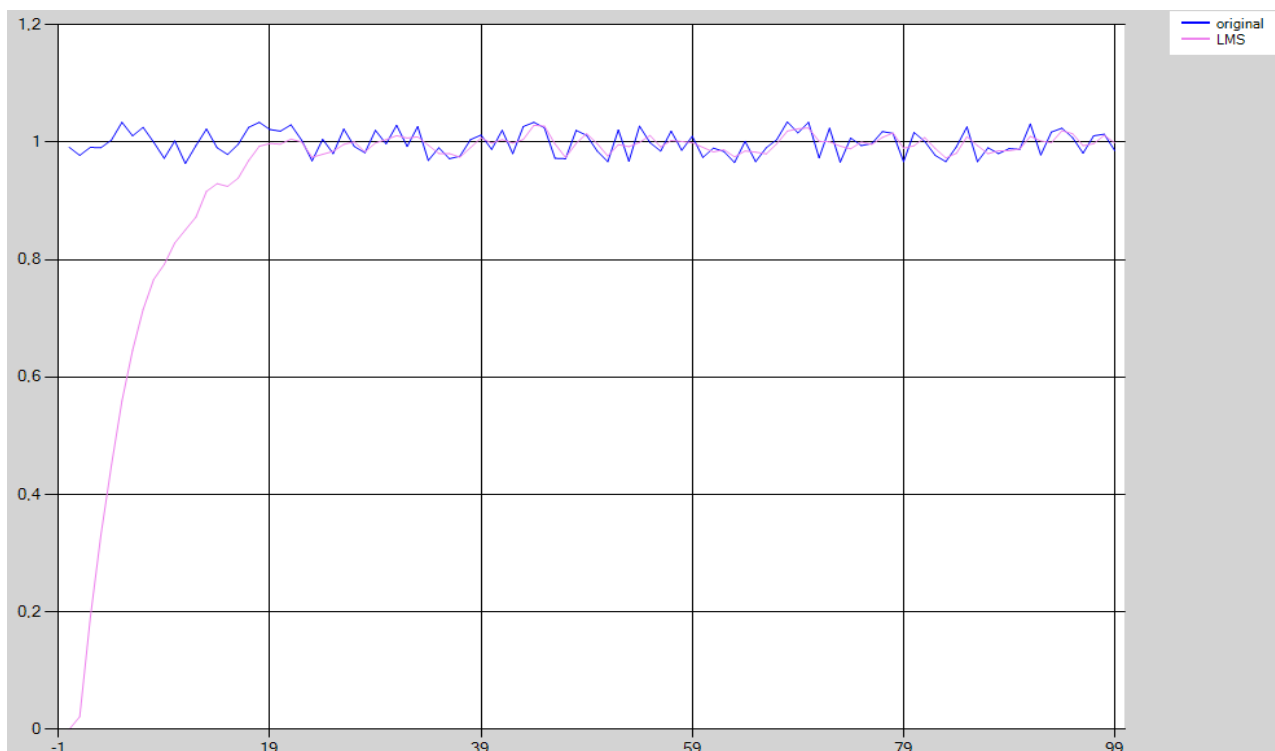


Рисунок 3.6 – Експериментальне дослідження сигналу за допомогою LMS алгоритму та відфільтровані дані.

Далі підключаємо фільтр Вінера за допомогою кнопки «Wiener filter» (рисунок. 3.7):

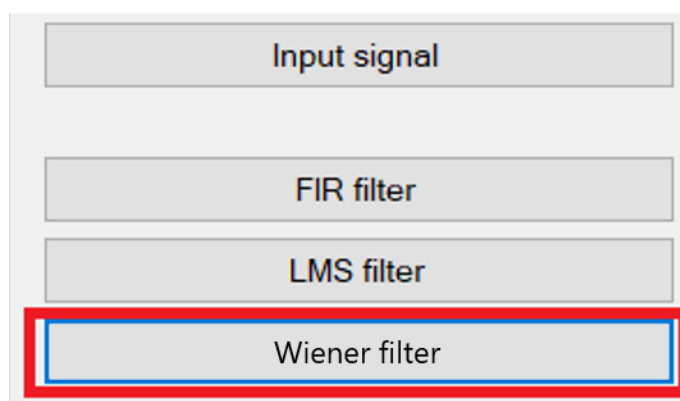


Рисунок 3.7 – Вибір фільтру Вінера в графі меню програми

На рисунку 3.8, можемо згенерований сигнал та відфільтрований фільтром Вінера дані:

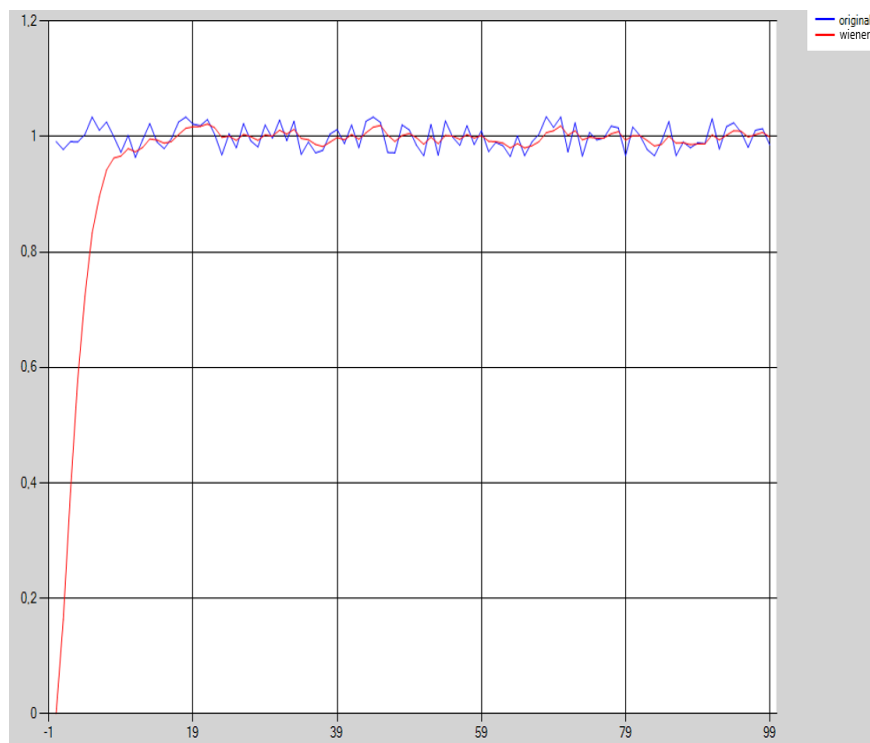


Рисунок 3.8 – Експериментальне дослідження сигналу за допомогою фільтру Вінера (LMS неадаптивний алгоритм)

На рисунку 3.9 наведена порівняльна характеристика фільтрації функції Хевісайда з різними типами фільтрів.

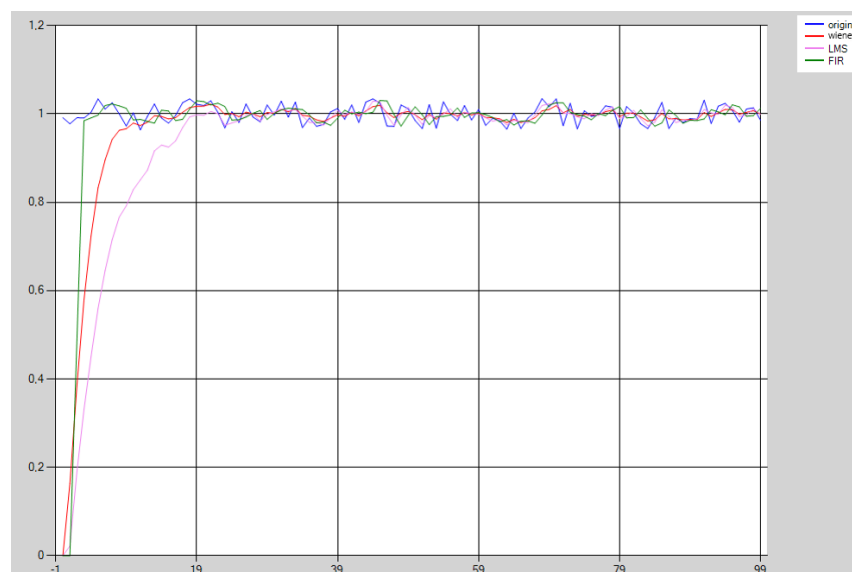


Рисунок 3.9 – Одиничний(окремий) сигнал та фільтри FIR, LMS, Вінера

На рисунку 3.10 можемо розглянути детальну фільтрацію сигналу різними типами фільтрів на відрізку від 0 до 60 ітерацій:

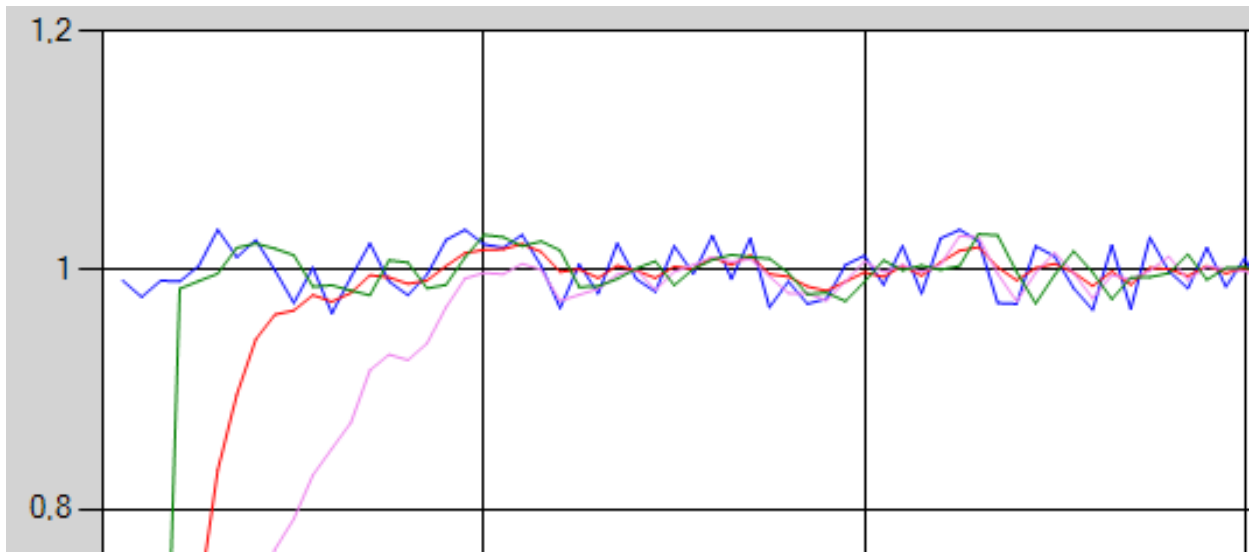


Рисунок 3.10 Ретельний розгляд фільтрації сигналу за допомогою фільтру FIR, LMS(адаптивний фільтр), Wiener.

На рисунку можна побачити одиничний(окремий) вхідний сигнал з похибкою АЦП (на рисунку 3.10 зображено блакитним кольором) та результат фільтрації трьох алгоритмів:

- 1 Wiener filter (червоний колір) – виконується фільтрація так, що вихідні дані майже наближені до;
- 2 FIR filter (зелений колір) – виконується згладжування піків сигналу та виконує зсув сигналу вперед;
- 3 LMS filter (рожевий колір) – виконується наближення до вихідного сигналу, але з менш згладженими піками;

Отже, розглянувши прямий(окремий) сигнал, та його методи фільтрування, можемо зробити висновок, що прямого сигналу (функція Хевісайда), досить непоганими методами фільтрації є метод фільтрації Вінера та LMS фільтр (найменших середніх квадратів).

### 3.2 Синусоїдальний сигнал зі зміною за дисперсією

Для фільтрації різними типами фільтрів спочатку згенеруємо окремий синусоїдальний сигнал з додаванням до нього похибки АЦП:

Реалізація окремого синусоїдального сигналу з похибкою АЦП в коді приведена в Додатку В.

Для того, щоб згенерувати сигнал у робочій моделі потрібно обрати тип сигналу (Рисунок 3.11):

- 1 Обираємо синусоїдальний тип сигналу;
- 2 Тиснемо на кнопку виводу сигналу;

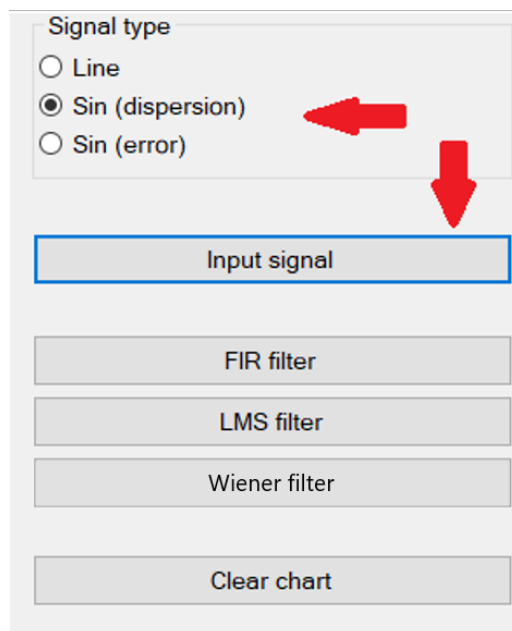


Рисунок 3.11 – Вибір сигналу з похибкою АЦП в моделі інтерфейсу програми

Згенерований сигнал можна побачити у вікні графіку (рисунок 3.12):

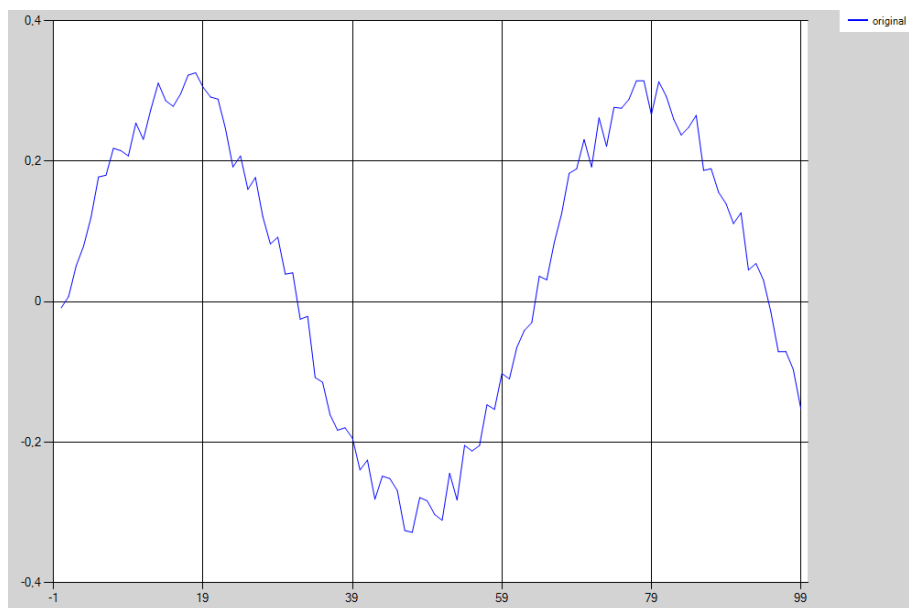


Рисунок 3.12 – Генерація сигналу синусу з похибкою АЦП за дисперсією в програмі

Моделюємо генерацію фільтру за допомогою FIR фільтру. Обираємо в інтерфейсі меню кнопок FIR filter, як показано на рисунку 3.3 і отримуємо у вікні графіку відфільтрований сигнал (рисунок. 3.13)

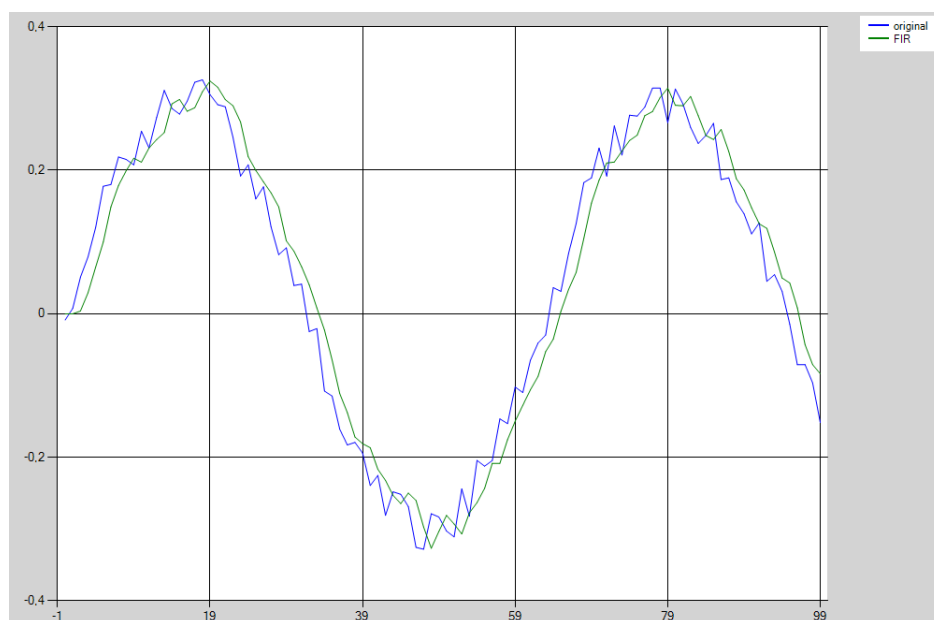


Рисунок 3.13 – Експериментальне дослідження сигналу за допомогою алгоритму FIR фільтру

Виконуємо модуляцію фільтру LMS, обравши перед цим кнопку «LMS filter».  
Отримуємо змодульований сигнал (рисунок 3.14):

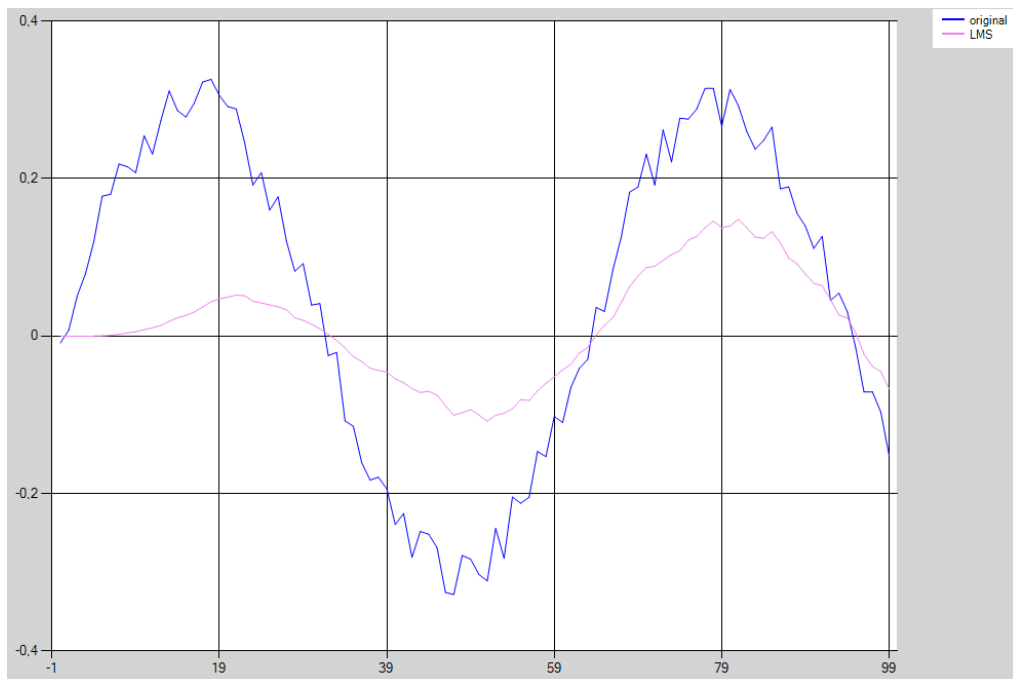


Рисунок 3.14 – Експериментальне дослідження сигналу за допомогою алгоритму LMS(адаптивний алгоритм)

Підключаємо фільтр Вінера за допомогою кнопки «Wiener filter»

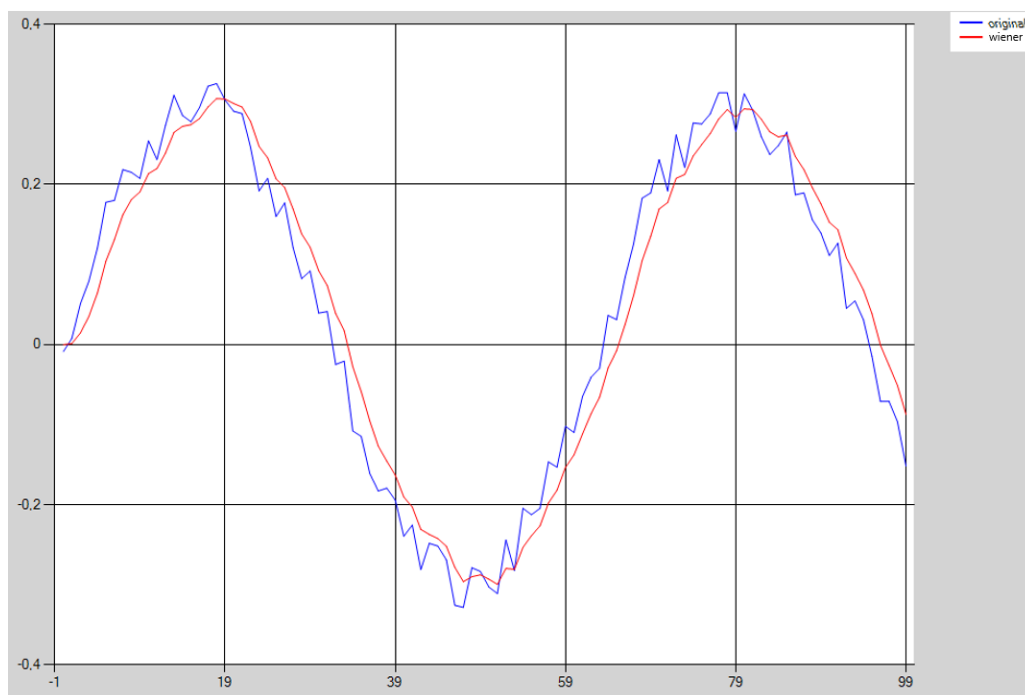


Рисунок 3.15 – Експериментальне дослідження сигналу за допомогою алгоритму LMS(неадаптивний алгоритм) фільтр Вінера

Порівняльна характеристика різними типами фільтрів синусоїдального сигналу зображена на рисунку 3.16:

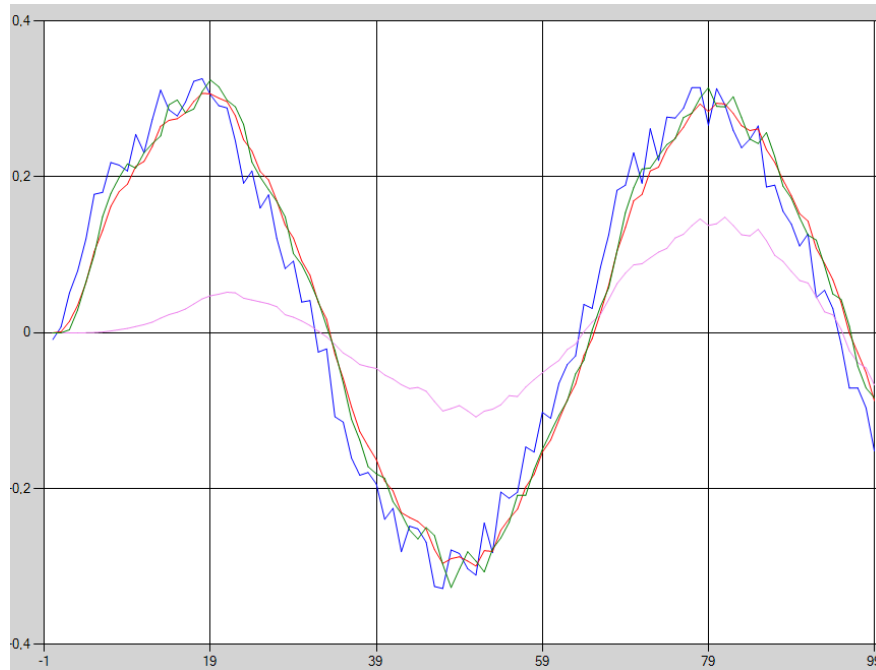


Рисунок 3.16 – Експериментальне дослідження синусоїдального сигналу з похибкою АЦП за дисперсією. Фільтри FIR, LMS, Wiener.

На рисунку отримали результати використання трьох алгоритмів фільтрації та синусоїдальний сигнал з похибкою АЦП, яка змінюється за дисперсією (синій колір):

1 FIR filter(фільтр з скінченною імпульсною характеристикою) (зелений колір), який наближає сигнал до початкового синусоїдального сигналу та згладжує його;

2 LMS filter (фільтр найменших квадратів)(рожевий колір) – виконує фільтрацію так, що сигнал на виході має меншу амплітуду, але виходить досить добре згладженим;



3 Wiener filter (фільтр Вінера) (червоний колір) – виконує фільтрацію та реалізує сигнал на виході який майже ідентичний до початкового сигналу, з майже ідеальними параметрами.

Отже, було проведено експериментальне дослідження синусоїдального сигналу та його методів фільтрації, можемо прийти до висновку, що для такого сигналу оптимальними методами фільтрації є метод фільтрації за допомогою фільтру Вінера (Wiener filter), та LMS фільтру (фільтру з скінченною імпульсною характеристикою). LMS фільтр має високий показник згладжування шуму, проте видає менш якісний результат.

### 3.3 Синусоїдальний сигнал зі зміною за похибкою

Для початку генеруємо синусоїдальний сигнал з додаванням до нього похибки АЦП

Реалізація коду синусоїдального сигналу з похибкою приведена в Додатку Г

Для того, щоб згенерувати сигнал у графічному інтерфейсі потрібно обрати тип сигналу (Рисунок 3.17):

- Тип сигналу – Sin (error);
- Натиснути кнопку для виводу сигналу на графік Input Signal;

Signal type

☐ Line
☐ Sin (dispersion)
☒ Sin (error)

Input signal

FIR filter

LMS filter

Kalman filter

Clear chart

Рисунок 3.17 – Вибір типу сигналу в графічному інтерфейсу програми

Згенерований сигнал можна побачити у вікні графіку (рисунок 3.18):

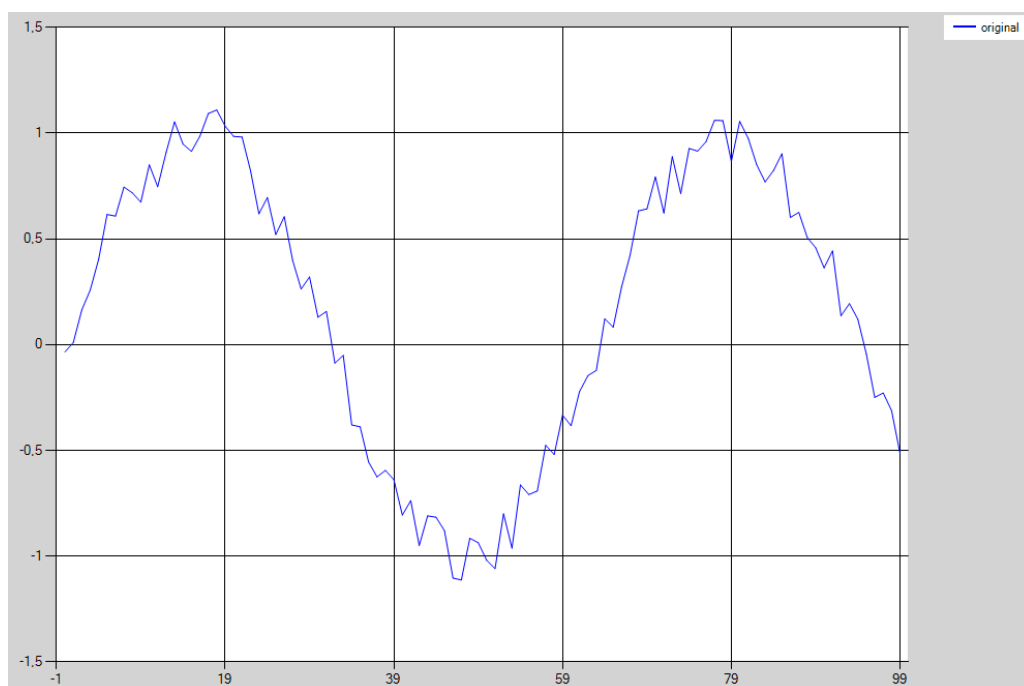


Рисунок 3.18 – Моделювання окремого синусоїдального сигналу з похибкою АЦП

Проведемо генерацію фільтру із скінченною імпульсною характеристикою. В меню інтерфейсу кнопок обираємо FIR filter. Отриманий змодульований можемо розглянути на рисунку 3.19:

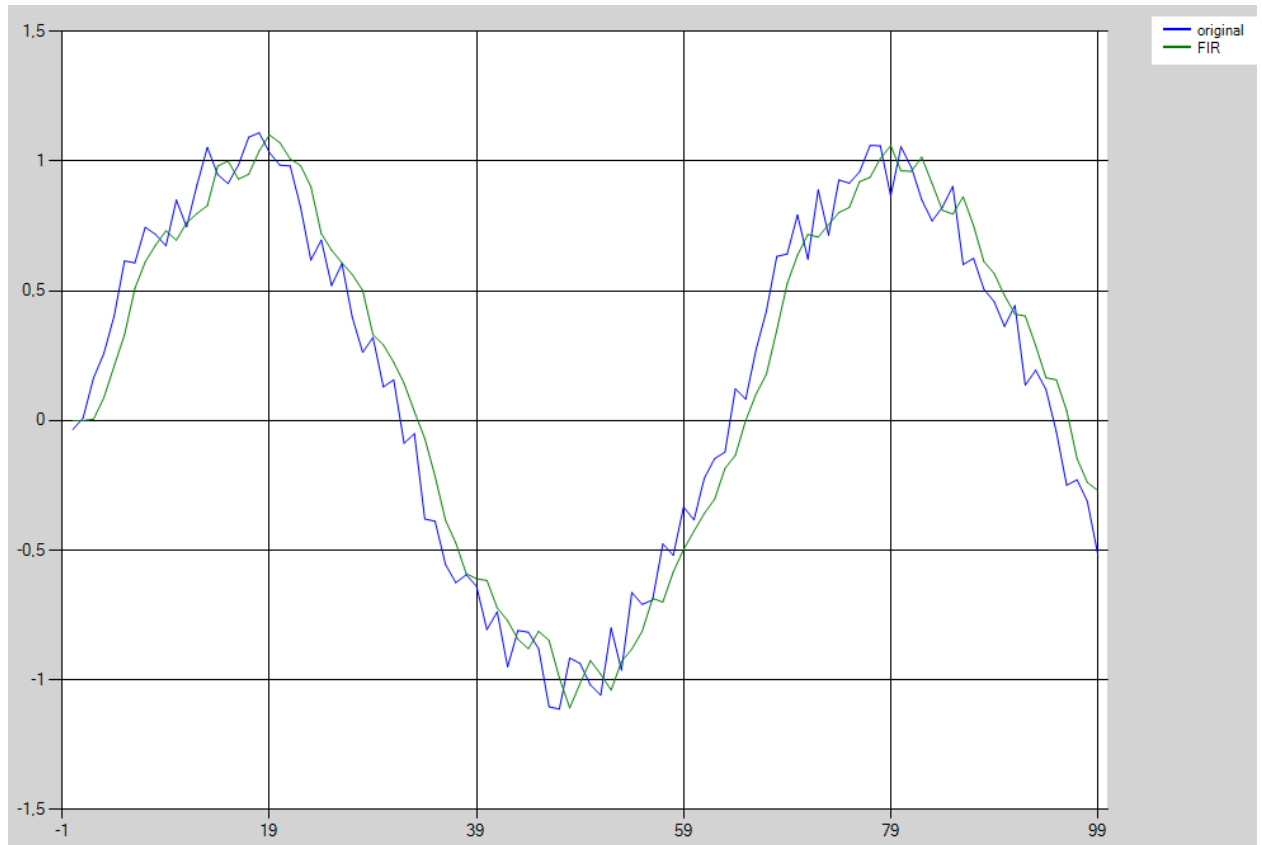


Рисунок 3.19 – Експериментальне дослідження синусоїдального сигналу з похибкою, фільтр з безкінечною імпульсною характеристикою (FIR)

Виконуємо модуляцію LMS фільтру натиснувши на кнопку «LMS filter» отриманий сигнал виводимо на графік (рисунок 3.20):

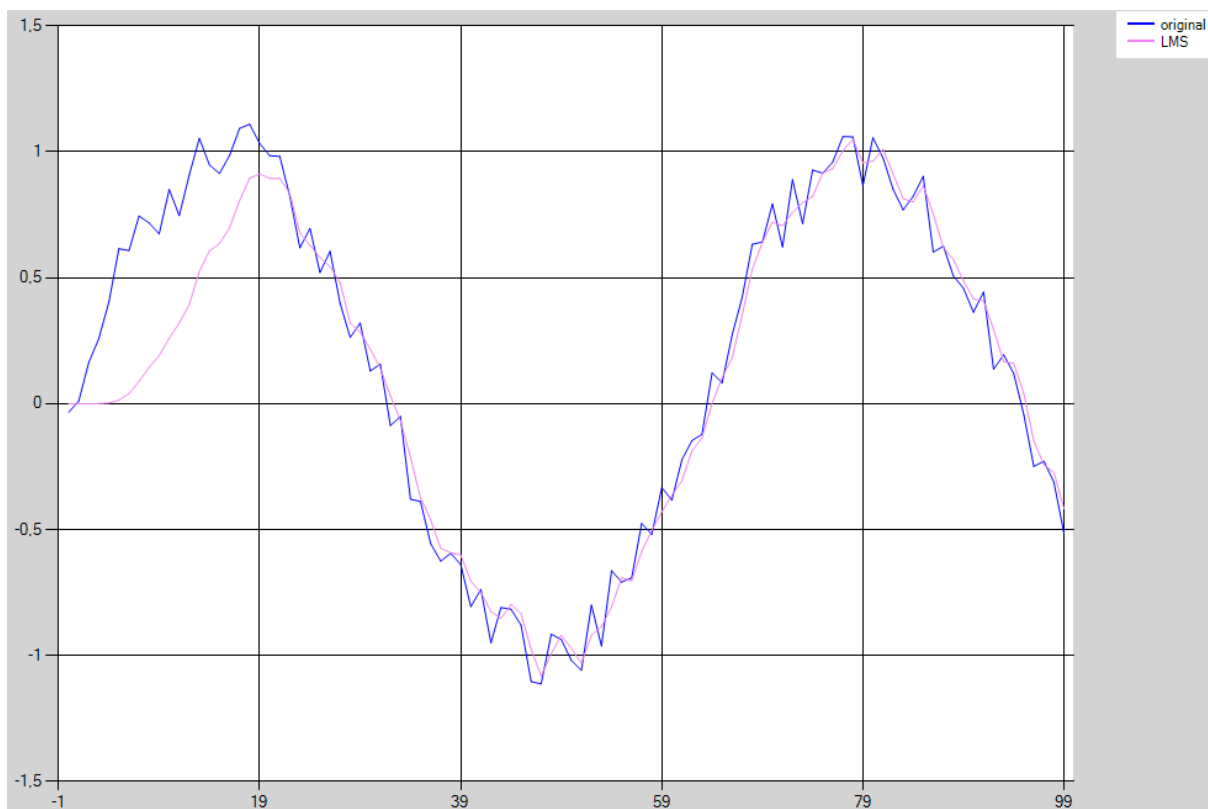


Рисунок 3.20 – Експериментальне дослідження синусоїдального сигналу з похибкою, LMS фільтр(адаптивний фільтр)

Підключимо фільтр Вінера за допомогою кнопки «Wiener filter», як зображено на рисунку 3.7 та результат фільтрації зображено на рисунку 3.21:

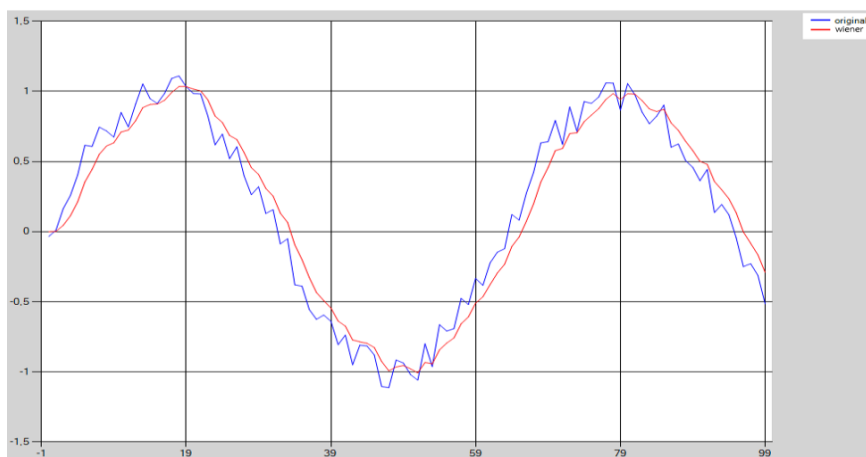


Рисунок 3.21 – Експериментальне дослідження синусоїдального сигналу з похибкою, LMS фільтр(неадаптивний фільтр). Wiener фільтр.

Порівняльна характеристика фільтрації синусоїдального сигналу різними типами фільтрів зображена на рисунку 3.22:

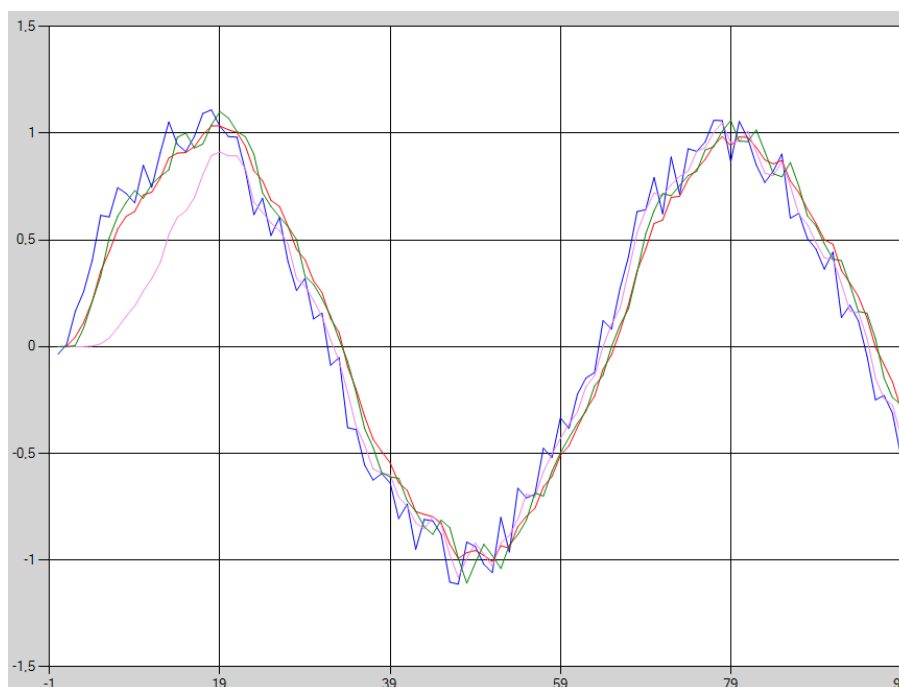


Рисунок 3.22 – Експериментальне дослідження синусоїдального сигналу з похибкою. Фільтр FIR, Вінера, LMS.

На рисунку 3.22 зображено синусоїдальний вхідний сигнал з додаванням похибки АЦП, який змінюється за похибкою (зображений синім кольором) та результати використання обраних алгоритмів фільтрації:

- 1 FIR filter (зелений колір) – виконує згладжування піків сигналу таким чином, що наближає його до значення між початковим сигналом та сигналом з шумом;
- 2 Фільтр найменших квадратів (рожевий колір) – виконує фільтрацію таким чином, що при заданих параметрах сигналу, відфільтровані дані більш наближені до вихідних даних;
- 3 Фільтр Вінера (червоний колір) – виконується фільтрація сигналу з похибкою таким чином, що вихідні дані з фільтру мають майже ідеальні характеристики та фільтрований сигнал виходить найбільш наближеним до початкового синусоїдального сигналу.

Після експериментально дослідження синусоїдального сигналу та його методів фільтрації, можемо прийти до того, що для синусоїдального сигналу оптимальним та ефективним методом фільтрації є метод фільтрації Вінера. Це пояснюється тим, що даний алгоритм виконує фільтрацію таким чином, що вихідні дані мають наближені характеристики до сигналу без похибки. Порівнюючи його з такими методами фільтрації як LMS фільтр та FIR фільтр, то вони показують менш бажаний результат, проте при певному типі сигналу ці фільтри мають досить високий показник згладжування.

					BB71.120004.001	Арк.
						52
Зм.	Арк.	№ докум.	Підп.	Дата		

## ВИСНОВКИ ДО РОЗДІЛУ 3

В розділі був проведений експеримент, який визначав ефективність методів фільтрації серед 3х використаних широковідомих фільтрів.

На основі розробленої програмної моделі та експериментального дослідження, в якому був проведений порівняльний аналіз алгоритмів, та методів фільтрації, можна зробити висновок, що алгоритм фільтрації Вінера в порівнянні являється достатньо ефективний та точним. Результати які ми отримали на виході відповідають ідеальним характеристикам сигналів та являються найбільш наближеними до початкових параметрів сигналів.

					BB71.120004.001	Арк.
						53
Зм.	Арк.	№ докум.	Підп.	Дата		

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1 А. Павлов. Вейвлет-анализ структуры сигналов: Нестационарные, короткие и зашумлённые процессы (Russian Edition), 2011 – 15-36 ст.
- 2 Лезин Ю. С. Оптимальные фильтры и накопители импульсных сигналов. Издание второе, переработанное и дополненное. Изд-во «Советское радио», 448 стр.
- 3 А. Н. Колмогоров. Введение в анализ. Москва: Издательство Московского университета, 1966р.
- 4 Гельфанд А. М., Хмельник С. И. Цифровая фильтрация многомерных взаимозависимых нестационарных процессов – М: Lulu Inc. 200 г – 19-52 ст.
- 5 Шахратин Б.И. Фильтр Виннера и Калмана. Учебное пособие для вузов, 2-е издание, 2015г. - 396с.
- 6 Никитин Н.П., Лузин В.И. Прием и обработка сигналов и цифровых системах передачи: учеб. пособие, 2013г. – 124с.
- 7 Притула. Магістерська дисертація. Учебное заведение: National Technical University of Ukraine Kyiv Politechnic Institute .



## ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

```
#pragma once
```

```
#ifndef generate_h
```

```
#define generate_h
```

```
#include <iostream>
```

```
void generate_line(double* input, int size)
```

```
{
```

```
    srand(0);
```

```
    for (int i = 0; i < size; i++)
```

```
        input[i] = 1. + ((rand() % 100 / 100. - 0.5) / sqrt(12)) / 4;
```

```
}
```

```
void generate_sin_dis(double* input, int size)
```

```
{
```

```
    srand(0);
```

```
    for (int i = 0; i < size; i++)
```

```
        input[i] = 0.3 * sin(0.1 * i) + ((rand() % 100 / 100. - 0.5) / sqrt(12)) / 4;
```

```
}
```

					BB71.120004.001	Арк.
						55
Зм.	Арк.	№ докум.	Підп.	Дата		

```

void generate_sin_err(double* input, int size)

{

    srand(0);

    for (int i = 0; i < size; i++)

        input[i] = sin(0.1 * i) + ((rand() % 100 / 100. - 0.5) / sqrt(12)) / 1;

}

```

```

#endif // !generate_h

```

```

#pragma once

```

```

#pragma once

```

```

#ifndef fir_h

```

```

#define fir_h

```

```

#include <iostream>

```

```

#define M_PI 3.1415926

```

```

void fir(double in[], double out[], int sizeIn)

```

```

{

```

```

    const int N = 4;

```

```

    double Fd = 200;

```

					BB71.120004.001	Арк.
						56
Зм.	Арк.	№ докум.	Підп.	Дата		

```
double Fs = 1;
```

```
double Fx = 2;
```

```
long double H[N] = { 0 };
```

```
long double H_id[N] = { 0 };
```

```
long double W[N] = { 0 };
```

```
double Fc = (Fs + Fx) / (2 * Fd);
```

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

```
    if (i == 0) H_id[i] = 2 * M_PI*Fc;
```

```
    else H_id[i] = sinl(2 * M_PI*Fc*i) / (M_PI*i);
```

```
    W[i] = 0.5 - 0.5 * cos((2 * M_PI * i) / (N - 1));
```

```
    H[i] = H_id[i] * W[i];
```

```
}
```

```
double SUM = 0;
```

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

```
    SUM += H[i];
```

					BB71.120004.001	Арк.
						57
Зм.	Арк.	№ докум.	Підп.	Дата		

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

```
    H[i] /= SUM;
```

```
for (int i = 0; i < sizeIn + 1; i++) {
```

```
    out[i] = 0.;
```

```
    for (int j = 0; j < N - 1; j++)
```

```
        if (i - j > 0)
```

```
            out[i] += H[j] * in[i - j];
```

```
    }
```

```
}
```

```
#endif // !fir_h
```

```
#pragma once
```

```
#ifndef lms_h
```

```
#define lms_h
```

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

					BB71.120004.001	Арк. 58
Зм.	Арк.	№ докум.	Підп.	Дата		

```

#define mu 0.09

#define M 2

//

////double H[M] = { 1, 0.5, 0.25, 0.125, 0.0625 }; //the main system

double H[M] = { /* 0.0625, 0.125, */ 0.25, 0.75 };

void lms(double* input, double * output, int size)

{

    double D, Y, E;

    double W[M] = { 0.0 };

    double X[M] = { 0.0 };

    double* Desired = new double[size];

    for (int i = 0; i < size; i++)

        Desired[i] = { 0.0 };

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

        for (int j = 0; j < M; j++) {

            if (i - j >= 0)

                Desired[i] += input[i - j] * H[j];

        }

    }

```

```

for (int T = 0; T < size; T++) {

    for (int m = T; m > T - M; m--) {

        if (m >= 0)

            X[M + (m - T) - 1] = input[m];

        else break;

    }

    D = Desired[T];

    Y = 0;

    for (int i = 0; i < M; i++)

        Y += (W[i] * X[i]);

    E = D - Y;

    for (int i = 0; i < M; i++)

        W[i] = W[i] + (mu * E * X[i]);

    output[T] = Y;

}

}

```

					BB71.120004.001	Арк.
						60
Зм.	Арк.	№ докум.	Підп.	Дата		

```
#endif // !lms_h
```

```
#pragma once
```

```
#using <system.dll>
```

```
#include <math.h>
```

```
#include <iostream>
```

```
#include <ctime>
```

```
#include "fir.h"
```

```
#include "wiener.h"
```

```
#include "lms.h"
```

```
#include "generate.h"
```

```
namespace optimalfilters {
```

```
#define I 100
```

```
using namespace System;
```

```
using namespace System::ComponentModel;
```

```
using namespace System::Collections;
```

```
using namespace System::Windows::Forms;
```

```
using namespace System::Data;
```

```
using namespace System::Drawing;
```

					BB71.120004.001	Арк.
						61
Зм.	Арк.	№ докум.	Підп.	Дата		

```
//using namespace System::IO;

using namespace System::Collections::Generic;

using namespace System::Drawing::Drawing2D;

//using namespace System::Drawing;

using namespace System::Windows::Forms::DataVisualization::Charting;
```

```
double* input_signal = new double[I];
```

```
/// <summary>
```

```
/// Сводка для FormMain
```

```
/// </summary>
```

```
public ref class FormMain : public System::Windows::Forms::Form
```

```
{
```

```
public:
```

```
    FormMain(void)
```

```
    {
```

```
        InitializeComponent();
```

```
        //
```

```
        //TODO: добавьте код конструктора
```

```
        //
```

```
    }
```

					BB71.120004.001	Арк.
						62
Зм.	Арк.	№ докум.	Підп.	Дата		



protected:

```
/// <summary>
```

```
/// Освободить все используемые ресурсы.
```

```
/// </summary>
```

```
~FormMain()
```

```
{
```

```
    if (components)
```

```
    {
```

```
        delete components;
```

```
    }
```

```
}
```

protected:

```
private: System::Windows::Forms::RadioButton^ radioButton_sin;
```

```
private: System::Windows::Forms::RadioButton^ radioButton_line;
```

```
private: System::Windows::Forms::Button^ button_wiener;
```

```
private: System::Windows::Forms::Button^ button_lms;
```

```
private: System::Windows::Forms::Button^ button_fir;
```

```
private: System::Windows::Forms::Button^ button_original;
```

```
private: System::Windows::Forms::Button^ button_clear;
```

					BB71.120004.001	Арк.
						63
Зм.	Арк.	№ докум.	Підп.	Дата		

```
private: System::Windows::Forms::DataVisualization::Charting::Chart^ chart1;

private: System::Windows::Forms::RadioButton^ radioButton_sin_err;

private: System::Windows::Forms::SplitContainer^ splitContainer1;

private: System::Windows::Forms::GroupBox^ groupBox_signal_type;
```

```
private:
```

```
/// <summary>
```

```
/// Обязательная переменная конструктора.
```

```
/// </summary>
```

```
System::ComponentModel::Container ^components;
```

```
#pragma region Windows Form Designer generated code
```

```
/// <summary>
```

```
/// Требуемый метод для поддержки конструктора — не изменяйте
```

```
/// содержимое этого метода с помощью редактора кода.
```

```
/// </summary>
```

```
void InitializeComponent(void)
```

```
{
```

```
        System::Windows::Forms::DataVisualization::Charting::ChartArea^
chartArea2 = (gcnew System::Windows::Forms::DataVisualization::Charting::Char-
tArea());
```

					BB71.120004.001	Арк.
						64
Зм.	Арк.	№ докум.	Підп.	Дата		

```

        System::Windows::Forms::DataVisualization::Charting::Legend^
legend2 = (gcnew System::Windows::Forms::DataVisualization::Charting::Legend());

        System::Windows::Forms::DataVisualization::Charting::Series^ se-
ries2 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

        this->radioButton_sin_err = (gcnew System::Windows::Forms::Radi-
oButton());

        this->button_clear = (gcnew System::Windows::Forms::Button());

        this->radioButton_sin = (gcnew System::Windows::Forms::Radio-
Button());

        this->radioButton_line = (gcnew System::Windows::Forms::Radio-
Button());

        this->button_wiener = (gcnew System::Windows::Forms::Button());

        this->button_lms = (gcnew System::Windows::Forms::Button());

        this->button_fir = (gcnew System::Windows::Forms::Button());

        this->button_original = (gcnew System::Windows::Forms::Button());

        this->chart1 = (gcnew System::Windows::Forms::DataVisualiza-
tion::Charting::Chart());

        this->splitContainer1 = (gcnew System::Windows::Forms::SplitCon-
tainer());

        this->groupBox_signal_type = (gcnew System::Win-
dows::Forms::GroupBox());

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->BeginInit();

```

```

        (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->splitContainer1))->BeginInit();

        this->splitContainer1->Panel1->SuspendLayout();

        this->splitContainer1->Panel2->SuspendLayout();

        this->splitContainer1->SuspendLayout();

        this->groupBox_signal_type->SuspendLayout();

        this->SuspendLayout();

        //

        // radioButton_sin_err

        //

        this->radioButton_sin_err->AutoSize = true;

        this->radioButton_sin_err->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular,

            System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(204)));

        this->radioButton_sin_err->Location = System::Drawing::Point(4, 66);

        this->radioButton_sin_err->Margin = System::Windows::Forms::Padding(2, 2, 2, 2);

        this->radioButton_sin_err->Name = L"radioButton_sin_err";

        this->radioButton_sin_err->Size = System::Drawing::Size(91, 21);

        this->radioButton_sin_err->TabIndex = 7;

```

					BB71.120004.001	Арк.
						66
Зм.	Арк.	№ докум.	Підп.	Дата		

```

this->radioButton_sin_err->TabStop = true;

this->radioButton_sin_err->Text = L"Sin (error)";

this->radioButton_sin_err->UseVisualStyleBackColor = true;

//

// button_clear

//

this->button_clear->Anchor          =          static_cast<System::Win-
dows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Left   |   Sys-
tem::Windows::Forms::AnchorStyles::Right));

this->button_clear->Font = (gcnew System::Drawing::Font(L"Mi-
crosoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Draw-
ing::GraphicsUnit::Point,

static_cast<System::Byte>(204)));

this->button_clear->Location = System::Drawing::Point(11, 337);

this->button_clear->Margin = System::Windows::Forms::Padding(2,
2, 2, 2);

this->button_clear->Name = L"button_clear";

this->button_clear->Size = System::Drawing::Size(144, 30);

this->button_clear->TabIndex = 6;

this->button_clear->Text = L"Clear chart";

this->button_clear->UseVisualStyleBackColor = true;

this->button_clear->Click += gcnew System::EventHandler(this,
&FormMain::button_clear_Click);

```

					BB71.120004.001	Арк.
						67
Зм.	Арк.	№ докум.	Підп.	Дата		

```

//

// radioButton_sin

//

this->radioButton_sin->AutoSize = true;

this->radioButton_sin->Font = (gcnew System::Drawing::Font(L"Mi-
crosoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Draw-
ing::GraphicsUnit::Point,

static_cast<System::Byte>(204)));

this->radioButton_sin->Location = System::Drawing::Point(4, 44);

this->radioButton_sin->Margin = System::Windows::Forms::Pad-
ding(2, 2, 2, 2);

this->radioButton_sin->Name = L"radioButton_sin";

this->radioButton_sin->Size = System::Drawing::Size(125, 21);

this->radioButton_sin->TabIndex = 5;

this->radioButton_sin->Text = L"Sin (dispersion)";

this->radioButton_sin->UseVisualStyleBackColor = true;

//

// radioButton_line

//

this->radioButton_line->AutoSize = true;

this->radioButton_line->Checked = true;

```

```

        this->radioButton_line->Font      =      (gcnew      System::Draw-
ing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular,

        System::Drawing::GraphicsUnit::Point,      static_cast<Sys-
tem::Byte>(204)));

        this->radioButton_line->Location = System::Drawing::Point(4, 22);

        this->radioButton_line->Margin  =  System::Windows::Forms::Pad-
ding(2, 2, 2, 2);

        this->radioButton_line->Name = L"radioButton_line";

        this->radioButton_line->Size = System::Drawing::Size(53, 21);

        this->radioButton_line->TabIndex = 4;

        this->radioButton_line->TabStop = true;

        this->radioButton_line->Text = L"Line";

        this->radioButton_line->UseVisualStyleBackColor = true;

        //

        // button_wiener

        //

        this->button_wiener->Anchor      =      static_cast<System::Win-
dows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Left   |   Sys-
tem::Windows::Forms::AnchorStyles::Right));

        this->button_wiener->Enabled = false;

        this->button_wiener->Font = (gcnew System::Drawing::Font(L"Mi-
crosoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Draw-
ing::GraphicsUnit::Point,

```

```

        static_cast<System::Byte>(204)));

this->button_wiener->Location = System::Drawing::Point(11, 281);

this->button_wiener->Margin    =    System::Windows::Forms::Pad-
ding(2, 2, 2, 2);

this->button_wiener->Name = L"button_wiener";

this->button_wiener->Size = System::Drawing::Size(144, 30);

this->button_wiener->TabIndex = 3;

this->button_wiener->Text = L"Wiener filter";

this->button_wiener->UseVisualStyleBackColor = true;

this->button_wiener->Click += gcnew System::EventHandler(this,
&FormMain::button_wiener_Click);

//

// button_lms

//

this->button_lms->Anchor          =          static_cast<System::Win-
dows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Left    |    Sys-
tem::Windows::Forms::AnchorStyles::Right));

this->button_lms->Enabled = false;

this->button_lms->Font    =    (gcnew System::Drawing::Font(L"Mi-
crosoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Draw-
ing::GraphicsUnit::Point,

        static_cast<System::Byte>(204)));

this->button_lms->Location = System::Drawing::Point(11, 246);

```

					BB71.120004.001	Арк.
						70
Зм.	Арк.	№ докум.	Підп.	Дата		



```

this->button_lms->Margin = System::Windows::Forms::Padding(2,
2, 2, 2);

this->button_lms->Name = L"button_lms";

this->button_lms->Size = System::Drawing::Size(144, 30);

this->button_lms->TabIndex = 2;

this->button_lms->Text = L"LMS filter";

this->button_lms->UseVisualStyleBackColor = true;

this->button_lms->Click += gcnew System::EventHandler(this,
&FormMain::button_lms_Click);

//

// button_fir

//

this->button_fir->Anchor = static_cast<System::Win-
dows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Left | Sys-
tem::Windows::Forms::AnchorStyles::Right));

this->button_fir->Enabled = false;

this->button_fir->Font = (gcnew System::Drawing::Font(L"Microsoft
Sans Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::Graphics-
Unit::Point,

static_cast<System::Byte>(204)));

this->button_fir->Location = System::Drawing::Point(11, 211);

this->button_fir->Margin = System::Windows::Forms::Padding(2, 2,
2, 2);

```

					BB71.120004.001	Арк.
						71
Зм.	Арк.	№ докум.	Підп.	Дата		

```

this->button_fir->Name = L"button_fir";

this->button_fir->Size = System::Drawing::Size(144, 30);

this->button_fir->TabIndex = 1;

this->button_fir->Text = L"FIR filter";

this->button_fir->UseVisualStyleBackColor = true;

this->button_fir->Click += gcnew System::EventHandler(this,
&FormMain::button_fir_Click);

//

// button_original

//

this->button_original->Anchor = static_cast<System::Win-
dows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Left | Sys-
tem::Windows::Forms::AnchorStyles::Right));

this->button_original->Font = (gcnew System::Drawing::Font(L"Mi-
crosoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Draw-
ing::GraphicsUnit::Point,

static_cast<System::Byte>(204)));

this->button_original->Location = System::Drawing::Point(11, 153);

this->button_original->Margin = System::Windows::Forms::Pad-
ding(2, 2, 2, 2);

this->button_original->Name = L"button_original";

this->button_original->Size = System::Drawing::Size(144, 30);

this->button_original->TabIndex = 0;

```

					BB71.120004.001	Арк.
						72
Зм.	Арк.	№ докум.	Підп.	Дата		

```

this->button_original->Text = L"Input signal";

this->button_original->UseVisualStyleBackColor = true;

this->button_original->Click += gcnew System::EventHandler(this,
&FormMain::button_original_Click);

//

// chart1

//

this->chart1->BackColor = System::Drawing::Color::LightGray;

chartArea2->Name = L"ChartArea1";

this->chart1->ChartAreas->Add(chartArea2);

this->chart1->Dock = System::Windows::Forms::DockStyle::Fill;

legend2->Name = L"Legend1";

this->chart1->Legends->Add(legend2);

this->chart1->Location = System::Drawing::Point(0, 0);

this->chart1->Margin = System::Windows::Forms::Padding(2, 2, 2,
2);

this->chart1->Name = L"chart1";

this->chart1->Palette = System::Windows::Forms::DataVisualiza-
tion::Charting::ChartColorPalette::Chocolate;

series2->ChartArea = L"ChartArea1";

series2->ChartType = System::Windows::Forms::DataVisualiza-
tion::Charting::SeriesChartType::Line;

```

```
series2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans
Serif", 10.2F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsU-
nit::Point,
```

```
static_cast<System::Byte>(204)));
```

```
series2->Legend = L"Legend1";
```

```
series2->Name = L"original";
```

```
this->chart1->Series->Add(series2);
```

```
this->chart1->Size = System::Drawing::Size(606, 505);
```

```
this->chart1->TabIndex = 1;
```

```
this->chart1->Click += gcnew System::EventHandler(this,
&FormMain::chart1_Click);
```

```
//
```

```
// splitContainer1
```

```
//
```

```
this->splitContainer1->Dock = System::Windows::Forms::Dock-
Style::Fill;
```

```
this->splitContainer1->Location = System::Drawing::Point(0, 0);
```

```
this->splitContainer1->Margin = System::Windows::Forms::Pad-
ding(2, 2, 2, 2);
```

```
this->splitContainer1->Name = L"splitContainer1";
```

```
//
```

```
// splitContainer1.Panel1
```

					BB71.120004.001	Арк.
						74
Зм.	Арк.	№ докум.	Підп.	Дата		

```

//

this->splitContainer1->Panel1->Controls->Add(this->chart1);

//

// splitContainer1.Panel2

//

this->splitContainer1->Panel2->Controls->Add(this->groupBox_sig-
nal_type);

this->splitContainer1->Panel2->Controls->Add(this->button_clear);

this->splitContainer1->Panel2->Controls->Add(this->button_origi-
nal);

this->splitContainer1->Panel2->Controls->Add(this->button_fir);

this->splitContainer1->Panel2->Controls->Add(this->button_lms);

this->splitContainer1->Panel2->Controls->Add(this->button_wie-
ner);

this->splitContainer1->Size = System::Drawing::Size(776, 505);

this->splitContainer1->SplitterDistance = 606;

this->splitContainer1->SplitterWidth = 3;

this->splitContainer1->TabIndex = 2;

//

// groupBox_signal_type

//

```

					BB71.120004.001	Арк.
						75
Зм.	Арк.	№ докум.	Підп.	Дата		

```

        this->groupBox_signal_type->Anchor = static_cast<System::Win-
dows::Forms::AnchorStyles>((System::Windows::Forms::AnchorStyles::Left | Sys-
tem::Windows::Forms::AnchorStyles::Right));

```

```

        this->groupBox_signal_type->Controls->Add(this->radio-
Button_line);

```

```

        this->groupBox_signal_type->Controls->Add(this->radio-
Button_sin);

```

```

        this->groupBox_signal_type->Controls->Add(this->radio-
Button_sin_err);

```

```

        this->groupBox_signal_type->Font = (gcnew System::Draw-
ing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::FontStyle::Regular,
        System::Drawing::GraphicsUnit::Point, static_cast<Sys-
tem::Byte>(204)));

```

```

        this->groupBox_signal_type->Location = System::Draw-
ing::Point(11, 25);

```

```

        this->groupBox_signal_type->Margin = System::Win-
dows::Forms::Padding(2, 2, 2, 2);

```

```

        this->groupBox_signal_type->Name = L"groupBox_signal_type";

```

```

        this->groupBox_signal_type->Padding = System::Win-
dows::Forms::Padding(2, 2, 2, 2);

```

```

        this->groupBox_signal_type->Size = System::Drawing::Size(144,
98);

```

```

        this->groupBox_signal_type->TabIndex = 8;

```

```

        this->groupBox_signal_type->TabStop = false;

```

					BB71.120004.001	Арк.
						76
Зм.	Арк.	№ докум.	Підп.	Дата		

```

this->groupBox_signal_type->Text = L"Signal type";

//

// FormMain

//

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScale-
Mode::Font;

this->ClientSize = System::Drawing::Size(776, 505);

this->Controls->Add(this->splitContainer1);

this->Margin = System::Windows::Forms::Padding(2, 2, 2, 2);

this->Name = L"FormMain";

this->Text = L"Optimal filters";

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>chart1))->EndInit();

this->splitContainer1->Panel1->ResumeLayout(false);

this->splitContainer1->Panel2->ResumeLayout(false);

(cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>splitContainer1))->EndInit();

this->splitContainer1->ResumeLayout(false);

this->groupBox_signal_type->ResumeLayout(false);

this->groupBox_signal_type->PerformLayout();

this->ResumeLayout(false);

```

```

    }

#pragma endregion

private: System::Void button_original_Click(System::Object^ sender, Sys-
tem::EventArgs^ e) {

    chart1->Series->Clear();

    chart1->Series->Add("original");

    chart1->Series["original"]->ChartType = SeriesChartType::Line;

    chart1->Series["original"]->Color = Color::Blue;

    if (radioButton_line->Checked) {

        generate_line(input_signal, I);

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

            chart1->Series["original"]->Points->AddXY(i, input_sig-
nal[i]);

        }

    }

    else if (radioButton_sin->Checked) {

        generate_sin_dis(input_signal, I);

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

            chart1->Series["original"]->Points->AddXY(i, input_sig-
nal[i]);

        }

    }

```

					BB71.120004.001	Арк.
						78
Зм.	Арк.	№ докум.	Підп.	Дата		



```

    }

    else {

        generate_sin_err(input_signal, I);

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

            chart1->Series["original"]->Points->AddXY(i,      input_sig-
nal[i]);

        }

    }

    button_fir->Enabled = true;

    button_wiener->Enabled = true;

    button_lms->Enabled = true;

}

```

```

private: System::Void button_fir_Click(System::Object^ sender, System::Even-
tArgs^ e) {

    double* filtered_fir = new double[I];

    fir(input_signal, filtered_fir, I);

    chart1->Series->Add("FIR");

    chart1->Series["FIR"]->ChartType = SeriesChartType::Line;

    chart1->Series["FIR"]->Color = Color::Green;

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

```

```

        chart1->Series["FIR"]->Points->AddXY(i, filtered_fir[i]);

    }

}

```

```

private: System::Void button_lms_Click(System::Object^ sender, System::EventArgs^ e) {

```

```

    double* filtered_lms = new double[I];

```

```

    lms(input_signal, filtered_lms, I);

```

```

    chart1->Series->Add("LMS");

```

```

    chart1->Series["LMS"]->ChartType = SeriesChartType::Line;

```

```

    chart1->Series["LMS"]->Color = Color::Violet;

```

```

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

```

```

        chart1->Series["LMS"]->Points->AddXY(i, filtered_lms[i]);

```

```

    }

```

```

}

```

```

private: System::Void button_wiener_Click(System::Object^ sender, System::EventArgs^ e) {

```

```

    double* filtered_wiener = new double[I];

```

```

    double F = 1; //factor of real value to previous real value

```

```

double Q = 2; // measurement noise

double H = 1; // factor of measured value to real value

double R = 10; // environment noise

Wiener_filter* wiener = new Wiener_filter(Q, R, F, H); // parameters of the filter

wiener->set_state(input_signal[0], 0.25);           // set the start value of State and
Covariance

for (int i = 0; i < I; i++) {                       //

    wiener->correct(input_signal[i]);                // using the wiener filter

    filtered_wiener[i] = wiener->state;              //

}

chart1->Series->Add("wiener");

chart1->Series["wiener"]->ChartType = SeriesChartType::Line;

chart1->Series["wiener"]->Color = Color::Red;

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

    chart1->Series["wiener"]->Points->AddXY(i, filtered_wiener[i]);

}

}

```

```

private: System::Void button_clear_Click(System::Object^ sender, System::EventArgs^ e) {

    chart1->Series->Clear();

    button_fir->Enabled = false;

    button_wiener->Enabled = false;

    button_lms->Enabled = false;

}

```

```

private: System::Void chart1_Click(System::Object^ sender, System::EventArgs^ e) {

}

};

}

```

```
#include "FormMain.h"
```

```
using namespace System;
```

```
using namespace System::Windows::Forms;
```

```
//using namespace std;
```

```
[STAThread]
```

```
int main() {
```

					BB71.120004.001	Арк.
						82
Зм.	Арк.	№ докум.	Підп.	Дата		

```

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

optimalfilters::FormMain form;

Application::Run(%form);

}

```

## ДОДАТОК Б

```

void generate_line(double* input, int size)

{

    srand(0);

    for (int i = 0; i < size; i++)

        input[i] = 1. + ((rand() % 100 / 100. - 0.5) / sqrt(12)) / 4;

}

```

## ДОДАТОК В

```

void generate_sin_dis(double* input, int size)

{

    srand(0);

    for (int i = 0; i < size; i++)

        input[i] = 0.3 * sin(0.1 * i) + ((rand() % 100 / 100. - 0.5) / sqrt(12)) / 4;

}

```

					BB71.120004.001	Арк.
						83
Зм.	Арк.	№ докум.	Підп.	Дата		

## ДОДАТОК Г

```
void generate_sin_err(double* input, int size)

{

    srand(0);

    for (int i = 0; i < size; i++)

        input[i] = sin(0.1 * i) + ((rand() % 100 / 100. - 0.5) / sqrt(12)) / 1;

}
```

					BB71.120004.001	Арк.
						84
Зм.	Арк.	№ докум.	Підп.	Дата		