НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ім. Ігоря Сікорського

Навчально-науковий комплекс «Інститут прикладного системного аналізу»
(повна назва інституту/факультету)

Кафедра Системного проектування
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 Комп’ютерні науки
(код і назва)

на тему: Сайт розрахунку цифрових фільтрів. Підсистема розрахунків _____
_____
_____

Виконав (-ла): студент __4__ курсу, групи ____ДА-32_____
(шифр групи)

_____Пиж Богдан Андрійович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник_____старший викладач Британов О.А _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант економічний ____доцент к. е. н. Рощина Н. В. ___ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Нормоконтроль _____старший викладач Британов О.А._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2017 року

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського

Інститут (факультет) ННК «Інститут прикладного системного аналізу
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050101 Комп'ютерні науки
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Пижу Богдану Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Сайт розрахунку цифрових фільтрів. Підсистема розрахунків, керівник роботи_____старший викладач Бритов О.А_____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «10»___05____ 2017 р. № 1477-с

2. Термін подання студентом роботи _____09.06.2017 _____

3. Вихідні дані до роботи
Розробити додаток, що виконує розрахунок рекурсивних цифрових фільтрів низьких частот, високих частот, смугових та режекторних фільтрів з використанням апроксимуючих функцій Батерворта, Чебишова та Кауера. Вхідні дані: межі та відхилення смуг пропускання та затримання, частота дискретизації, тип фільтру та апроксимації, порядок фільтру. Вихідні дані: нулі та полюси фільтру, коефіцієнти передаточної функції, АЧХ, ЛАЧХ, ФЧХ, імпульсна та перехідна характеристики.

4. Зміст роботи
1) Проаналізувати існуючі сайти розрахунку цифрових фільтрів.
2) Обрати програмні засоби та методику розрахунку.
3) Дослідити математичні основи розрахунку цифрових фільтрів.
4) Розробити додаток, що реалізує досліджені математичні закономірності.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1) Презентація.
2) Вимоги до АЧХ цифрового фільтра – плакат.
3) Архітектура додатку – плакат.
4) Алгоритм розрахунку цифрового фільтра – плакат.

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01.02.2017

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Збір інформації	17.02.2017	
3	Аналіз існуючих сайтів	04.03.2017	
4	Вибір інструментальних засобів	10.03.2017	
5	Формулювання математичних закономірностей	04.04.2017	
6	Розробка додатку	15.05.2017	
7	Тестування додатку та отримання результатів	26.05.2017	
8	Оформлення дипломної роботи	01.06.2017	
9	Отримання допуску до захисту та подача роботи в ДЕК	09.06.2017	

Студент _____ _____
(підпис) (ініціали, прізвище)

Керівник роботи _____ _____
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломної роботи.

АНОТАЦІЯ

бакалаврської дипломної роботи Пижа Богдана Андрійовича

на тему «Сайт розрахунку цифрових фільтрів. Підсистема розрахунків»

Дипломна робота присвячена задачі розробки та реалізації веб-додатку, що пропонує функції розрахунку цифрових фільтрів он-лайн. Крім того, робота описує існуючі рішення задачі, їх переваги та недоліки, а також математичні основи розрахунку рекурсивних цифрових фільтрів. В ході виконання роботи було створено додаток мовою Java, що надає функції розрахунку цифрових фільтрів у формі веб-сервісу SOAP.

Загальний обсяг роботи: 102 сторінка, 6 рисунків, 9 таблиць, 7 посилань, 1 додаток на 21 сторінці.

Ключові слова: цифровий фільтр, Java, SOAP, веб-сервіс, веб-додаток, сайт, рекурсивний цифровий фільтр.

АННОТАЦИЯ

бакалаврской дипломной работы Пыжа Богдана Андреевича

на тему «Сайт расчёта цифровых фильтров. Подсистема расчётов»

Дипломная работа посвященя задаче разработки и реализации веб-приложения, выполняющего функции расчёта цифровых фильтров он-лайн. Кроме того, работа описывает существующие решения задачи, их преимущества и недостатки, а также математические основы расчёта рекурсивных цифровых фильтров. В ходе выполнения работы было создано приложения на языке Java, предоставляющее функции расчёта цифровых фильтров в форме веб-сервиса SOAP.

Общий объём работы: 102 страница, 6 рисунков, 9 таблиц, 7 библиографических наименований, 1 дополнение объёмом 21 страница.

Ключевые слова: цифровой фильтр, Java, SOAP, веб-сервис, веб-приложение, сайт, рекурсивный цифровой фильтр.

Ключові слова: цифровий фільтр, Java, SOAP, веб-сервіс, веб-додаток, сайт, рекурсивний цифровий фільтр.

ANNOTATION
on Bohdan Pyzh bachelor's degree
thesis: "Digital filter calculation website. Calculations subsystem"

Thesis is devoted to a problem of development and implementation of web-application that provides digital filter calculation functions on-line. Furthermore, thesis describes existing problem solutions, their advantages and disadvantages, and mathematical basis of FIR filters calculation. A Java application had been created in the course of a study providing digital filter calculation functions in a form of SOAP web-service.

Bachelor's work size: 102p., 6 pic., 9 tables, 7 references, 1 appendix for 21p.

Keywords: digital filter, Java, SOAP, web-service, web-application, website, IIR filter.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЦФ – цифровий фільтр

FIR – finite impulse response

IIR – infinite impulse response

АЧХ – амплітудно-частотна характеристика

ЛАЧХ – логарифмічна амплітудно-частотна характеристика

ФЧХ – фазо-частотна характеристика

ІХ – імпульсна характеристика

ФНЧ – фільтр низьких частот

ФВЧ – фільтр високих частот

СФ – смуговий фільтр

РФ – режекторний фільтр

ВСТУП

Сучасний період розвитку науки і техніки характеризується високою розповсюдженістю різноманітних алгоритмів обробки даних будь-яких форматів: зображень, звуків, відео та безлічі інших. Одна з найважливіших різновидів обробки даних є їх фільтрація, тобто відділення частини даних за певними критеріями. Базовим поняттям фільтрації даних в цифровому форматі є цифровий фільтр.

Цифровий фільтр – пристрій, що обробляє цифрові сигнали з метою виділення та/або затримання певних частотних компонент цього сигналу.

Актуальність.

Цифрові фільтри застосовуються будь-де, де застосовуються цифрові сигнали, а отже, є нерозривними з інформаційними технологіями та технологіями зв’язку. Найпростіший приклад застосування цифрового фільтру – придушення шуму шляхом відсічення від спектру сигналу певного діапазону частот, про які відомо, що вони не несуть корисної інформації. Іншим прикладом може бути виділення в сигналі періодичних компонентів з метою дослідити їх властивості.

Мета.

Метою даної роботи є розробка та реалізація веб-додатку, що дозволяє розраховувати параметри та характеристики цифрових фільтрів, не потребуючи встановлення громіздких програм та глибоких технічних знань від користувача. Дана робота зосереджується на рекурсивних цифрових фільтрах як на більш розповсюдженому та менш представленому серед аналогічних додатків типові фільтрів.

Основні завдання.

До основних завдань, що виносяться на дипломну роботу, відносяться:

- Дослідження існуючих сайтів, що пропонують функції розрахунку цифрових фільтрів;
- З’ясування основних вимог до функціоналу та інтерфейсу подібних сайтів;
- Вивчення підходів до побудови цифрових фільтрів з математичної точки зору;
- Розробка та реалізація додатку, що задовольняє поставлені вимоги;
- Аналіз результатів роботи додатку, визначення переваг та недоліків, а також шляхів подальшої розробки.

Об’єкт дослідження – сайт розрахунку цифрових фільтрів.

Предмет дослідження – підсистема розрахунків сайту розрахунку цифрових фільтрів.

1 ДОСЛІДЖЕННЯ ЗАСОБІВ ПРОЕКТУВАННЯ ЦИФРОВИХ БІХ ФІЛЬТРІВ

1.1 Поняття цифрового сигналу та цифрового фільтра

Сигналом називають певний код (символ, знак), що є носієм інформації та використовується для передачі повідомлень у системі зв’язку. Сигналом може бути будь-який фізичний процес, параметри якого змінюються у відповідності з повідомленням, що передається.

Поняття «сигнал» дозволяє абстрагуватися від конкретних фізичних величин, наприклад, струму, напруги, акустичної хвилі та розглядати поза фізичним контекстом явища, що пов’язані з кодуванням інформації та вилученням її з сигналу, що може бути спотворений шумами. Зазвичай сигнал представляється у вигляді функції від часу (неперервної або дискретної).

Цифровий сигнал – це сигнал, що можна представити у вигляді послідовності дискретних (цифрових) значень.

В електроніці для виділення бажаних компонентів спектру сигналу та/або затримання небажаних компонентів використовують пристрої, що називаються фільтрами. Відповідно, для обробки цифрових сигналів використовують цифрові фільтри.

Найбільш розповсюдженими нині є лінійні стаціонарні фільтри через простоту їх поведінки та математичного опису. Основними характеристиками цифрового фільтра є:

- імпульсна характеристика фільтра – реакція фільтра на одиничний імпульс;
- частотні характеристики фільтра;
- передаточна функція фільтра, що описує реакцію фільтра на довільний вхідний сигнал.

Проектування цифрового фільтра означає розрахунок передаточної функції фільтра, характеристики якого відповідають заданим умовам.

1.2 Область застосування цифрових фільтрів

Цифрові фільтри застосовуються в будь-якій області науки та техніки, де необхідно виконувати обробку сигналів, зокрема:

- спектральний аналіз;
- обробка зображень;
- обробка відео;
- обробка звука тощо.

Фільтрація даних може виконуватися з метою виділення періодичних компонент та дослідження їх властивостей, відкидання шумів або згладжування вибірки в цілому.

1.3 Види цифрових фільтрів

Цифрові фільтри поділяються на 2 види:

- Фільтри зі скінченною імпульсною характеристикою (FIR, нерекурсивні фільтри) – фільтри, імпульсна характеристика яких обмежена в часі, тобто, починаючи з певного моменту, вона точно дорівнює нулю. Знаменник передаточної функції такого фільтра є константою. Таким чином, фільтр не має зворотного зв’язку.
- Фільтри з нескінченною імпульсною характеристикою (IIR, рекурсивні фільтри) – фільтри, що використовують один або декілька своїх виходів у якості входів, тобто мають зворотний зв’язок. Імпульсна характеристика таких фільтрів нескінченна в часі, а передаточна функція має дробово-раціональний вигляд. На відміну від FIR, такі фільтри не завжди є стійкими.

1.4 Засоби розрахунку (проектування)цифрових фільтрів

Алгоритм цифрової фільтрації ґрунтується на розв’язанні різницевого рівняння з постійними коефіцієнтами [1]:

$$\sum_{i=0}^{R} b_i y(n-i) = \sum_{i=0}^{P} a_i (x-i) \tag{1.1}$$

де a_i, b_i – постійні коефіцієнти;

x, y – вихідні та результуючі дані для поточного і попередніх дискретних значень часу;

R – порядок різницевого рівняння;

P – кількість відліків вхідних даних;

n – поточний номер відліку.

Для реалізації алгоритм зручніше представити у вигляді розв’язку відносно поточного значення вихідного сигналу:

$$y(n) = \frac{1}{b_0}\left(\sum_{i=0}^{P} a_i (x-i) - \sum_{i=1}^{R} b_i y(n-i)\right)$$

або, позначивши $A_i = \frac{a_i}{b_0}, B_i = \frac{b_i}{b_0}$, запишемо розв’язок (1.1) у вигляді

$$y(n) = \sum_{i=0}^{P} A_i (x-i) - \sum_{i=1}^{R} B_i y(n-i) \tag{1.2}$$

Характеристики фільтра можуть бути обчислені на основі z-перетворення виразу (1.2), що має вигляд [1]

$$Y(z) = X(z)\sum_{i=0}^{P} A_i z^{-i} - Y(z)\sum_{i=0}^{R} B_i z^{-i}$$

Розв’язуючи відносно $Y(z)$, отримуємо

$$Y(z) = X(z)\frac{\sum_{i=0}^{P} A_i z^{-i}}{1 + \sum_{i=0}^{R} B_i z^{-i}}$$

Передаточна функція фільтра $H(z) = \frac{Y(z)}{X(z)}$ має наступний вигляд:

$$H(z) = \frac{\sum_{i=0}^{P} A_i z^{-i}}{1 + \sum_{i=0}^{R} B_i z^{-i}} \quad (1.3)$$

Взагалі при проектуванні ЦФ може бути поставлена задача апроксимації заданих частотних або часових характеристик. На практиці частіше задаються вимоги до частотних характеристик фільтра.

Для розрахунку параметрів ЦФ за заданими характеристиками було розроблено програмні засоби, серед прикладів яких можна назвати програмне середовище Matlab. Однак більшість подібних програм призначена для виконання багаторазових розрахунків фільтрів високих порядків, а також виконання супровідних обчислень, що підходить для розробки складних систем та пристроїв, але є надлишковим для задач, що потребують розрахунку одного чи декількох фільтрів нижчих порядків. Крім того, подібне програмне забезпечення потребує потужного апаратного забезпечення для коректної роботи, а також є занадто дорогим, щоб встановлювати для вирішення однієї задачі.

Альтернативою є онлайн-додатки, що дозволяють розраховувати цифрові та аналогові фільтри.

1.5 Приклади існуючих сайтів розрахунку цифрових фільтрів

Нижченаведені сайти було проаналізовано з точки зору наступних критеріїв:

- Доступні для побудови типи фільтрів;
- Доступні функції (зокрема, безкоштовні функції);
- Швидкість виконання розрахунків;
- Зручність використання інтерфейсу.

1.5.1 MicroModeler DSP [2]

Сайт дозволяє будувати наступні типи фільтрів:

- Рекурсивні:
  - Баттерворта;
  - Чебишова;
  - зворотній Чебишова;
  - Бесселя;
  - еліптичний (Кауера).
- Нерекурсивні:
  - гребінчастий;
  - рухомого середнього;
  - Найквіста (англ. Nyquist filter, Lth band filter);
  - піднятого косинуса (англ. raised cosine filter).
- Фільтри з довільною формою частотних характеристик;
- Фільтри з довільними полюсами та нулями;
- Диференціатори;
- Перетворювач Гільберта;
- Багатофазний фільтр (англ. polyphase filter).

Доступні функції:

- Виведення АЧХ, ФЧХ (в тому числі у логарифмічному масштабі), групової затримки, імпульсної та перехідної характеристик у вигляді графіків з можливістю змінювати вимоги до характеристик фільтра шляхом перетягування контрольних ліній на графіку;
- Виведення нулів та полюсів на одиничному колі з можливістю перетягувати їх, змінюючи значення;
- Тривимірний графік комплексної передаточної функції;
- Реалізація фільтра у вигляді каскаду (в безкоштовній версії обмежено 21 ланкою для нерекурсивних фільтрів та 4 порядком для рекурсивних фільтрів);

- Генерація програмного коду мовою С, що реалізує створений фільтр (в безкоштовній версії обмежено 21 ланкою для нерекурсивних фільтрів та 4 порядком для рекурсивних фільтрів);

Швидкість роботи – висока. Перезавантажувати сторінку після зміни параметрів не потрібно, розрахунки виконуються автоматично після внесення будь-яких змін у параметри фільтра.

Особливості інтерфейсу користувача:

- Мова – англійська;
- Доступ до функцій у вигляді меню з кнопками та вкладками вгорі сторінки (аналогічно до стрічки Microsoft Office), але для виконання дії необхідно не просто натиснути кнопку, а перетягнути її на будь-яке вікно фільтра, що не є інтуїтивно зрозумілим;
- Частоти задаються у вигляді безрозмірних цифрових частот, а нерівномірності в смугах пропускання та придушення задаються у вигляді безрозмірного коефіцієнту посилення або в децибелах за бажанням користувача;
- Вікна з графіками можна розгортати на всю сторінку за допомогою кнопки в кутку (кнопка не підписана та не має позначень);
- Графіки можна рухати, перетягуючи тло кнопкою миші та змінювати масштаб за допомогою колеса;
- Водночас можна отримати до 2 графіків у різних вікнах, у одному вікні можна накладати графіки будь-яких характеристик (включаючи нулі та полюси) замість перемикання між ними (тобто, щоб замість АЧХ вивести ФЧХ, необхідно вимкнути перегляд АЧХ і ввімкнути перегляд ФЧХ).

Підсумок: широка функціональність, хоча й обмежена в безкоштовній версії, висока швидкість роботи, але захаращений та не завжди інтуїтивно зрозумілий інтерфейс, виконаний лише англійською мовою.

1.5.2 TFilter [3]

Сайт дозволяє будувати нерекурсивні фільтри з довільною кількістю, розмірами та коефіцієнтами посилення смуг пропускання та придушення, а також із заданою кількістю ланок (або з мінімально можливою, за бажанням користувача). Відповідно до інформації на сайті, система знаходиться в бета-версії, ведеться розробка TFilter2.

Доступні функції:

- Побудова АЧХ у вигляді графіку;
- Виведення коефіцієнтів ланок фільтра;
- Побудова імпульсної характеристики;
- Генерація коду на С.

Швидкість роботи – висока.

Особливості інтерфейсу:

- Мова – англійська;
- Введення параметрів фільтра у вигляді компактної таблиці з інтуїтивно зрозумілим інтерфейсом;
- Перехід між функціями у вигляді вкладок вгорі сторінки;
- Простий, незахаращений, компактний інтерфейс.

Підсумок: функціональність обмежена нерекурсивними фільтрами та декількома характеристиками, інтерфейс лише англомовний, однак компактний та інтуїтивно зрозумілий.

1.5.3 Interactive Digital Filter Design (by Dr. Anthony J. Fisher) [4]

Сайт дозволяє будувати наступні типи фільтрів:

- Нерекурсивні:
    - піднятого косинуса;
    - перетворювач Гільберта;
- Рекурсивні:
    - Баттерворта;

- Чебишова;
- Бесселя;
- резонатор;
- пропорційно-інтегруючий регулятор.

Доступні функції:

- Виведення нулів та полюсів у вигляді комплексних чисел;
- Виведення різницевого рівняння фільтра;
- Генерація коду мовою С;
- Виведення АЧХ (або ЛАЧХ, за бажанням користувача) та ФЧХ у вигляді одного графіку;
- Виведення імпульсної та перехідної характеристик (на окремих графіках);

Швидкість роботи – висока.

Особливості інтерфейсу:

- Мова – англійська;
- Доступ до різних типів фільтрів через традиційні гіперпосилання;
- Виведення всіх характеристик на одній сторінці у фіксованому порядку;
- Відповідно до секції «Нещодавніх змін», востаннє сайт оновлювався в жовтні 1999 року.

Підсумок: непогана функціональність, однак інтерфейс, хоча й інтуїтивно зрозумілий, іноді вимагає виконання надлишкових дій (наприклад, постійно прокручувати сторінку з результатами до АЧХ).

1.5.4 Digital Filters Applet (by Paul Falstad) [5]

Сайт дозволяє будувати наступні типи фільтрів:

- Рекурсивні:
  - Батерворта;
  - Чебишова;

- зворотній Чебишова;
- еліптичний;
- резонатор;
- гребінчастий;
- рухомого середнього;
- фільтр з довільними нулями та полюсами;

- Нерекурсивні низьких та високих частот, смугові та режекторні довільного порядку.

Доступні функції:

- Виведення нулів та полюсів з можливістю перетягувати їх по одиничному колу;
- Виведення АЧХ, ФЧХ, імпульсної та перехідної характеристик.

Швидкість роботи – висока.

Особливості інтерфейсу:

- Мова – англійська;
- Параметри фільтрів задаються у вигляді повзунків, без можливості ввести точне числове значення;
- Функціональність реалізована у вигляді Java-аплету, тобто потребує встановлену Java на комп'ютері клієнту.

Підсумок: велика кількість доступних для побудови фільтрів, однак бідна функціональність та незручний англомовний інтерфейс.

1.5.5 Digital Filter Design (by Dr. A. R. Collins) [6]

Сайт дозволяє будувати нерекурсивні фільтри за допомогою вікна Кайзера-Бесселя.

Доступні функції:

- Виведення коефіцієнтів ланок фільтра;
- Виведення АЧХ;
- Виведення імпульсної характеристики.

Швидкість роботи – висока.

Особливості інтерфейсу:

- Мова – англійська;
- Всі параметри та характеристики знаходяться в одному вікні.

Підсумок: бідна функціональність та англомовний інтерфейс.

1.6 Підсумки огляду

Результати огляду найбільш використовуваних сайтів розрахунку цифрових фільтрів показують, що:

- всі подібні сайти мають лише англомовний інтерфейс, що може бути незручним для іншомовного (зокрема, україномовного) користувача;
- частина сайтів має достатню функціональність для розв’язання нескладних задач;
- інтерфейс більшості сайтів є незручним та/або захараченим;
- сайти мають високу швидкість роботи.

Оскільки сайти, що пропонують побудову нерекурсивних фільтрів, пропонують зручний інтерфейс та непогану функціональність, більш важливою задачею є створення сайту побудови цифрових рекурсивних фільтрів з наступними можливостями:

- Побудова фільтрів низьких та високих частот, смугових та режекторних фільтрів;
- Використання апроксимуючих функцій Батерворта, Чебишова, Кауера;
- Виведення полюсів та нулів, АЧХ, ЛАЧХ, ФЧХ, імпульсної та перехідної характеристик у вигляді графіків;
- Виведення коефіцієнтів передаточної функції.

Сайт повинен мати україномовний, зручний та інтуїтивно зрозумілий інтерфейс.

Крім того, доцільно буде розділити систему на 2 частини: інтерфейс користувача (власне сайт, який бачить користувач) та підсистему розрахунків

(серверний додаток, до якого звертається сайт, щоб виконати побудову фільтрів). Таким чином, всі необхідні розрахунки, включаючи побудову графіків, будуть виконуватися з боку серверу, що зменшить навантаження на комп’ютер користувача та пришвидшить роботу сайту. Ще однією перевагою такого підходу є незалежність інтерфейсу від підсистеми розрахунків, що дозволяє використовувати функції підсистеми розрахунків в ході розробки інших додатків (наприклад, мобільної версії системи) і, навпаки, використовувати інтерфейс сумісно з різними підсистемами розрахунків, що мають різний функціонал, за умови однотипності інтерфейсів цих підсистем.

1.7 Висновки

У даному розділі було проведено аналіз завдання дипломної роботи. Було досліджено поняття цифрового сигналу та цифрового фільтра, а також проведено аналіз існуючих засобів побудови цифрових фільтрів. За результатами аналізу можна зробити висновок, що програмне забезпечення, призначене для розв’язання складних обчислювальних задач, часто виявляється занадто громіздким та дорогим для побудови нескладних фільтрів. Цю проблему можна усунути, якщо використовувати сайти розрахунку цифрових фільтрів, однак існуючі рішення, хоча й пропонують достатню функціональність та швидкість для розв’язання нескладних задач, мають незручний інтерфейс, а також потребують знання англійської мови для використання.

2 АНАЛІЗ ПІДСИСТЕМИ РОЗРАХУНКІВ

2.1 Постановка задачі

На основі вимог до розроблюваного сайту розрахунку цифрових фільтрів можна сформулювати наступні вимоги до підсистеми розрахунків:

- Наявність функцій для розрахунку фільтрів нижніх і верхніх частот, смугових та режекторних фільтрів Батерворта, Чебишова та Кауера;
- Можливість розраховувати фільтри як за відомим порядком, так і за параметрами смуги затримання;
- Швидкість та стабільність роботи на невеликих порядках фільтрів (до 30);
- Наявність функцій для розрахунку нулів та полюсів, передаточної функцій, АЧХ, ЛАЧХ, ФЧХ, імпульсної та перехідної характерик фільтра;
- Кросплатформеність серверного додатку (оскільки невідомо, на яких операційних системах буде працювати додаток після впровадження);
- Забезпечення легкого доступу з боку інтерфейсу користувача незалежно від конкретної архітектури сайту.

2.2 Вибір інструментальних засобів

2.2.1 Метод зв’язку з інтерфейсом користувача

Оскільки необхідно забезпечити доступ з боку інтерфейсу користувача незалежно від доступних розробнику інтерфейсу платформ, бібліотек та фреймворків, доцільно використовувати загальновживані методи, підтримка яких широко розповсюджена. Серед таких можна назвати:

2.2.1.1 Звичайні HTTP-запити

HTTP-запити – найбільш простий та розповсюджений спосіб зв’язку між додатками, що також використовується і в веб-сервісах на різних рівнях;

Переваги: підтримується без встановлення додаткових бібліотек майже всіма платформами розробки веб-додатків; можна передавати дані в різних форматах, наприклад, XML, JSON та інші.

Недоліки: необхідно вручну описувати можливі запити як з боку підсистеми розрахунків, так і з боку інтерфейсу користувача, що ускладнює можливі зміни форматів запитів.

2.2.1.2 Веб-сервіси SOAP

Веб-сервіси SOAP – модель, що використовує XML для пересилання запитів та відповідей, а також для опису функціональності самого веб-сервісу;

Переваги: легко підключитися з боку інтерфейсу користувача, оскільки методи доступу до веб-сервісу детально описуються в WSDL-файлі, що слугує точкою доступу; відносно легко перетворювати XML на об’єкти, описані мовою програмування високого рівня незалежно від конкретної мови; використовує HTTP лише як пасивний засіб передачі даних, без необхідності налаштовувати параметри запитів; потенційно необмежена кількість різних запитів та відповідей для одного сервісу.

Недоліки: не підтримує форматів даних крім XML; великий розмір запитів та відповідей; необхідно детально описувати формати даних та запитів за допомогою XML-файлів.

2.2.1.3 Веб-сервіси REST

Веб-сервіси REST – модель, що активно використовує HTTP-запити різних типів для реалізації механізму веб-сервісу.

Переваги: не потребує детального опису даних та запитів; використовує існуючі інтерфейси HTTP для пересилання даних; відсутні вимоги до типів даних (аналогічно до HTTP); невеликий об’єм запитів та відповідей.

Недоліки: тісно пов’язаний з HTTP, тому має всі недоліки звичайних HTTP запитів; дозволяє визначити не більше 4 різних запитів на один веб-сервіс (відповідно до 4 типів HTTP-запитів).

2.2.1.4 Підсумки

Аналіз різних моделей зв’язку веб-додатків дозволяє зробити висновок про доцільність використання веб-сервісів SOAP, оскільки дана модель максимально полегшує підключення з боку інтерфейсу користувача завдяки детальному опису форматів даних і запитів, а також автоматично виконує всі необхідні HTTP-запити, дозволяючи зосередитись на моделі даних та розрахунках.

2.2.2 Форма реалізації фільтрів

В якості форми реалізації фільтрів доцільно обрати каскадну форму, як найбільш поширену та просту в реалізації. Передаточна функція в каскадній формі представляється у вигляді добутку

$$H(z) = A_0 \prod_{\mu=1}^{M} H_\mu(z), \tag{2.1}$$

де $H_\mu(z)$ – передаточні функції ланок першого або другого порядку:

$$H_\mu(z) = \begin{cases} \dfrac{1 + A_{1\mu} z^{-1}}{1 + B_{1\mu} z^{-1}} \\ \dfrac{1 + A_{1\mu} z^{-1} + A_{2\mu} z^{-2}}{1 + B_{1\mu} z^{-1} + B_{2\mu} z^{-2}}, \end{cases}$$

A_0 – масштабний коефіцієнт (постійний коефіцієнт);

M – кількість ланок.

2.2.3 Методика проектування цифрових фільтрів

Вихідні дані для проектування фільтра визначаються вимогами до його АЧХ, що має містити наступні елементи:

- смуга пропускання – діапазон частот, сигнал в яких майже не змінюється. Визначається межами (в герцах) та максимальним відхиленням ЛАЧХ сигналу від вихідного значення (в децибелах);
- смуга затримання – діапазон частот, сигнал в яких пригнічується до певного рівня. Визначається межами (в герцах) та максимальним

значенням сигналу відносно вихідного значення (в децибелах, звичайно менше 0);

- тип фільтру – характер розміщення смуг пропускання та затримання в межах АЧХ. Розрізняють 4 типи фільтрів:
    - фільтр низьких частот (ФНЧ) має одну смугу пропускання та одну смугу затримання, причому остання містить більш високі частоти, ніж перша;
    - фільтр високих частот (ФВЧ) має одну смугу пропускання та одну смугу затримання, причому остання містить більш низькі частоти, ніж перша;
    - смуговий фільтр (СФ) має одну смугу пропускання, що розміщена між двома смугами затримання;
    - режекторний фільтр (РФ) має одну смугу затримання, що розміщена між двома смугами пропускання;
- тип апроксимуючої функції фільтру, від якого залежить форма сигналу в смугах пропускання та затримання, а також можлива ширина переходу між смугами:
    - фільтри Батерворта мають гладку АЧХ як у смузі пропускання, так і у смузі затримання;
    - фільтри Чебишова мають гладку АЧХ у смузі затримання, але пульсації в смузі пропускання, при цьому смуга переходу вужча, ніж у фільтрів Батерворта;
    - фільтри Кауера (еліптичні фільтри) мають АЧХ з пульсаціями як у смузі пропускання, так і у смузі затримання, при цьому смуга переходу ще вужча, ніж у фільтрів Чебишова;
- порядок фільтру також може бути відомий наперед, в такому випадку для проектування фільтрів Батерворта та Чебишова параметри смуги затримання не потребуються.

Апроксимація частотних характеристик може бути виконана одним з трьох методів:

1) прямою апроксимацією в z-області;
2) за аналоговим прототипом;
3) за допомогою методів оптимізації.

На практиці пряма апроксимація виявляється занадто складною і майже не використовується, тому найбільш розповсюджені апроксимація за аналоговим прототипом і оптимізаційні методи.

Апроксимація характеристик ЦФ за аналоговим прототипом базується на використанні методів розрахунку передаточних функцій аналогових фільтрів і складається з:

а) розрахунку передаточної функції аналогового фільтра-прототипа;
б) перетворення передаточної функції аналогового фільтра в передаточну функцію ЦФ.

Передаточна функція аналогового прототипу розраховується згідно з відомими методиками проектування аналогових фільтрів [8].

Тип апроксимації визначає форму АЧХ фільтра. Апроксимація за Батервортом забезпечує максимально гладку АЧХ як в смузі пропускання, так и в смузі затримання. Апроксимація за Чебишовим дає АЧХ с рівновеликими пульсаціями в смузі пропускання (або затримання) и гладку в смузі затримання (або пропускання), при цьому перехідна смуга вужче, а подавлення в смузі затримання більше, ніж у фільтрів Батерворта. АЧХ фільтрів, отриманих за допомогою еліптичних функцій (апроксимація за Кауером), має пульсації як в смузі пропускання, так і в смузі затримання, а перехідна смуга виходить найвужча.

Для перетворення передаточних функцій аналогових фільтрів- прототипів в передаточні функції цифрових фільтрів можна використовувати наступні методи [1; 8]:

1) метод погодженого z-перетворення, оснований на заміщенні полюсів (нулів) аналогової передаточної функції $H_a(s)$ $s = -p$ полюсами

(нулями) цифрової передаточної функції e^{-pT} (тобто виконується заміна $s + p \to 1 - z^{-1}e^{-pT}$). Цей метод використовується для розрахунку фільтрів високих частот та режекторних фільтрів за прототипом, що має полюси і нулі. Якщо прототип має нулі p такі, що $\mathrm{Im}(p) > \frac{2\pi F_S}{2}$, де F_s – частота дискретизації, то характеристики ЦФ будуть спотворені внаслідок накладення;

2) метод інваріантного перетворення імпульсної характеристики, оснований на використанні відліків ІХ аналогового прототипу, взятих з частотою дискретизації, в якості відліків ІХ цифрового фільтра, тобто ІХ цифрового фільтра отримуються із ІХ аналогового фільтра згідно з виразом

$$h(n) = h_a(nT_s),$$

де T_s – період дискретизації.

Аналогова передаточна функція перетворюється в цифрову шляхом заміни:

$$\frac{1}{s - p_i} \to \frac{1}{1 - z^{-1}e^{p_i T_s}},$$

де p_i – полюси аналогової передаточної функції. Метод може використовуватись для розрахунку фільрів низьких частот і смугових фільтрів, для яких $|H_a(j\omega)| = 0$ при $|\omega| \geq \frac{\pi}{T_s}$. Цей метод забезпечує лінійне перетворення аналогових частот в цифрові;

3) метод білінійного z-перетворення, яке має вигляд

$$s = \frac{2}{T_s}\frac{z - 1}{z + 1} \qquad (2.2)$$

Цей метод зв’язаний з методами чисельного інтегрування [1] .В цьому випадку уявна вісь комплексної *s*-площини відображається на одиничне коло *z*-площини однозначно, завдяки чому ефекти накладання в частотній області відсутні і значення АЧХ фільтра на різних частотах не спотворюються. Співвідношення аналогових і цифрових частот в цьому випадку має вигляд

$$2\pi f T_s = 2\ \mathrm{arctg}\frac{\omega T_s}{2},$$

де f – цифрова частота;

ω – аналогова частота.

Оскільки це співвідношення нелінійне, білінійне z-перетворення спотворює співвідношення між смугами пропускання і затримання, але нерівномірність АЧХ в цих смугах (A_p і A_a) залишається незмінною. Нелінійність перетворення частот для ЦФ низьких та високих частот, смугових та режекторних фільтрів легко компенсується за допомогою корегуючих множників. Завдяки цьому білінійне z-перетворення є єдиним методом, який придатний для проектування всіх 4-х згаданих типів ЦФ. Крім того, позитивним моментом є те, що при незначній модифікації співвідношення (2.2) ми можемо використовувати нормовані аналогові фільтри-прототипи (для яких границя полоси пропускання, або частота зрізу, ω_p=1).

Апроксимація на основі оптимізаційних методів використовується при проектуванні ЦФ з нестандартними АЧХ. Це можуть бути цифрові диференціатори, інтегратори і компенсатори, багатосмугові ЦФ та ін., апроксимація яких аналітичними методами неможлива. Оптимізаційні методи використовуються для мінімізації відхилення АЧХ ЦФ, що проектується, від заданої форми. При цьому найчастіше використовують методи нелінійного програмування.

Таким чином, в межах даної роботи є доцільним використовувати апроксимацію за аналоговим прототипом методом білінійного z-перетворення.

2.2.4 Мова програмування

Для забезпечення однакової роботи серверного додатку на будь-якій платформі необхідно використовувати кросплатформену мову програмування високого рівня. Найбільш розповсюдженою та потужною такою мовою є Java, що виконується за допомогою віртуальної машини і тому створені додатки

будуть працювати однаково незалежно від операційної системи, під керуванням якої вони працюють.

2.2.5 Використовувані бібліотеки та фреймворки

Веб-сервіс можна побудувати за допомогою вбудованих засобів Java (а саме JAX-WS) або з використанням фреймворку Spring. Веб-сервіс SOAP потребує створення WSDL файлу з детальним описом запитів та даних, що доступні користувачеві веб-сервісу, однак, на відміну від JAX-WS, Spring дозволяє автоматично створювати файл опису веб-сервісу на основі XSD файлів (XML схем), що описують лише формати даних та класів Java, що містять налаштування веб-сервісу (наприклад, URL). Крім того, Spring надає можливості з тестування додатків та розробки власне сайту (якщо така необхідність виникне), а також замість WAR-файлу, що необхідно розгортати на налаштованому веб-сервері, можна зібрати додаток у вигляді звичайного JAR-файлу, який достатньо запустити на будь-якому комп'ютері, щоб користуватися функціями веб-сервісу через IP цього комп'ютера.

Таким чином, розробку веб-сервісу доцільніше виконувати за допомогою фреймворку Spring.

2.2.6 Засоби автоматизації збірки

Оскільки Spring є, по суті, збіркою фреймворків, кожен з яких надає різну функціональність, доцільним є замість того, щоб завантажувати бібліотеки Spring та підключати їх до проекту вручну, використовувати один з існуючих засобів автоматизації збірки. Основними інструментами такого типу є Apache Ant, Maven та Gradle.

Apache Ant використовує XML для опису проекту та залежностей в імперативному вигляді (тобто, у вигляді списку команд, які необхідно виконати для збірки). Однак XML як ієрархічна мова не є придатною для імперативних описів, тому даний засіб підходить для випадків, коли процес збірки є складним та розгалуженим, а список залежностей, навпаки, просто описати.

Maven також використовує XML для опису залежностей проекту, але в декларативному стилі, описуючи модулі, що необхідно завантажити та підключити до проекту. Власне збірка виконується у вигляді послідовності закріплених цілей (goals), кожна з яких може бути описана декларативно. Maven добре підходить для вирішення складних залежностей, але не для розгалужених послідовностей збірки.

Gradle поєднує потужність та гнучкість Ant з декларативними описами Maven, однак замість XML використовує окрему JVM-мову Groovy.

Оскільки для побудови веб-сервісу не потрібні розгалужені імперативні описи, але необхідно забезпечити правильне підключення бібліотек, доцільним буде використовувати Maven в якості засобу автоматизації збірки.

2.3 Висновки

У даному розділі було проведено аналіз задачі розробки підсистеми розрахунків для сайту розрахунку цифрових фільтрів. Було сформульовано вимоги до серверного додатку, а також проведено та обґрунтовано вибір інструментальних засобів для виконання завдання роботи. Було обрано наступні засоби:

- метод зв’язку з інтерфейсом – веб-сервіс SOAP;
- форма реалізації фільтра – каскадна;
- метод побудови цифрових фільтрів – апроксимація за аналоговим прототипом методом білінійного *z*-перетворення.
- мова програмування – Java;
- використовуваний фреймворк – Spring;
- засіб автоматизації збірки – Maven.

3 ДОСЛІДЖЕННЯ МАТЕМАТИЧНИХ ОСНОВ ПОБУДОВИ ЦИФРОВИХ ФІЛЬТРІВ

3.1 Основні поняття та позначення

Частотні характеристики фільтра є функціями виразу $fT_s = \frac{f}{F_s}$, де T_s – період дискретизації, F_s – частота дискретизації, f – поточна частота, що є безрозмірним і називається цифровою частотою. Далі використовуються цифрові частоти, якщо не вказано інше.

АЧХ задається на інтервалі цифрових частот $0 \le f \le 0.5$, тому що частотні характеристики ЦФ симетричні відносно частоти $f = 0.5$. Введемо позначення для параметрів АЧХ в залежності від типів фільтра відносно смуг пропускання та затримання:

Таблиця 3.1. Межі смуг пропускання та затримання

Тип фільтра	**Смуга пропускання**	**Смуга затримання**
ФНЧ	$0 \le f \le f_p$	$f_a \le f \le 0.5$
ФВЧ	$f_p \le f \le 0.5$	$0 \le f \le f_a$
СФ	$f_{p1} \le f \le f_{p2}$	$\begin{cases} 0 \le f \le f_{a1} \\ f_{a2} \le f \le 0.5 \end{cases}$
РФ	$\begin{cases} 0 \le f \le f_{p1} \\ f_{a2} \le f \le 0.5 \end{cases}$	$f_{a1} \le f \le f_{a2}$

В межах смуги пропускання значення ЛАЧХ фільтра не повинно відхилятись від нормованого значення (тобто 0 дБ) більше, ніж на A_p дБ, а в смузі затримання її значення не повинно перевищувати величину A_a дБ.

Апроксимація частотних характеристик фільтра базується на квадраті модуля АЧХ, який представлений у вигляді

$$|H(j\omega)|^2 = \frac{1}{1 + \varepsilon^2 A_N^2\left(\frac{\omega}{\omega_p}\right)},$$

де $A_N\left(\frac{\omega}{\omega_p}\right)$ – апроксимуюча функція порядку N;

ε – затухання АЧХ на межі смуги пропускання ω_p;

ω_p – гранична частота смуги пропускання, на якій квадрат модуля АЧХ зменшується до $\frac{1}{1+\varepsilon^2}$.

Зазвичай спочатку розраховують нормовану АЧХ НЧ з ω_p=1, а потім виконують її частотне перетворення. В цьому випадку

$$|H(j\omega)|^2 = \frac{1}{1 + \varepsilon^2 A_N^2(\omega)}, \qquad (3.1)$$

Знаменник (3.1) характеризує втрати, або затухання, АЧХ і є позитивною функцією аргумента ω:

$$L(\omega) = 1 + \varepsilon^2 A_N^2(\omega) \qquad (3.2)$$

При $\omega=\omega_a$ коефіцієнт пригнічення дорівнює A^2.

В випадку проектування цифрових фільтрів ω_a аналогового прототипа обчислюється за формулами, які визначаються методом перетворення аналогового прототипу в цифровий фільтр. Таблиця 3.2 вміщує формули для білінійного перетворення.

Логарифмічну АЧХ фільтра можна обчислити через апроксимуючу функцію як

$$10\lg|H(j\omega)|^2 = -10\lg\left(1 + \varepsilon^2 A_N^2(\omega)\right)$$

або

$$20\lg|H(j\omega)| = -10\lg L(\omega)$$

Параметри апроксимуючої функції розраховуються на основі заданих вимог до АЧХ цифрового фільтра і в залежності від способу перетворення аналогового фільтра в цифровий.

Таблиця 3.2. Формули перетворення вимог до цифрового фільтра на вимоги до нормованого фільтра-прототипу низьких частот

Тип фільтра	Вимоги до прототипу
ФНЧ	$\omega_a = \frac{\operatorname{tg}\frac{2\pi f_a}{2}}{\operatorname{tg}\frac{2\pi f_p}{2}}$
ФВЧ	$\omega_a = \frac{\operatorname{tg}\frac{2\pi f_p}{2}}{\operatorname{tg}\frac{2\pi f_a}{2}}$
СФ	$-\omega_a \geq \frac{\alpha(\beta - \cos 2\pi f_{a1})}{\sin 2\pi f_{a1}}, \omega_a \leq \frac{\alpha(\beta - \cos 2\pi f_{a2})}{\sin 2\pi f_{a2}},$ $\alpha = \operatorname{ctg}\frac{2\pi(f_{p2} - f_{p1})}{2}, \beta = \frac{\sin\left(2\pi(f_{p1} + f_{p2})\right)}{\sin 2\pi f_{p1} + \sin 2\pi f_{p2}}$
РФ	$-\omega_a \geq \frac{\alpha \sin 2\pi f_{a1}}{(\beta - \cos 2\pi f_{a1})}, \omega_a \leq \frac{\alpha \sin 2\pi f_{a2}}{(\beta - \cos 2\pi f_{a2})},$ $\alpha = \frac{\cos 2\pi f_{p1} - \cos 2\pi f_{p2}}{\sin 2\pi f_{p1} + \sin 2\pi f_{p2}}, \beta = \frac{\sin\left(2\pi(f_{p1} + f_{p2})\right)}{\sin 2\pi f_{p1} + \sin 2\pi f_{p2}}$

При білінійному перетворенні нерівномірності АЧХ аналогового фільтра зберігаються, таким чином,

$$20 \lg |H(j\omega_p)| \geq -A_p;$$

$$20 \lg |H(j\omega_a)| \leq -A_a$$

або

$$\left|H\left(j\omega_p\right)\right|^2 = 10^{-\frac{A_p}{10}} = \frac{1}{1+\varepsilon^2};$$

$$|H(j\omega_a)|^2 = 10^{-\frac{A_a}{10}} = \frac{1}{A^2}.$$

Звідси витікає

$$\varepsilon = \sqrt{10^{\frac{A_p}{10}} - 1},$$
$$A = \sqrt{10^{\frac{A_a}{10}}} \tag{3.3}$$

З другого боку, при білінійному перетворенні частоти пов'язані між собою однозначним, але нелінійним співвідношенням

$$\frac{\omega}{\omega_p} = \frac{\mathrm{tg}(\pi f T_s)}{\mathrm{tg}(\pi f_p T_s)}$$

Це дає можливість розрахувати ω_a для нормованого аналогового прототипу.

3.2 Види апроксимуючих функцій

Для апроксимації Батерворта функція $A_N(\omega)$ є поліномом *N*-го ступеня відносно змінної ω:

$$A_N(\omega) = \omega^N \tag{3.4}$$

Коли ω збільшується в межах $0 \le \omega \le \infty$, $A_N(\omega)$ монотонно зростає, а $|H(j\omega)|^2$ монотонно зменшується. Апроксимація за Батервортом забезпечує максимально гладку АЧХ як в смузі пропускання, так і в смузі ослаблення.

Апроксимація за Чебишовим в якості функції $A_N(\omega)$ використовує поліном Чебишова *N*-го ступеня:

$$A_N(\omega) = T_N(\omega) = \begin{cases} \cos(N\, cos^{-1}\, \omega), |\omega| \le 1, \\ \mathrm{ch}(N\, \mathrm{ch}^{-1}\omega), |\omega| \ge 1 \end{cases} \tag{3.5}$$

Результатом апроксимації за Чебишовим є АЧХ з рівновеликими пульсаціями в смузі пропускання, яка є гладкою в смузі затримання. Перехідна смуга при апроксимації за Чебишовим вужча, ніж при апроксимації за Батервортом.

При апроксимації за Кауером функція $A_N(\omega)$ визначається через еліптичну функцію Якобі:

$$A_N(\omega) = U_N(\omega) = \begin{cases} \operatorname{sn}\left(N\frac{K_1}{K}\operatorname{sn}^{-1}(\omega,k),k_1\right) \text{ для непарних } N \\ \operatorname{sn}\left(K_1 + N\frac{K_1}{K}\operatorname{sn}^{-1}(\omega,k),k_1\right) \text{ для парних } N \end{cases} \quad (3.6)$$

Апроксимація за Кауером дає АЧХ з пульсаціями як в смузі пропускання, так і в смузі затримання.

3.3 Розрахунок порядку апроксимуючої функції фільтра-прототипу

Порядок апроксимуючої функції визначається, виходячи із умови забезпечення необхідного затухання на частоті ω_a:

$$L(\omega_a) = 1 + \varepsilon^2 A_N^2(\omega_a) = A^2$$

або

$$A_N(\omega_a) = \sqrt{\frac{A^2 - 1}{\varepsilon^2}}$$

З урахуванням співвідношення між A, ε і A_p, A_a, отримуємо:

$$A_N(\omega_a) = \sqrt{\frac{10^{\frac{A_a}{10}} - 1}{10^{\frac{A_p}{10}} - 1}}$$

Позначивши $\frac{10^{\frac{A_a}{10}}-1}{10^{\frac{A_p}{10}}-1} = D$, запишемо співвідношення для визначення порядка у вигляді

$$A_N(\omega_a) = \sqrt{D}. \quad (3.7)$$

Отримане в ході розрахунків значення N в більшості випадків не буде цілим, тому його необхідно округлити до найближчого більшого цілого.

Для апроксимації Батерворта $A_N(\omega_a) = \omega_a^N = \sqrt{D}$, тому

$$N = \frac{\ln\sqrt{D}}{\ln\omega_a} = \frac{\ln D}{2\ln\omega_a} \quad (3.8)$$

Для апроксимації Чебишова

$$A_N(\omega) = T_N(\omega) = \mathrm{ch}(N\ \mathrm{ch}^{-1}\omega_a) = \sqrt{D} \Rightarrow N = \frac{\mathrm{ch}^{-1}\sqrt{D}}{\mathrm{ch}^{-1}\omega_a}.$$

Враховуючи, що $\mathrm{ch}^{-1}x = \ln\left(x + \sqrt{x^2-1}\right)$, вираз для N можна записати у вигляді

$$N = \frac{\ln\left(\sqrt{D} + \sqrt{D-1}\right)}{\ln\left(\omega_a + \sqrt{\omega_a^2 - 1}\right)}.$$

В більшості випадків значення D набагато перевищує 1, тому можна прийняти $\ln\left(\sqrt{D} + \sqrt{D-1}\right) \approx \ln 2\sqrt{D}$, таким чином,

$$N = \frac{\ln 2\sqrt{D}}{\ln\left(\omega_a + \sqrt{\omega_a^2 - 1}\right)} = \frac{\ln 4D}{2\ln\left(\omega_a + \sqrt{\omega_a^2 - 1}\right)} \qquad (3.9)$$

Порядок апроксимуючої функції Кауера визначається із співвідношення $N\frac{K'}{K} = \frac{K_1'}{K_1}$, де $K = K(k); K' = K(k') = K\left(\sqrt{1-k^2}\right); K_1 = K(k_1); K_1' = K(k_1') = K\left(\sqrt{1-k_1^2}\right)$.

Скористаємося співвідношенням $\mathrm{sn}(K_1, k_1) = 1$. Тоді

$$\frac{1}{\sqrt{k_1}} \frac{2q_1^{\frac{1}{4}} \sum_{m=0}^{\infty} (-1)^m q_1^{m(m+1)} \sin\left((2m+1)\frac{\pi K_1}{2K}\right)}{1 + 2\sum_{m=1}^{\infty} (-1)^m q_1^{m^2} \cos\left(2m\frac{\pi K_1}{2K}\right)} = 1,$$

де $q_1 = e^{-\pi\frac{K_1'}{K_1}}$ або $K_1 = 4\sqrt{q_1}\left(\frac{1+q_1^2+q_1^6+\cdots}{1+2q_1+2q_1^4+\cdots}\right)^2$

В більшості випадків величина $k_1 = \sqrt{\frac{10^{\frac{A_p}{10}}-1}{10^{\frac{A_a}{10}}-1}} = \frac{1}{\sqrt{D}}$ досить мала, тому можна прийняти $k_1 \approx 0$, а $k_1' = \sqrt{1-k_1^2} \approx 1, \frac{K_1'}{K_1} \gg 1$ і $q_1 \ll 1$.

Тоді $K_1^2 = 16q_1 = 16e^{-\pi\frac{K_1'}{K_1}} = 16e^{-\pi N\frac{K'}{K}} = 16q^N$. Звідси

$$N = \frac{\ln 16D}{\ln\frac{1}{q}} \tag{3.10}$$

Параметр q можна обчислити зі співвідношення $\mathrm{dn}(0,k) = 1$:

$$\mathrm{dn}(0,k) = \sqrt{k'}\frac{\theta_3\left(\frac{0}{2K},q\right)}{\theta_0\left(\frac{0}{2K},q\right)} = 1$$

або

$$\sqrt{k'} = \frac{1-2q+2q^4-2q^9+\cdots}{1+2q+2q^4+2q^9+\cdots}$$

Наближено можна прийняти $q_0 = \frac{1}{2}\frac{1-\sqrt{k'}}{1+\sqrt{k'}}, q = q_0 + 2q_0^5 + 15q_0^9 + 150q_0^{13}$.

3.4 Розрахунок нулів та полюсів апроксимуючої функції фільтра-прототипу

АЧХ фільтру-прототипу можна представити у вигляді $|H(j\omega)|^2 \overset{\omega=\frac{s}{j}}{\longrightarrow} H(s)H(-s)$. Нулі та полюси цієї функції визначаються на основі коренів функції $L(\omega^2) \overset{\omega=\frac{s}{j}}{\longrightarrow} L(-s^2)$.

Позначимо також $M = \begin{cases} \frac{N+1}{2} \text{ для непарних } N \\ \frac{N}{2} \text{ для парних } N \end{cases}$ – загальна кількість ланцюгів другого порядку.

Передаточна функція в термінах нулів і полюсів має вигляд

$$H(s) = a_0\prod_{\mu=1}^{M}\frac{s-z_\mu}{s-p_\mu},$$

Дійсні полюси $s = s_\mu$ породжують ланцюги першого порядку виду $\frac{1}{s-s_\mu}$.

Комплексні спряжені пари полюсів $s = \sigma_\mu \pm j\Omega_\mu$ породжують ланцюги другого порядку виду $\frac{1}{(s-\sigma_\mu-j\Omega_\mu)(s-\sigma_\mu+j\Omega_\mu)}$.

Масштабний коефіцієнт a_0 обирається таким чином, щоб максимальне значення передаточної функції дорівнювало одиниці.

Для апроксимації Батерворта передаточна функція має лише полюси, тому що функція $L(-s^2) = 1 + \varepsilon^2(-s^2)^N$ має тільки нулі

$$s_i = \begin{cases} \varepsilon^{-\frac{1}{N}} e^{j(2i-1)\frac{\pi}{2N}} \text{ для парних } N \\ \varepsilon^{-\frac{1}{N}} e^{j(i-1)\frac{\pi}{N}} \text{ для непарних } N \end{cases},$$

які розташовані рівномірно по колу з радіусом $\sigma_0 = \varepsilon^{-\frac{1}{N}} = \left(10^{\frac{A_p}{10}} - 1\right)^{-\frac{1}{2N}}$ симетрично відносно дійсної осі з кутовою відстанню $\frac{\pi}{N}$.

Звідси вирази для обчислення комплексних спряжених пар полюсів передаточної функції:

$$\sigma_\mu \pm j\Omega_\mu = \begin{cases} \sigma_0 \left(\sin\frac{2\mu+1}{2N}\pi \pm j\cos\frac{2\mu+1}{2N}\pi\right) \text{ для непарних } N \\ \sigma_0 \left(\sin\frac{2\mu-1}{2N}\pi \pm j\cos\frac{2\mu-1}{2N}\pi\right) \text{ для парних } N \end{cases} \quad (3.11)$$

де μ відповідає номеру ланцюга другого порядку від 1 до M.

Якщо N непарне, то один з полюсів є дійсним:

$$s_0 = -\sigma_0 \quad (3.12)$$

З умови $H(s=0) = 1$ знаходимо масштабний коефіцієнт:

$$a_0 = \begin{cases} \sigma_0 \prod_{\mu=1}^{M} (\sigma_\mu^2 + \Omega_\mu^2) \text{ для непарних } N \\ \prod_{\mu=1}^{M} (\sigma_\mu^2 + \Omega_\mu^2) \text{ для парних } N \end{cases} \quad (3.13)$$

Для апроксимації Чебишова передаточна функція має полюси, що є коренями рівняння $1 + \varepsilon^2 \left(\text{ch}\, N\, \text{ch}^{-1}\frac{s}{j}\right)^2 = 0$.

Оскільки $s = \sigma \pm j\Omega$, то $\text{ch}^{-1}\frac{s}{j} = \text{ch}^{-1} -j\sigma + \Omega = u + jv$ і $\text{ch}\big(N(u+jv)\big) = \pm\frac{j}{\varepsilon}$.

З першого рівняння знаходимо:

$$-j\sigma + \Omega = \operatorname{ch}(u + jv) = \operatorname{ch} u \cos v + j \operatorname{sh} u \sin v$$

З другого рівняння знаходимо:

$$\operatorname{ch}(Nu + jNv) = \operatorname{ch} Nu \cos Nv + j \operatorname{sh} Nu \sin Nv = \pm\frac{j}{\varepsilon}$$

або

$$\begin{cases} \operatorname{ch} Nu \cos Nv = 0, \\ \operatorname{sh} Nu \sin Nv = \pm\dfrac{1}{\varepsilon}. \end{cases}$$

Перше рівняння виконується при $\cos Nv = 0$, тому $v = \frac{(2k-1)\pi}{2N}, k \in \mathbb{N}$. При цьому $\sin Nv = \pm 1$, тому з другого рівняння отримуємо $u = \pm\frac{1}{N}\operatorname{sh}^{-1}\frac{1}{\varepsilon}$.

Таким чином,

$$\sigma_i = \pm \operatorname{sh}\left(\frac{1}{N}\operatorname{sh}^{-1}\frac{1}{\varepsilon}\right)\sin\frac{2i-1}{2N}\pi,$$

$$\Omega_i = \operatorname{ch}\left(\frac{1}{N}\operatorname{sh}^{-1}\frac{1}{\varepsilon}\right)\cos\frac{2i-1}{2N}\pi.$$

$\frac{\sigma_i^2}{\operatorname{sh}^2 u} + \frac{\Omega_i^2}{\operatorname{sh}^2 u} = 1$, тому полюси $s = \sigma_i \pm j\Omega_i$ розташовані на еліпсі, осі якого мають довжину

$$\sigma_0 = \operatorname{sh}\left(\frac{1}{N}\operatorname{sh}^{-1}\frac{1}{\varepsilon}\right), \Omega_0 = \operatorname{ch}\left(\frac{1}{N}\operatorname{sh}^{-1}\frac{1}{\varepsilon}\right)$$

Позначимо $\Lambda = \frac{1}{N}\operatorname{sh}^{-1}\frac{1}{\varepsilon}$ та підставимо $\varepsilon = \sqrt{10^{\frac{A_p}{10}} - 1}$ так, що

$$\Lambda = \frac{1}{N}\operatorname{sh}^{-1}\frac{1}{\varepsilon} = \frac{1}{N}\ln\left(\frac{1}{\varepsilon} + \sqrt{1 + \frac{1}{\varepsilon^2}}\right) = \frac{1}{N}\ln\sqrt{\frac{10^{\frac{A_p}{20}} + 1}{10^{\frac{A_p}{20}} - 1}},$$

отримаємо остаточні вирази для обчислення полюсів для апроксимації за Чебишовим у вигляді

$$\begin{gathered} \sigma_\mu \pm j\Omega_\mu = \sigma_0 \sin\frac{2\mu-1}{2N}\pi \pm j\Omega_0 \cos\frac{2\mu-1}{2N}\pi, \\ \sigma_0 = \operatorname{sh}\Lambda, \Omega_0 = \operatorname{ch}\Lambda, \end{gathered} \tag{3.14}$$

$$\Lambda = \frac{1}{2N}\ln\left(\frac{10^{\frac{A_p}{20}}+1}{10^{\frac{A_p}{20}}-1}\right)$$

де μ відповідає номеру ланцюга другого порядку від 1 до M.

Якщо N непарне, то один з полюсів є дійсним:

$$s_0 = -\sigma_0 \tag{3.15}$$

Масштабний коефіцієнт визначимо з умови

$$H(s) = \begin{cases} 1 \text{ для непарних } N \\ \dfrac{1}{\sqrt{1+\varepsilon^2}} = 10^{-\frac{A_p}{20}} \text{ для парних } N \end{cases}$$

$$a_0 = \begin{cases} \sigma_0 \prod_{\mu=1}^{M} (\sigma_\mu^2 + \Omega_\mu^2) \text{ для непарних } N \\ 10^{-\frac{A_p}{20}} \prod_{\mu=1}^{M} (\sigma_\mu^2 + \Omega_\mu^2) \text{ для парних } N \end{cases} \tag{3.16}$$

У випадку апроксимації за Кауером передаточна функція має як полюси, так і нулі, що обчислюються з коренів функції $L(-s^2) \overset{\omega=\frac{s}{j}}{\longleftarrow} L(\omega^2)$, яка має вигляд

$$L(\omega^2) = \begin{cases} 1 + \varepsilon^2 \operatorname{sn}^2\left(N\dfrac{K_1}{K}\operatorname{sn}^{-1}(\omega,k),k_1\right) \text{ для непарних } N \\ 1 + \varepsilon^2 \operatorname{sn}^2\left(K_1 + N\dfrac{K_1}{K}\operatorname{sn}^{-1}(\omega,k),k_1\right) \text{ для парних } N \end{cases}$$

Позначимо $z = \operatorname{sn}^{-1}(\omega,k)$ і знайдемо спочатку корені функції $L(z)$. Для непарних N функцію $L(z)$ можна записати у вигляді

$$L(z) = \left(1 + j\varepsilon \operatorname{sn}\left(N\frac{K_1}{K}z,k_1\right)\right)\left(1 - j\varepsilon \operatorname{sn}\left(N\frac{K_1}{K}z,k_1\right)\right).$$

Якщо z_1 - корінь першого множника, то $-z_1$ - корінь другого множника, тому що еліптичний синус є непарною функцією z. Отже, нулі і полюси $L(z)$ можна визначити, розв'язуючи рівняння $\operatorname{sn}\left(N\frac{K_1}{K}z,k_1\right) = \frac{j}{\varepsilon}$.

В більшості випадків величина $k_1 = \frac{1}{\sqrt{D}}$ мала, тому можна прийняти $k_1 \approx 0$ і, враховуючи, що $K_1(0) = \frac{\pi}{2}$, можна записати

$$\operatorname{sn}\left(N\frac{K_1}{K}z, k_1\right) = \sin N\frac{\pi}{2}Kz = \frac{j}{\varepsilon}$$

або

$$-jN\frac{\pi}{2K}z = \operatorname{sh}^{-1}\frac{1}{\varepsilon}.$$

Враховуючи, що $\operatorname{sh}^{-1} x = \ln\left(x + \sqrt{x^2+1}\right)$, отримуємо вираз для нуля функції $L(z)$:

$$z_0 = j\frac{K}{\pi N}\ln\left(\frac{1}{\varepsilon} + \sqrt{\frac{1}{\varepsilon^2}+1}\right) = j\frac{K}{2\pi N}\ln\frac{10^{\frac{A_p}{20}}+1}{10^{\frac{A_p}{20}}-1}.$$

Позначимо $\Lambda = \frac{1}{2N}\ln\frac{10^{\frac{A_p}{20}}+1}{10^{\frac{A_p}{20}}-1}$, тоді $z_0 = j\frac{K}{\pi}\Lambda = jv_0$.

Оскільки функція $\operatorname{sn}\left(N\frac{K_1}{K}z, k_1\right)$ є періодичною з періодом $\frac{4K}{N}$, $z_i = z_0 + \frac{4Ki}{N}$ також є нулями $L(z)$.

Нулі функції $L(\omega^2)$ знаходимо на основі відношення між ω і z:

$$z = \operatorname{sn}^{-1}(\omega, k)$$

або

$$\frac{\omega}{\sqrt{k}} = \operatorname{sn}(z, k).$$

Нулі функції $L(-s^2)$ можна визначити, поклавши $s = j\omega$. Для $i = 0$ існує нуль функції $L(-s^2)$

$$s = \sigma_0 = j\Omega_0 = j\sqrt{k}\operatorname{sn}(jv_0, k).$$

Всього існує $N-1$ різних комплексних нулів для *i* від 1 до $N-1$:

$$s = \sigma_i + j\Omega_i = j\sqrt{k}\operatorname{sn}\left(jv_0 + \frac{4Ki}{N}, k\right).$$

Можна показати, що $\operatorname{sn}\left(jv_0 + \frac{4Ki}{N}, k\right) = (-1)^i \operatorname{sn}\left(jv_0 \pm \frac{2Ki}{N}, k\right)$, тому

$$\sigma_i + j\Omega_i = j\sqrt{k}(-1)^i \operatorname{sn}\left(jv_0 \pm \frac{2Ki}{N}, k\right), i = 1,2, \dots, \frac{N-1}{2}.$$

Враховуючи формулу додавання

$$\operatorname{sn}(z_1 + z_2, k) = \frac{\operatorname{sn}(z_1,k)\operatorname{cn}(z_2,k)\operatorname{dn}(z_2,k) + \operatorname{cn}(z_1,k)\operatorname{sn}(z_2,k)\operatorname{dn}(z_1,k)}{1 + k^2\operatorname{sn}^2(z_1,k)\operatorname{sn}^2(z_2,k)},$$

а також формули взаємозв'язку між еліптичними функціями, можна записати вираз для комплексних нулів $L(-s^2)$ у вигляді

$$\sigma_i + j\Omega_i = \frac{(-1)^i\sigma_0 V_i \pm j\Omega_{ki}W}{1 + \sigma_0^2\Omega_{ki}^2},$$

де $i = 1,2, \dots, \frac{N-1}{2}$;

$$\sigma_0 = j\sqrt{k}\operatorname{sn}(jv_0, k); V_i = \sqrt{(1 - k\Omega_{ki}^2)\left(1 - \frac{\Omega_{ki}^2}{k}\right)};$$

$$\Omega_{ki} = \sqrt{k}\operatorname{sn}\left(\frac{2Ki}{N}, k\right); W = \sqrt{(1 + k\sigma_0^2)\left(1 + \frac{\sigma_0^2}{k}\right)}.$$

При парному N функція $L(z) = \operatorname{sn}\left(N\frac{K_1}{K}z + K_1, k_1\right)$ має нулі

$$\pm(\sigma_i + j\Omega_i) = \frac{\pm\left(\sigma_0 V_i \pm j(-1)^i\Omega_{ki}W\right)}{1 + \sigma_0^2\Omega_{ki}^2}.$$

Таким чином, нулі и полюси $H(s)$ при парному N розраховуються аналогічно випадку непарного N, тільки i в виразі для Ω_i змінюється на $i - \frac{1}{2}$.

Використовуючи формули обчислення еліптичного синуса через θ-функції, отримуємо співвідношення для обчислення полюсів и нулів $H(s)$:

$$\sigma_0 = j\sqrt{k}\operatorname{sn}(jv_0, k) = j\frac{\theta_1\left(\frac{jv_0}{2K}, q\right)}{\theta_0\left(\frac{jv_0}{2K}, q\right)} = j\frac{2q^{\frac{1}{4}}\sum_{m=0}^{\infty}(-1)^m q^{m(m+1)}\sin\left(\frac{2m+1}{2K}\pi jv_0\right)}{1 + 2\sum_{m=1}^{\infty}(-1)^m q^{m^2}\cos\left(\frac{2m}{2K}\pi jv_0\right)}$$

Після підстановки $v_0 = \frac{K\Lambda}{\pi}$ отримуємо

$$\sigma_0 = j\frac{2q^{\frac{1}{4}}\sum_{m=0}^{\infty}(-1)^m q^{m(m+1)}\sin j(2m+1)\Lambda}{1 + 2\sum_{m=1}^{\infty}(-1)^m q^{m^2}\cos j2m\Lambda}$$

і, враховуючи, що $\sin jx = -\frac{\operatorname{sh} x}{j}, \cos jx = \operatorname{ch} x,$

$$\sigma_0 = \frac{2q^{\frac{1}{4}}\sum_{m=0}^{\infty}(-1)^m q^{m(m+1)}\,\mathrm{sh}(2m+1)\Lambda}{1+2\sum_{m=1}^{\infty}(-1)^m q^{m^2}\,\mathrm{ch}\,2m\Lambda} \tag{3.17}$$

Аналогічно можна показати, що

$$\Omega_{ki} = \sqrt{k}\,\mathrm{sn}\left(\frac{2K\lambda}{N}, k\right) = \frac{2q^{\frac{1}{4}}\sum_{m=0}^{\infty}(-1)^m q^{m(m+1)}\sin\frac{2m+1}{N}\pi\lambda}{1+2\sum_{m=1}^{\infty}(-1)^m q^{m^2}\cos\frac{2m}{N}\pi\lambda}, \tag{3.18}$$

де $\lambda = \begin{cases} i \text{ для непарних } N \\ i-\frac{1}{2} \text{ для парних } N \end{cases}, i = 1,2,\dots,M.$

Розрахований таким чином прототип має частоту зрізу, що дорівнює $\omega_p = \sqrt{k}$. Для отримання нормованого прототипу необхідно відкорегувати отримані нулі і полюси, розділивши їх на ω_p. Це можна виконати, замінивши вирази для обчислення коефіцієнтів V_i, W наступним чином:

$$V_i = \frac{\sqrt{(1-k\Omega_{ki}^2)\left(1-\frac{\Omega_{ki}^2}{k}\right)}}{\sqrt{k}};\ W = \frac{\sqrt{(1+k\sigma_0^2)\left(1+\frac{\sigma_0^2}{k}\right)}}{\sqrt{k}}. \tag{3.19}$$

Для практичних обчислень суми рядів в (3.17) та (3.18) можна замінити на наближення при $m \le 1$:

$$\sigma_0 = \frac{2q^{\frac{1}{4}}(\mathrm{sh}\,\Lambda - q^2\,\mathrm{sh}\,3\Lambda)}{1-2q\,\mathrm{ch}\,2\Lambda}, \tag{3.20}$$

$$\Omega_{ki} = \sqrt{k}\,\mathrm{sn}\left(\frac{2K\lambda}{N}, k\right) = \frac{2q^{\frac{1}{4}}\left(\sin\frac{\pi\lambda}{N} - q^2\sin\frac{3\pi\lambda}{N}\right)}{1-2q\cos\frac{2\pi\lambda}{N}}. \tag{3.21}$$

Остаточно отримуємо комплексні спряжені пари полюсів апроксимуючої функції Кауера у вигляді

$$\sigma_\mu \pm j\Omega_\mu = \frac{\sigma_0 V_i}{1+\sigma_0^2\Omega_{ki}^2} \pm \frac{j\Omega_{ki}W}{1+\sigma_0^2\Omega_{ki}^2} \tag{3.22}$$

з одним дійсним полюсом у випадку непарного N

$$s_0 = -\frac{\sigma_0}{\sqrt{k}}, \tag{3.23}$$

комплексні спряжені пари нулів у вигляді

$$\sigma_{z\mu} \pm j\Omega_{z\mu} = \pm j\frac{1}{\Omega_{ki}\sqrt{k}} \tag{3.24}$$

та масштабний коефіцієнт у вигляді

$$a_0 = \begin{cases} \sigma_0\sqrt{k}\prod_{\mu=1}^{M}\left(\left(\sigma_\mu^2 + \Omega_\mu^2\right)\Omega_{ki}^2\right) \text{ для непарних } N \\ 10^{-\frac{A_p}{20}}\prod_{\mu=1}^{M}\left(\left(\sigma_\mu^2 + \Omega_\mu^2\right)\Omega_{ki}^2\right) \text{ для парних } N. \end{cases} \tag{3.25}$$

3.5 Перетворення нулів та полюсів фільтра-прототипу на нулі та полюси цифрового фільтра

Перетворення нулів та полюсів прототипу на нулі та полюси цифрового фільтра у випадку білінійного перетворення виконується за допомогою формул, наведених у таблиці 3.3 шляхом підстановки $s = s_0$ у випадку дійсних полюсів або нулів та підстановки $s = \sigma \pm j\Omega$ у випадку комплексних спряжених пар з подальшим коригуванням масштабного коефіцієнта. Новий масштабний коефіцієнт визначається в залежності від кількості і типу нулів та полюсів ланцюга.

3.5.1 Фільтри низьких частот

У випадку дійсного полюса або нуля маємо

$$s = s_0 = c\frac{1 - z^{-1}}{1 + z^{-1}};$$

$$s_0 + s_0 z^{-1} = c - cz^{-1};$$

$$z^{-1}(s_0 + c) = c - s_0;$$

$$z^{-1} = \frac{c - s_0}{c + s_0} \Leftrightarrow z = \frac{c + s_0}{c - s_0}. \tag{3.26}$$

Ланцюги першого порядку, що можна отримати з апроксимацій Батерворта, Чебишова та Кауера, завжди мають один дійсний полюс та не мають нулів, тобто мають вигляд

Таблиця 3.3. Формули перетворення фільтру-прототипу на цифровий фільтр

Тип фільтра	Вимоги до прототипу
ФНЧ	$s = c\dfrac{1-z^{-1}}{1+z^{-1}}, c = \operatorname{ctg}\dfrac{2\pi f_p}{2}$
ФВЧ	$s = c\dfrac{1+z^{-1}}{1-z^{-1}}, c = \operatorname{tg}\dfrac{2\pi f_p}{2}$
СФ	$s = \alpha\dfrac{1-2\beta z^{-1}+z^{-2}}{1-z^{-2}},$ $\alpha = \operatorname{ctg}\dfrac{2\pi\left(f_{p2}-f_{p1}\right)}{2}, \beta = \dfrac{\sin\left(2\pi\left(f_{p1}+f_{p2}\right)\right)}{\sin 2\pi f_{p1} + \sin 2\pi f_{p2}}$
РФ	$s = \alpha\dfrac{1-z^{-2}}{1-2\beta z^{-1}+z^{-2}},$ $\alpha = \dfrac{\cos 2\pi f_{p1} - \cos 2\pi f_{p2}}{\sin 2\pi f_{p1} + \sin 2\pi f_{p2}}, \beta = \dfrac{\sin\left(2\pi\left(f_{p1}+f_{p2}\right)\right)}{\sin 2\pi f_{p1} + \sin 2\pi f_{p2}}$

$$\frac{1}{s-s_0} = \frac{1}{c\dfrac{1-z^{-1}}{1+z^{-1}} - s_0} = \frac{1+z^{-1}}{c - cz^{-1} - s_0 - s_0 z^{-1}} = \frac{1}{c-s_0}\frac{1+z^{-1}}{1+B_{1\mu}z^{-1}}.$$

Таким чином, щоб привести ланцюг до стандартного вигляду $\frac{1+A_{1\mu}z^{-1}}{1+B_{1\mu}z^{-1}}$, масштабний коефіцієнт необхідно домножити на коефіцієнт

$$a_{0\mu} = \frac{1}{c-s_0}.$$

Крім того, цифровий фільтр буде мати нуль в точці -1, якого не було у прототипу.

У випадку комплексної спряженої пари маємо

$$s = \sigma \pm j\Omega = c\frac{1-z^{-1}}{1+z^{-1}};$$

$$\sigma \pm j\Omega + \sigma z^{-1} \pm j\Omega z^{-1} = c - cz^{-1};$$

$$z^{-1}(c + \sigma \pm j\Omega) = c - \sigma \mp j\Omega;$$

$$z = \frac{c+\sigma \pm j\Omega}{c-\sigma \mp j\Omega} = \frac{c^2-\sigma^2-\Omega^2}{(c-\sigma)^2+\Omega^2} \pm j\frac{2c\Omega}{(c-\sigma)^2+\Omega^2} \qquad (3.27)$$

Ланцюг другого порядку у випаду апроксимації Батерворта або Чебишова матиме вигляд

$$\frac{1}{(s-\sigma-j\Omega)(s-\sigma+j\Omega)}=\frac{1}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}.$$

Після підстановки та спрощення отримаємо

$$\frac{1}{\sigma^2+\Omega^2-2\sigma c+c^2}\frac{(1+z^{-1})^2}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

З’являється нуль другого порядку в точці -1, а масштабний коефіцієнт домножується на $\frac{1}{\sigma^2+\Omega^2-2\sigma c+c^2}$.

У випадку апроксимації Кауера ланцюги другого порядку мають комплексну спряжену пару нулів та комплексну спряжену пару полюсів. В такому випадку нулі та полюси перетворюються окремо, а ланцюг має вигляд

$$\frac{(s-\sigma_z-j\Omega_z)(s_z-\sigma_z+j\Omega_z)}{(s-\sigma-j\Omega)(s-\sigma+j\Omega)}=\frac{(\sigma_z^2+\Omega_z^2)+(-2\sigma_z)s+s^2}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}.$$

Після підстановки та спрощення отримаємо

$$\frac{\sigma_z^2+\Omega_z^2-2\sigma_z c+c^2}{\sigma^2+\Omega^2-2\sigma c+c^2}\frac{1+A_{1\mu}z^{-1}+A_{2\mu}z^{-2}}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

Отже, додаткових нулів чи полюсів не з’являється, а масштабний коефіцієнт домножується на $\frac{\sigma_z^2+\Omega_z^2-2\sigma_z c+c^2}{\sigma^2+\Omega^2-2\sigma c+c^2}$.

3.5.2 Фільтри високих частот

У випадку дійсного полюса або нуля маємо

$$s=s_0=c\frac{1+z^{-1}}{1-z^{-1}};$$

$$s_0-s_0z^{-1}=c+cz^{-1};$$

$$z^{-1}(s_0+c)=s_0-\text{c};$$

$$z^{-1}=\frac{s_0-c}{s_0+c}\Leftrightarrow z=\frac{s_0+c}{s_0-c} \qquad (3.28)$$

Ланцюг має вигляд

$$\frac{1}{s-s_0}=\frac{1}{c\dfrac{1+z^{-1}}{1-z^{-1}}-s_0}=\frac{1-z^{-1}}{c+cz^{-1}-s_0+s_0z^{-1}}=\frac{1}{c-s_0}\frac{1-z^{-1}}{1+B_{1\mu}z^{-1}}.$$

Отже, з'являється нуль у точці 1, а масштабний коефіцієнт домножується на $\frac{1}{c-s_0}$.

У випадку комплексної спряженої пари маємо

$$s=\sigma\pm j\Omega=c\frac{1+z^{-1}}{1-z^{-1}};$$

$$\sigma\pm j\Omega-\sigma z^{-1}\mp j\Omega z^{-1}=c+cz^{-1};$$

$$z^{-1}(c+\sigma\pm j\Omega)=-c+\sigma\pm j\Omega;$$

$$z=\frac{-c+\sigma\pm j\Omega}{c+\sigma\pm j\Omega}=\frac{-c^2+\sigma^2+\Omega^2}{(c-\sigma)^2+\Omega^2}\mp j\frac{2c\Omega}{(c-\sigma)^2+\Omega^2} \qquad (3.29)$$

Ланцюг другого порядку у випаду апроксимації Батерворта або Чебишева матиме вигляд $\frac{1}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}$. Після підстановки та спрощення отримаємо

$$\frac{1}{\sigma^2+\Omega^2-2\sigma c+c^2}\frac{(1-z^{-1})^2}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

З'являється нуль другого порядку в точці 1, а масштабний коефіцієнт домножується на $\frac{1}{\sigma^2+\Omega^2-2\sigma c+c^2}$.

У випадку апроксимації Кауера ланцюг має вигляд $\frac{(\sigma_z^2+\Omega_z^2)+(-2\sigma_z)s+s^2}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}$. Після підстановки та спрощення отримаємо

$$\frac{\sigma_z^2+\Omega_z^2-2\sigma_z c+c^2}{\sigma^2+\Omega^2-2\sigma c+c^2}\frac{1+A_{1\mu}z^{-1}+A_{2\mu}z^{-2}}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

Отже, додаткових нулів чи полюсів не з'являється, а масштабний коефіцієнт домножується на $\frac{\sigma_z^2+\Omega_z^2-2\sigma_z c+c^2}{\sigma^2+\Omega^2-2\sigma c+c^2}$, аналогічно ФНЧ.

3.5.3 Смугові фільтри

У випадку дійсного полюса або нуля маємо

$$s=s_0=\ \alpha\frac{1-2\beta z^{-1}+z^{-2}}{1-z^{-2}};$$

$$s_0 - s_0 z^{-2} = \alpha - 2\alpha\beta z^{-1} + \alpha z^{-2};$$

$$z^{-2}(\alpha + s_0) - 2\alpha\beta z^{-1} + (\alpha - s_0) = 0.$$

Необхідно розв'язати квадратне рівняння з дискримінантом $D = 4(\alpha^2\beta^2 - - \alpha^2 + s_0^2)$. Якщо $D \geq 0$, то отримаємо 2 дійсних корені $z_{1,2}^{-1} = \frac{2\alpha\beta \pm 2\sqrt{\alpha^2\beta^2 - \alpha^2 + s_0^2}}{2(\alpha + s_0)} = \frac{\alpha\beta \pm d}{\alpha + s_0}$, якщо позначити $d = \sqrt{D}$. Звідси

$$z_{1,2} = \frac{\alpha + s_0}{\alpha\beta \pm d} \tag{3.30}$$

Якщо ж $D < 0$, то, позначивши $d = \sqrt{-D}$, отримаємо 2 комплексних спряжених корені

$$z_{1,2}^{-1} = \frac{\alpha\beta}{\alpha + s_0} \pm j\frac{d}{\alpha + s_0}$$

або

$$z_{1,2} = \frac{\alpha\beta(\alpha + s_0)}{\alpha^2\beta^2 + d^2} \mp j\frac{d(\alpha + s_0)}{\alpha^2\beta^2 + d^2} \tag{3.31}$$

Ланцюг має вигляд

$$\frac{1}{s - s_0} = \frac{1}{\alpha\dfrac{1 - 2\beta z^{-1} + z^{-2}}{1 - z^{-2}} - s_0} = \frac{1}{\alpha - s_0}\frac{(1 - z^{-1})(1 + z^{-1})}{1 + B_{1\mu}z^{-1} + B_{2\mu}z^{-2}}.$$

Отже, з'являються нулі в точках 1 та -1, а масштабний коефіцієнт домножується на $\frac{1}{\alpha - s_0}$.

У випадку комплексної спряженої пари маємо

$$s = \sigma \pm j\Omega = \alpha\frac{1 - 2\beta z^{-1} + z^{-2}}{1 - z^{-2}};$$

$$\sigma \pm j\Omega - \sigma z^{-2} \pm j\Omega z^{-2} = \alpha - 2\alpha\beta z^{-1} + \alpha z^{-2};$$

$$z^{-2}(\alpha + \sigma \pm j\Omega) - 2\alpha\beta z^{-1} + (\alpha - \sigma \mp j\Omega) = 0.$$

Отримали квадратне рівняння з комплексними коефіцієнтами та дискримінантом $D = 4(\alpha^2\beta^2 - \alpha^2 + \sigma^2 - \Omega^2 + j2\sigma\Omega) = 4D_1$.

Позначимо $d_{1,2} = x \pm jy$ комплексні квадратні корені з D_1. Ці корені можна знайти за допомогою формули

$$(a+jb)^{\frac{1}{2}}=\begin{cases}\sqrt{\frac{\sqrt{a^2+b^2}+a}{2}}+j\sqrt{\frac{\sqrt{a^2+b^2}-a}{2}}, b\geq 0,\\ \sqrt{\frac{\sqrt{a^2+b^2}+a}{2}}-j\sqrt{\frac{\sqrt{a^2+b^2}-a}{2}}, b<0\end{cases} \quad (3.32)$$

Тоді

$$z_{1,2,3,4}^{-1}=\frac{\alpha\beta\pm(x\pm jy)}{\alpha+\sigma\pm j\Omega}.$$

Після розкриття дужок та приведення до дійсного знаменника отримуємо:

$$\begin{aligned} z_{1,2}&=\frac{\alpha^2\beta+\alpha\beta\sigma+x\alpha+x\sigma+y\Omega}{(\alpha+\sigma)^2+\Omega^2}\pm j\frac{-\alpha\beta\Omega-x\Omega+\alpha y+\sigma y}{(\alpha+\sigma)^2+\Omega^2}\\ z_{3,4}&=\frac{\alpha^2\beta+\alpha\beta\sigma-x\alpha-x\sigma-y\Omega}{(\alpha+\sigma)^2+\Omega^2}\pm j\frac{-\alpha\beta\Omega+x\Omega-\alpha y-\sigma y}{(\alpha+\sigma)^2+\Omega^2}\end{aligned} \quad (3.33)$$

Ланцюг другого порядку у випаду апроксимації Батерворта або Чебишева матиме вигляд $\frac{1}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}$. Після підстановки та спрощення отримаємо

$$\frac{1}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}\frac{(1-z^{-1})^2(1+z^{-1})^2}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

З’являються нулі другого порядку в точках 1 та -1, а масштабний коефіцієнт домножується на $\frac{1}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}$.

У випадку апроксимації Кауера ланцюг має вигляд $\frac{(\sigma_z^2+\Omega_z^2)+(-2\sigma_z)s+s^2}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}$. Після підстановки та спрощення отримаємо

$$\frac{\sigma_z^2+\Omega_z^2-2\sigma_z\alpha+\alpha^2}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}\frac{1+A_{1\mu}z^{-1}+A_{2\mu}z^{-2}}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

Отже, додаткових нулів чи полюсів не з’являється, а масштабний коефіцієнт домножується на $\frac{\sigma_z^2+\Omega_z^2-2\sigma_z\alpha+\alpha^2}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}$.

3.5.4 Режекторні фільтри

У випадку дійсного полюса або нуля маємо

$$s = s_0 = \alpha\frac{1 - z^{-2}}{1 - 2\beta z^{-1} + z^{-2}};$$

$$s_0 - 2s_0\beta z^{-1} + s_0 z^{-2} = \alpha - \alpha z^{-2};$$

$$z^{-2}(s_0 + \alpha) - 2s_0\beta z^{-1} + (s_0 - \alpha) = 0.$$

Необхідно розв'язати квадратне рівняння з дискримінантом $D = 4(s_0^2\beta^2 - - \alpha^2 + s_0^2) = 4D_1$. Якщо $D_1 \geq 0$ і $d = \sqrt{D_1}$, то отримаємо 2 дійсних корені

$z_{1,2}^{-1} = \frac{s_0\beta \pm d}{s_0 + \alpha}$. Звідси

$$z_{1,2} = \frac{\alpha + s_0}{s_0\beta \pm d} \tag{3.34}$$

Якщо ж $D < 0$, то, позначивши $d = \sqrt{-D}$, отримаємо 2 комплексних спряжених корені

$$z_{1,2}^{-1} = \frac{s_0\beta}{\alpha + s_0} \pm j\frac{d}{\alpha + s_0}$$

або

$$z_{1,2} = \frac{s_0\beta(\alpha + s_0)}{s_0^2\beta^2 + d^2} \mp j\frac{d(\alpha + s_0)}{s_0^2\beta^2 + d^2} \tag{3.35}$$

Ланцюг має вигляд

$$\frac{1}{s - s_0} = \frac{1}{\alpha\dfrac{1 - z^{-2}}{1 - 2\beta z^{-1} + z^{-2}} - s_0} = \frac{1}{\alpha - s_0}\frac{1 - 2\beta z^{-1} + z^{-2}}{1 + B_{1\mu}z^{-1} + B_{2\mu}z^{-2}}.$$

Розклавши чисельник на множники, отримуємо

$$1 - 2\beta z^{-1} + z^{-2} = \left(1 - \beta z^{-1} - jz^{-1}\sqrt{1 - \beta^2}\right)\left(1 - \beta z^{-1} + jz^{-1}\sqrt{1 - \beta^2}\right)$$

Отже, з'являються нулі в точках $\beta \pm j\sqrt{1 - \beta^2}$, а масштабний коефіцієнт домножується на $\frac{1}{\alpha - s_0}$.

У випадку комплексної спряженої пари маємо

$$s = \sigma \pm j\Omega = \alpha\frac{1 - z^{-2}}{1 - 2\beta z^{-1} + z^{-2}};$$

$$\sigma \pm j\Omega - 2\beta\sigma z^{-1} \mp j2\beta\Omega z^{-1} + \sigma z^{-2} \pm j\Omega z^{-2} = \alpha - \alpha z^{-2};$$

$$z^{-2}(\alpha+\sigma\pm j\Omega)-2\beta z^{-1}(\sigma+j\Omega)+(-\alpha+\sigma\pm j\Omega z^{-2})=0.$$

Отримали квадратне рівняння з комплексними коефіцієнтами та дискримінантом $D=4\big((\sigma^2-\Omega^2)(\beta^2-1)+\alpha^2+j2\sigma\Omega(\beta^2-1)\big)=4D_1$.

Позначимо $d_{1,2}=x\pm jy$ комплексні квадратні корені з D_1, знайдені за допомогою (3.32). Тоді

$$z_{1,2,3,4}^{-1}=\frac{\beta(\sigma\pm j\Omega)\pm(x\pm jy)}{\alpha+\sigma\pm j\Omega}.$$

Після розкриття дужок та приведення до дійсного знаменника отримуємо:

$$\begin{aligned} z_{1,2}&=\frac{\alpha\beta\sigma+\beta\sigma^2+\alpha x+\sigma x+\beta\Omega^2+\Omega y}{(\alpha+\sigma)^2+\Omega^2}\pm j\frac{\alpha\beta\Omega-x\Omega+\sigma y+\alpha y}{(\alpha+\sigma)^2+\Omega^2}\\ z_{3,4}&=\frac{\alpha\beta\sigma+\beta\sigma^2-\alpha x-\sigma x+\beta\Omega^2-\Omega y}{(\alpha+\sigma)^2+\Omega^2}\pm j\frac{\alpha\beta\Omega+x\Omega-\sigma y-\alpha y}{(\alpha+\sigma)^2+\Omega^2}\end{aligned} \qquad (3.36)$$

Ланцюг другого порядку у випаду апроксимації Батерворта або Чебишева матиме вигляд $\frac{1}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}$. Після підстановки та спрощення отримаємо

$$\frac{1}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}\frac{(1-2\beta z^{-1}+z^{-2})^2}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

З’являються нулі другого порядку в точках $\beta\pm j\sqrt{1-\beta^2}$, а масштабний коефіцієнт домножується на $\frac{1}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}$.

У випадку апроксимації Кауера ланцюг має вигляд $\frac{(\sigma_z^2+\Omega_z^2)+(-2\sigma_z)s+s^2}{(\sigma^2+\Omega^2)+(-2\sigma)s+s^2}$. Після підстановки та спрощення отримаємо

$$\frac{\sigma_z^2+\Omega_z^2-2\sigma_z\alpha+\alpha^2}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}\frac{1+A_{1\mu}z^{-1}+A_{2\mu}z^{-2}}{1+B_{1\mu}z^{-1}+B_{2\mu}z^{-2}}.$$

Отже, додаткових нулів чи полюсів не з’являється, а масштабний коефіцієнт домножується на $\frac{\sigma_z^2+\Omega_z^2-2\sigma_z\alpha+\alpha^2}{\sigma^2+\Omega^2-2\sigma\alpha+\alpha^2}$, аналогічно смуговим фільтрам.

3.6 Обчислення передаточної функції цифрового фільтра

Після обчислення всіх нулів та полюсів цифрового фільтра, а також масштабного коефіцієнта A_0 передаточну функцію фільтра можна обчислити наступним чином:

$$H(z) = A_0 \prod_{\mu=1}^{M} H_\mu(z),$$

де M – загальна кількість ланцюгів фільтру;

$H_\mu(z)$ – передаточна функція μ-того ланцюга, що має вигляд

$$H_\mu(z) = \frac{\prod_{i=1}^{P}(1 - z_{zi}z^{-1})}{\prod_{i=1}^{Q}(1 - z_{pi}z^{-1})},$$

де P – кількість нулів ланцюга (менше або дорівнює порядку ланцюга);

Q – кількість полюсів ланцюга (дорівнює порядку ланцюга);

z_{zi} – i-тий нуль ланцюга;

z_{pi} – i-тий полюс ланцюга.

Після розкриття дужок передаточна функція приводиться до каскадної форми (2.1).

3.7 Обчислення частотних характеристик цифрового фільтра

З (2.1) можна обчислити АЧХ та ФЧХ цифрового фільтра, виконавши підстановку $z = e^{j\omega} = e^{j2\pi f}$, де f – цифрова частота. Крім того, для фільтра у каскадній формі можна записати

$$|H(\omega)| = A_0 \prod_{\mu=1}^{M} \left|H_\mu(\omega)\right|;$$

$$LH(\omega) = 20\left(\lg A_0 + \sum_{\mu=1}^{M} \lg\left|H_\mu(\omega)\right|\right);$$

$$\Phi(\omega) = \sum_{\mu=1}^{M} \Phi_\mu(\omega),$$

де $|H|$ – АЧХ фільтра;

LH – ЛАЧХ фільтра;

Ф – ФЧХ фільтра.

Таким чином,

$$|H_\mu(f)| = \sqrt{\frac{\left(1 + A_{1\mu}\cos 2\pi f + A_{2\mu}\cos 4\pi f\right)^2 + \left(A_{1\mu}\sin 2\pi f + A_{2\mu}\sin 4\pi f\right)^2}{\left(1 + B_{1\mu}\cos 2\pi f + B_{2\mu}\cos 4\pi f\right)^2 + \left(B_{1\mu}\sin 2\pi f + B_{2\mu}\sin 4\pi f\right)^2}}$$

$$\Phi_\mu(f) = \operatorname{arctg}\frac{B_{1\mu}\sin 2\pi f + B_{2\mu}\sin 4\pi f}{1 + B_{1\mu}\cos 2\pi f + B_{2\mu}\cos 4\pi f} -$$

$$- \operatorname{arctg}\frac{A_{1\mu}\sin 2\pi f + A_{2\mu}\sin 4\pi f}{1 + B_{1\mu}\cos 2\pi f + A_{2\mu}\cos 4\pi f}$$

3.8 Обчислення імпульсної та перехідної характеристик цифрового фільтра

Імпульсні та перехідні характеристики фільтра обчислюються шляхом симуляції роботи фільтру в часі при поданні на вхід сигналів певної форми.

Для фільтру в каскадній формі (2.1) можна пропустити сигнал через кожний ланцюг послідовно, після чого помножити на масштабний коефіцієнт. Таким чином, для кожного ланцюга буде виконуватися рівність

$$y_\mu(n) = x_\mu(n) + A_{1\mu}x_\mu(n-1) + A_{2\mu}x_\mu(n-2) - B_{1\mu}y_\mu(n-1) - B_{2\mu}y_\mu(n-2),$$

де $x_\mu(n)$ – сигнал на вході μ-того ланцюга в момент часу n;

$y_\mu(n)$ – сигнал на виході μ-того ланцюга в момент часу n.

Оскільки ланки з’єднані послідовно, то $x_\mu(n) = y_{\mu-1}(n)$. Позначимо $x(n) = x_1(n)$ – сигнал на вході фільтру та $y(n) = A_0 y_M(n)$ – сигнал на виході фільтру.

Оскільки імпульсна характеристика для нерекурсивних фільтрів є нескінченною в часі, доцільно розрахувати лише певну кількість відліків, необхідну для побудови графіку.

Для того, щоб $y(n)$ представляв собою імпульсну характеристику фільтра, потрібно подати на вхід сигнал $x(n) = \begin{cases} 1, n = 0 \\ 0, n \neq 0. \end{cases}$

Для розрахунку перехідної характеристики необхідно подати на вхід фільтра сигнал $x(n) = \begin{cases} 1, n \geq 0 \\ 0, n < 0. \end{cases}$

3.9 Висновки

У даному розділі було проведено дослідження процесу розрахунку цифрових фільтрів з математичної точки зору. Було отримано формули для обчислення нулів, полюсів та масштабних коефіцієнтів аналогових прототипів Батерворта, Чебишова та Кауера, а також формули, що дозволяють перетворити розраховані нулі та полюси на нулі та полюси цифрового фільтру необхідного типу, водночас скоригувавши масштабний коефіцієнт. Крім того, було розглянуто розрахунок коефіцієнтів передаточної функції в каскадній формі, а також АЧХ, ЛАЧХ, ФЧХ, імпульсної та перехідної характеристик цифрового фільтру.

4 РОЗРОБКА ПІДСИСТЕМИ РОЗРАХУНКІВ

4.1 Побудова веб-сервісу

Обрані в розділі 2 інструменти та досліджені в розділі 3 математичні закономірності дозволяють розробити та реалізувати серверний додаток, що реалізує одну функцію у вигляді веб-сервіса. В якості запиту додаток приймає параметри фільтру та вимоги до його ЛАЧХ, а повертає структуру, що містить всі необхідні для задоволення функціональних вимог до сайту дані. Файли з кодом та XML-описами наведені в додатку А.

Запит містить наступні поля:

- Частоту дискретизації, Гц;
- Тип фільтру – низьких частот, високих частот, смуговий або режекторний;
- Тип апроксимації – Батерворта, Чебишева або Кауера (еліптичний фільтр);
- Межі смуги (смуг) затримання, Гц (не обов’язково, якщо вказаний порядок фільтра та обрана не еліптична апроксимація);
- Межі смуги (смуг) пропускання, Гц;
- Максимальне відхилення сигналу в смузі пропускання, дБ;
- Максимальний рівень сигналу в смузі затримання, дБ (не обов’язково, якщо вказаний порядок фільтра та обрана не еліптична апроксимація);
- Крок за частотою для обчислення АЧХ, ЛАЧХ та ФЧХ, Гц;
- Кількість відліків за часом для обчислення імпульсної та перехідної характеристик;
- Порядок фільтру (не обов’язково, якщо вказані параметри смуги затримання);
- Тип запиту – необхідно побудувати фільтр заданого порядку чи з заданими параметрами смуги затримання.

Відповідь на запит є структурою, що містить наступні дані:

- Список нулів фільтру;
- Список полюсів фільтру;
- Передаточна функція у вигляді масштабного множника та коефіцієнтів окремих ланцюгів;
- Список значень АЧХ, ЛАЧХ та ФЧХ для частот від 0 до половини частоти дискретизації з заданим кроком;
- Список значень імпульсної та перехідної характеристик з заданою кількістю відліків.

Запит, відповідь та формати даних, що передаються в них, описані в файлі «iir.xsd» (див. додаток А). За цим же файлом було автоматично згенеровано код мовою Java, який описує формати повідомлень. Даний код не наводиться в додатку через його надлишковість, однак використовується в інших файлах.

Сам проект та бібліотеки, від яких він залежить, описані в файлі «pom.xml». Це головний файл, що використовується системою Maven для опису залежностей проекту та процесу збірки.

4.2 Огляд архітектури системи

Код додатку міститься в декількох файлах класів мовою Java. Нижче наведений короткий опис кожного з них:

- Application – клас, що містить точку входу в додаток Spring. Є допоміжним та не містить коду, пов’язаного з цифровими фільтрами.
- FilterBuilder – основний клас додатку. Містить методи, що розраховують фільтр за заданими параметрами.
- FilterEndpoint – клас, що описує точку доступу веб-сервісу.
- FilterZeroPoles – клас, що описує фільтр (аналоговий чи цифровий) у вигляді загального масштабного коефіцієнту та списку ланцюгів типу ZPSection;

- ZPSection - клас, що описує ланцюг фільтру у вигляді нулів та полюсів. Як нуль, так і полюс можуть бути відсутніми, дійсними або комплексними; крім того, ланцюг може містити 2 дійсних нулі або полюси.
- WebServiceConfig – допоміжний клас, що описує параметри веб-сервісу, зокрема, URL точки доступу та XSD схему, що використовується для автоматичної генерації WSDL файлу.

Архітектура додатку наведена на рис. 4.1.

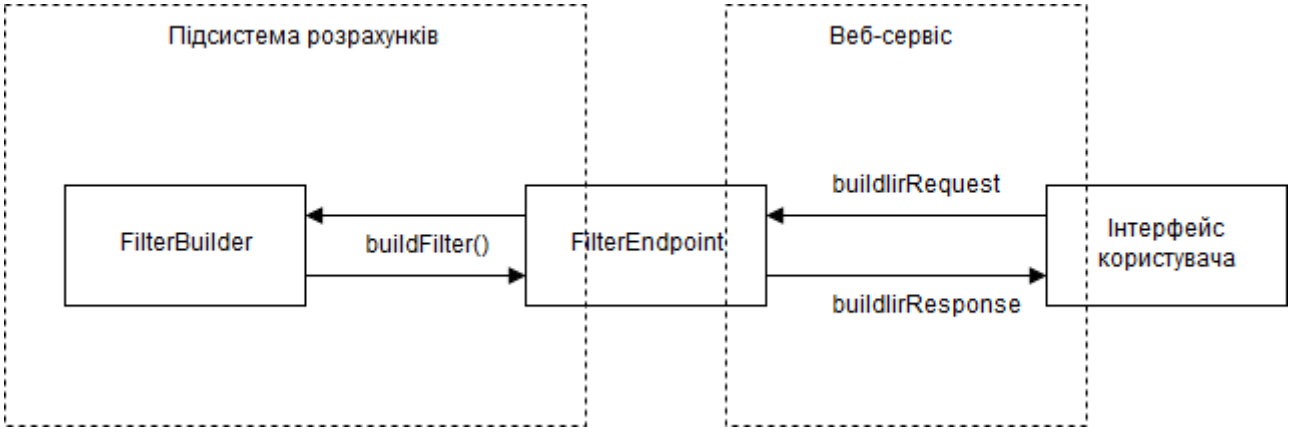

Рисунок 4.1 – Архітектура додатку

4.3 Розрахунок цифрових фільтрів

Методи, що реалізують математичні закономірності, досліджені в розділі 3, містяться в класі FilterBuilder:

- Конструктор класу видобуває з класу запиту параметри смуг пропускання та затримання, а також зберігає дані запиту в полі класу;
- Метод build виконує власне процес побудови ЦФ в такому порядку:
    1) Розраховує порядок фільтру за допомогою методу filterOrder, або, якщо порядок був заданий, обчислює допоміжні параметри для СФ та РФ, а також фільтрів Кауера;
    2) Розраховує фільтр-прототип у вигляді ланцюгів з нулями та полюсами за допомогою методу prototypeZeroPoles;

3) Перетворює розраховані нулі та полюси прототипу на нулі та полюси цифрового фільтра з перерахуванням масштабного коефіцієнта за допомогою методу digitalize;
4) Обчислює коефіцієнти передаточної функції за відомими нулями та полюсами за допомогою методу transferFunction;
5) Заповнює списки нулів та полюсів фільтра для передачі їх у відповідь;
6) Обчислює АЧХ, ЛАЧХ, ФЧХ, імпульсну та перехідну характеристики фільтра за розрахованою передаточною функцією та заданими кроком за частотою та кількістю відліків за часом за допомогою методів amplitude, logAmplitude, phase, pulse та step відповідно;
7) Збирає всі обчислені дані у вигляді об’єкту класу buildIirResponse та повертає як результат виконання.

Окрім методів, що викликаються методом build, клас містить допоміжні методи, що викликаються іншими для спрощення обчислень:

- bandFilterHelper – допоміжна функція, що використовується під час обчислення вимог до прототипу, якщо задані вимоги до СФ або РФ;
- prototypeFrequency – власне обчислює параметр ω_a для визначення порядку фільтра у відповідності з підрозділом 3.3, а також параметри α і β, якщо задані вимоги до СФ або РФ;
- checkOrderParity – допоміжна функція, яка приймає 2 цілих числа та повертає перше з них, якщо порядок фільтру непарний, та друге, якщо порядок фільтру парний;
- sqrt – допоміжна функція, що обчислює квадратний корінь з комплексного числа. Повертає також комплексне число.

4.4 Висновки

У даному розділі було проведено аналіз отриманих результатів, а саме додатку мовою Java, який виконує функції з розрахунку цифрових фільтрів у формі веб-сервісу. Було оглянуто архітектуру системи та основні класи, що складають додаток. Крім того, було описано функції, що використовуються під час побудови ЦФ.

Розроблений додаток успішно розраховує параметри та характеристики цифрових фільтрів низьких та високих частот, смугових та режекторних фільтрів на основі апроксимації Батерворта, Чебишова та Кауера (еліптичні фільтри). Серед даних, що повертає додаток, є полюси та нулі фільтру, коефіцієнти передаточної функції, АЧХ, ЛАЧХ, ФЧХ, імпульсна та перехідна характеристики. Таким чином, додаток задовольняє вимогам, поставленим в розділі 2.

5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для побудови цифрових фільтрів. Продукт представляє собою серверний додаток, що надає можливості з розрахунку параметрів цифрових фільтрів.

Програмний продукт призначено для використання сумісно з окремо розробленим інтерфейсом користувача (веб-сайтом).

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

– для кожної функції визначаються повні річні витрати й кількість робочих часів.

– для кожної функції наоснові оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

– після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

5.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи розрахунку цифрових фільтрів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

– програмний продукт повинен функціонувати на серверах із стандартним набором компонент;

– забезпечувати високу швидкість обробки великих об’ємів даних у реальному часі;

– забезпечувати зручність і простоту взаємодії з розробником інтерфейсу користувача;

– передбачати мінімальні витрати на впровадження програмного продукту.

5.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який аналізує цифровий фільтр за заданими вимогами до його характеристик та розраховує

його основні параметри. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 – вибір мови програмування;

F_2 – розпізнавання вхідних даних;

F_3 – методологія розрахунку цифрових фільтрів.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F_1:

а) мова програмування C#;

б) мова програмування Java;

Функція F_2:

а) SOAP веб-сервіс;

б) звичайні HTTP-запити;

Функція F_3:

а) апроксимація за аналоговим прототипом;

б) апроксимація за допомогою методів оптимізації.

5.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 5.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 5.1).

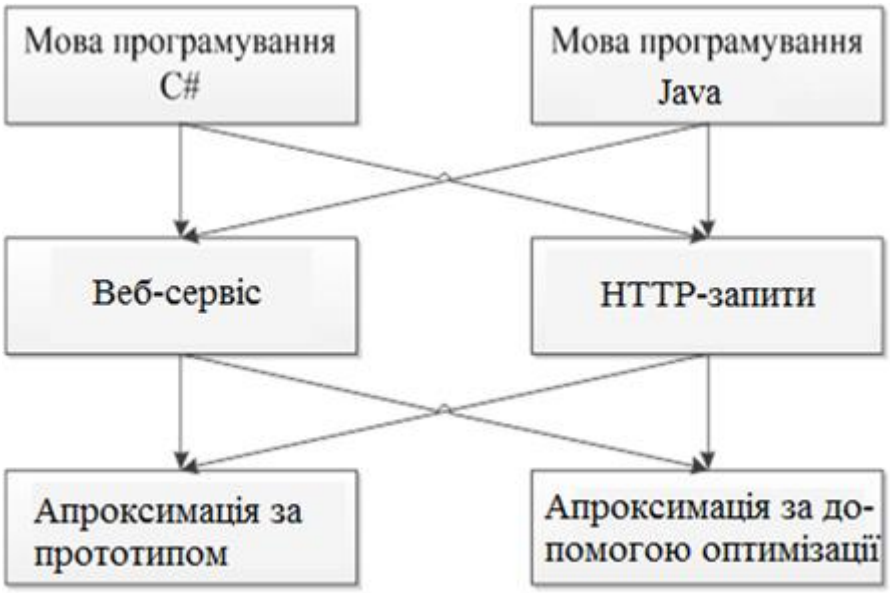

Рисунок 5.1 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 5.1. Позитивно-негативна матриця

Основні функції	**Варіанти реалізації**	**Переваги**	**Недоліки**
F1	*А*	Більше вбудованих можливостей	Прив’язка до Windows
	Б	Крос-платформений	Необхідні додаткові бібліотеки
F2	А	Складніша реалізація	Просто використовувати
	Б	Проста реалізація	Складніше використання
F3	*А*	Простіші розрахунки	Менше можливостей
	Б	Довільні параметри фільтра	Більше розрахунків

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція *F1:*

Серверні додатки недоцільно прив’язувати до однієї операційної системи, тому варіант а) має бути відкинутий

Функція *F2:*

Варіанти а) і б) є прийнятними для побудови сайту, тому є сенс розглядати їх обидва.

Функція *F3:*

Апроксимації за прототипом достатньо для розрахунку більшості цифрових фільтрів, більш складні методи призначені для використання у «важких» додатках та не підходять для сайту, тому варіант б) має бути відкинутий.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1а – F2а – F3а
2. F1а – F2б – F3а

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

5.2 Обґрунтування системи параметрів ПП

5.2.1 Опис параметрів

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

– *X1* – швидкодія мови програмування;

– *X2* – зручність доступу для розробника сайту;

– *X3* – час обробки даних;

– *X4* – потенційний об’єм програмного коду.

X1: Відображає швидкодію операцій залежно від обраної мови програмування.

X2: Відображає кількість операцій, які необхідно виконати розробнику інтерфейсу користувача, щоб отримати доступ до можливостей додатку.

X3: Відображає час, який витрачається на дії.

X4: Показує розмір програмного коду який необхідно створити безпосередньо розробнику.

5.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.2.

Таблиця 5.2. Основні параметри ПП

Назва Параметра	**Умовні позначення**	**Одиниці виміру**	**Значення параметра**		
			гірші	**середні**	**кращі**
Швидкодія мови програмування	X1	Оп/мс	2000	11000	19000
Зручність доступу для розробника сайту	X2	Оп	15	7	2
Час обробки даних алгоритмом	X3	мс	800	420	60
Потенційний об’єм програмного коду	X4	кількість строк коду	3000	2200	1100

За даними таблиці 5.2 будуються графічні характеристики параметрів – рис. 5.2 – рис. 5.5.

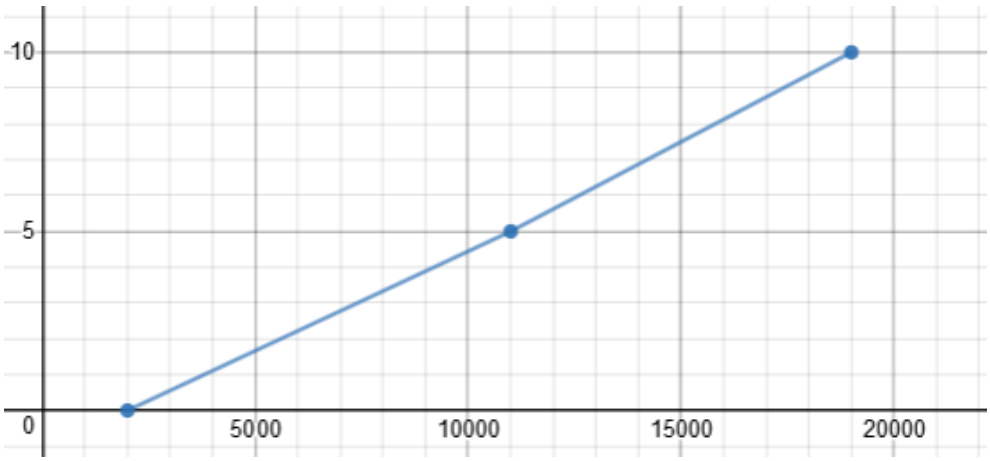

Рисунок 5.2 – X1, швидкодія мови програмування

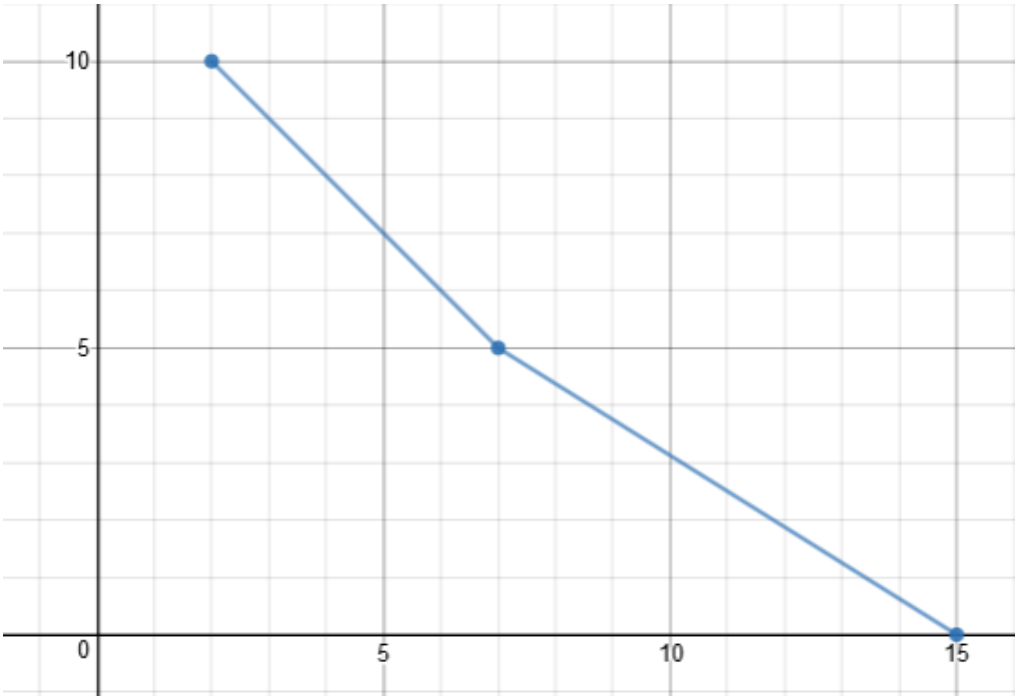

Рисунок 5.3 – X2, зручність доступу для розробника сайту

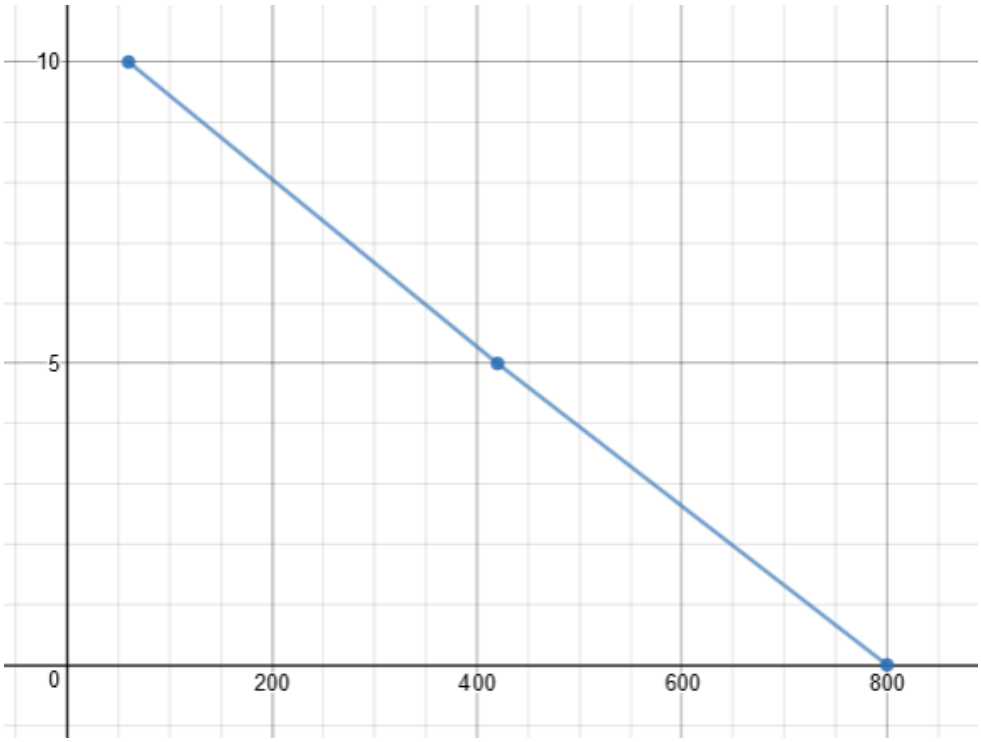

Рисунок 5.4 – X3, час обробки даних алгоритмом

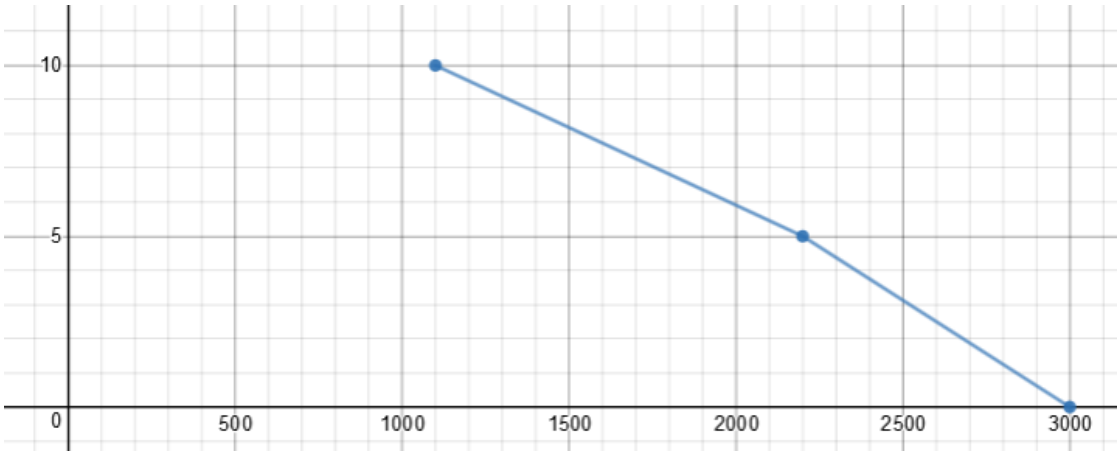

Рисунок 5.5 – X4, потенційний об’єм програмного коду

5.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3. Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Зручність доступу для розробника сайту	Оп	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки даних алгоритмом	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420{,}75}{7^2(5^3 - 5)} = 1{,}03 > W_k = 0{,}67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4.

Таблиця 5.4. Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1{,}0 \text{ при } X_i = X_j \\ 0{,}5 \text{ при } X_i < X_j \end{cases}$$

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i},$$

де $b_i = \sum_{i=1}^{N} a_{ij}$.

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'},$$

де $b_i' = \sum_{i=1}^{N} a_{ij} b_j$.

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5. Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

5.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X3 (*час обробки даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X2 (*зручність доступу для розробника сайту) відповідає або варіанту а) 8 або варіанту б) 4 операції.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j},$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра;B_i – оцінка i–го параметра в балах.

Таблиця 5.6. Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1,X3)	А	11000	3,6	0,215	0,774
F2(X2)	А	15	3,4	0,154	0,524
	Б	7	4,1	0,283	1,1603
F3(X4)	А	800	2,4	0,348	0,835

За даними з таблиці 5.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}],$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,774 + 1,1603 + 0,835 = 2,7693

K_{K2} = 0,774 + 0,524 + 0,835 = 2,133

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М} \quad (5.1)$$

де T_P – трудомісткість розробки ПП;

$K_П$ – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

$K_М$ – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: T_P = 90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_П$ = 1.7. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто T_P = 27 людино-днів, $K_П$ = 0.9, $K_{СК}$ = 1, $K_{СТ}$ = 0.8:

$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328{,}64$ людино-годин;

$T_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 6000 грн., один фінансовий аналітик з окладом 9000грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн.,$$

де М – місячний оклад працівників;T_m – кількість робочих днів тиждень;t – кількість робочих годин в день.

$$С_ч = \frac{6000 + 6000 + 9000}{3 \cdot 21 \cdot 8} = 41{,}67 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д,$$

де $С_ч$– величина погодинної оплати праці програміста;$Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 41,67· 1328.64 · 1.2 = 66437,31 грн.

II. $С_{ЗП}$ = 41,67· 1345.52 · 1.2 = 67281,38 грн.

Відрахування на єдиний соціальний внесок становить 22%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.3677$ = 66437,31· 0.22 = 14616,21 грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.3677$ = 67281,38 · 0.22 = 14801,9 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 6000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З$ = 12 · 6000· 0,2 = 14400 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З)$ = 14400 · (1 + 0.2) = 17280 грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22$ = 17280 · 0,22 = 3801,6 грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 8000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 8000 = 2300$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

$К_А$– річна норма амортизації;

$Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 8000 \cdot 0.05 = 460$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році;

$Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів;

$Д_Р$ – кількість днів планових ремонтів устаткування;

t –кількість робочих годин в день;

$К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot Ц_{ЕН} = 1706,4 \cdot 0,156 \cdot 1,94 = 516,42$ грн.,

де $N_С$ – середньо-споживча потужність приладу;

$Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 8000 \cdot 0,67 = 5360$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС} = 17280 + 3801,6 + 2300 + 460 + 516,42 + 5360 = 29718,02$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС} / Т_{ЕФ} = 29718,02 / 1706,4 = 17,42$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_М = C_{М\text{-}Г} \cdot T$$

I. $C_М = 17{,}42 \cdot 1328{,}64 = 23144{,}91$ грн.;

II. $C_М = 17{,}42 \cdot 1345.52 = 23438{,}96$ грн.;

Накладні витрати складають 67% від заробітної плати:

$$C_Н = C_{ЗП} \cdot 0{,}67$$

I. $C_Н = 66437{,}31 \cdot 0{,}67 = 44513$ грн.;

II. $C_Н = 67281{,}38 \cdot 0{,}67 = 45078{,}52$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$$C_{ПП} = C_{ЗП} + C_{ВІД} + C_М + C_Н$$

I. $C_{ПП} = 66437{,}31 + 14616{,}21 + 23144{,}91 + 44513 = 148711{,}43$ грн.;

II. $C_{ПП} = 67281{,}38 + 14801{,}9 + 23438{,}96 + 45078{,}52 = 150600{,}76$ грн.;

5.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1} = 5{,}214 / 148711{,}43 = 0{,}35 \cdot 10^{-4}$;

$К_{ТЕР2} = 3{,}569 / 150600{,}76 = 0{,}24 \cdot 10^{-4}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1} = 0{,}35 \cdot 10^{-4}$.

5.6 Висновки до розділу

У даному розділі було проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

У першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації;

на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 0,35 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

– мова програмування – Java;

– використання технології веб-сервісів;

– побудова фільтрів методом апроксимації за прототипом.

Даний варіант виконання програмного комплексу дає користувачу непоганий функціонал і швидкодію, а також полегшує роботу розробника інтерфейсу користувача.

ВИСНОВКИ

Дипломна робота присвячена задачі розробки сайту розрахунку цифрових фільтрів. Дана пояснювальна записка є частиною комплексної дипломної роботи та описує підсистему розрахунків, що разом з інтерфейсом користувача складає сайт розрахунку цифрових фільтрів.

У результаті виконання даної дипломної роботи було створено серверний додаток мовою Java, який виконує функції з розрахунку цифрових фільтрів у формі веб-сервісу SOAP. В ході розробки використовувався фреймворк Spring та засіб автоматизації збірки Maven.

Додаток успішно розраховує параметри та характеристики цифрових фільтрів низьких та високих частот, смугових та режекторних фільтрів на основі апроксимації Батерворта, Чебишова та Кауера (еліптичні фільтри). Серед даних, що повертає додаток, є полюси та нулі фільтру, коефіцієнти передаточної функції, АЧХ, ЛАЧХ, ФЧХ, імпульсна та перехідна характеристики.

Створений додаток може використовуватися разом із інтерфейсом користувача для доступу до сайту з точки зору звичайного користувача, або ж як окремий веб-сервіс в ході розробки обчислювальних систем (наприклад, САПР) з використанням засобів, що підтримують протокол SOAP.

Подальше вдосконалення додатку може вестись у наступних напрямах:

- Розрахунок інших типів нерекурсивних фільтрів, наприклад, Чебишова II роду, Бесселя тощо;
- Генерація коду на мові програмування низького рівня (наприклад, C) для швидкого включення цифрових фільтрів у програмний проект користувача;
- Оптимізація обчислень шляхом математичних перетворень деяких формул.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петренко А.И. Основы автоматизации проектирования / Петренко А.И. – К. : Техника, 1982. – 295 с.
2. MicroModeler DSP – Community Edition – Not for commercial use. – Режим доступу: http://www.micromodeler.com/dsp/. – Дата доступу: 12.06.2017.
3. TFilter – Free online FIR filter design. – Режим доступу: http://t-filter.engineerjs.com/. – Дата доступу: 12.06.2017.
4. Interactive Digital Filter Design. – Режим доступу: https://www-users.cs.york.ac.uk/~fisher/mkfilter/. – Дата доступу: 12.06.2017.
5. Digital Filters Applet. – Режим доступу: http://www.falstad.com/dfilter/. – Дата доступу: 12.06.2017.
6. Digital Filter Design. – Режим доступу: http://www.arc.id.au/FilterDesign.html. – Дата доступу: 12.06.2017.
7. Антонью А. Цифровые фильтры: анализ и проектирование: Пер. с англ. / Антонью А. — М.: Радио и связь, 1983. — 320 с, ил.
8. Пелед А., Лиу Б. Цифровая обработка сигналов. / Пелед А., Лиу Б. – Киев: Техника, 1980. – 263 с.

ДОДАТОК А. КОД ПРОГРАМИ

Лістинг файлу «pom.xml»:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>

 <groupId>digifilter</groupId>
 <artifactId>digifilter-server-app</artifactId>
 <version>0.3.0</version>

 <dependencies>
  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>4.3.7.RELEASE</version>
  </dependency>
  <dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-context</artifactId>
   <version>4.3.7.RELEASE</version>
  </dependency>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter</artifactId>
   <version>1.5.2.RELEASE</version>
  </dependency>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
   <version>1.5.2.RELEASE</version>
  </dependency>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web-services</artifactId>
   <version>1.5.2.RELEASE</version>
  </dependency>
  <dependency>
   <groupId>wsdl4j</groupId>
   <artifactId>wsdl4j</artifactId>
   <version>1.6.3</version>
  </dependency>
  <dependency>
   <groupId>org.codehaus.mojo</groupId>
   <artifactId>jaxb2-maven-plugin</artifactId>
   <version>2.3.1</version>
  </dependency>
 </dependencies>

 <properties>
  <java.version>1.8</java.version>
 </properties>

 <build>
  <defaultGoal>spring-boot:run</defaultGoal>
  <plugins>
   <plugin>
    <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-maven-plugin</artifactId>
    <version>1.5.2.RELEASE</version>
    <executions>
     <execution>
      <goals>
       <goal>repackage</goal>
      </goals>
     </execution>
    </executions>
    <configuration>
     <mainClass>calculations.Application</mainClass>
    </configuration>
   </plugin>
   <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>jaxb2-maven-plugin</artifactId>
    <version>2.3.1</version>
    <executions>
     <execution>
      <id>xjc</id>
      <goals>
       <goal>xjc</goal>
      </goals>
      <configuration>
       <sources>
        <source>src/main/resources/iir.xsd</source>
       </sources>
       <outputDirectory>src/main/java</outputDirectory>
       <clearOutputDir>false</clearOutputDir>
       <packageName>autogenerated</packageName>
       <arguments>
        <argument>-Xvalue-constructor</argument>
       </arguments>
      </configuration>
     </execution>
    </executions>
    <dependencies>
     <dependency>
      <groupId>org.jvnet.jaxb2_commons</groupId>
      <artifactId>jaxb2-value-constructor</artifactId>
      <version>3.0</version>
     </dependency>
     <dependency>
      <groupId>org.jvnet.jaxb2_commons</groupId>
      <artifactId>jaxb2-basics</artifactId>
      <version>1.11.1</version>
     </dependency>
    </dependencies>
   </plugin>
   <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.6.1</version>
    <configuration>
     <source>1.7</source>
     <target>1.7</target>
    </configuration>
   </plugin>
  </plugins>
 </build>
</project>
```

Лістинг файлу «iir.xsd»:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="digifilter-
server"
     targetNamespace="digifilter-server" elementFormDefault="qualified">

 <xs:element name="buildIirResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="zeroes" type="tns:complexList"/>
    <xs:element name="poles" type="tns:complexList"/>
    <xs:element name="transferFunction">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="factor" type="xs:double"/>
       <xs:element name="sections" type="tns:sectionList"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="amplitude" type="tns:doubleList"/>
    <xs:element name="logAmplitude" type="tns:doubleList"/>
    <xs:element name="phase" type="tns:doubleList"/>
    <xs:element name="pulse" type="tns:doubleList"/>
    <xs:element name="step" type="tns:doubleList"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:element name="buildIirRequest">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="sampleRate" type="xs:double"/>
    <xs:element name="iirType" type="tns:filterType"/>
    <xs:element name="approxType" type="tns:functionType"/>
    <xs:element name="stopbandFrequencies" type="tns:doubleList"/>
    <xs:element name="passbandFrequencies" type="tns:doubleList"/>
    <xs:element name="passbandRipple" type="xs:double"/>
    <xs:element name="stopbandAttenuation" type="xs:double"/>
    <xs:element name="frequencyStep" type="xs:double"/>
    <xs:element name="timeCounts" type="xs:int"/>
    <xs:element name="filterOrder" type="xs:int"/>
    <xs:element name="requestType" type="tns:requestType"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:complexType name="sectionList">
  <xs:sequence>
   <xs:element maxOccurs="unbounded" name="values" type="tns:section"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="section">
  <xs:sequence>
   <xs:element name="numerator" type="tns:doubleList"/>
   <xs:element name="denominator" type="tns:doubleList"/>
  </xs:sequence>
 </xs:complexType>

 <xs:simpleType name="filterType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="lowPass"/>
   <xs:enumeration value="highPass"/>
```

```
   <xs:enumeration value="bandPass"/>
   <xs:enumeration value="bandStop"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="functionType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="butterworth"/>
   <xs:enumeration value="chebyshev"/>
   <xs:enumeration value="elliptic"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="requestType">
  <xs:restriction base="xs:string">
   <xs:enumeration value="stopband"/>
   <xs:enumeration value="order"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:complexType name="doubleList">
  <xs:sequence>
   <xs:element maxOccurs="unbounded" name="values" type="xs:double"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="complex">
  <xs:sequence>
   <xs:element name="re" type="xs:double"/>
   <xs:element name="im" type="xs:double"/>
  </xs:sequence>
 </xs:complexType>

 <xs:complexType name="complexList">
  <xs:sequence>
   <xs:element maxOccurs="unbounded" name="values" type="tns:complex"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Лістинг файлу «Application.java»:

```
package calculations;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application {
 public static void main(String[] args) {
  SpringApplication.run(Application.class, args);
 }
}
```

Лістинг файлу «FilterBuilder.java»:

```
package calculations;

import autogenerated.*;
```

```
import java.util.*;

public class FilterBuilder {
 private final double stopbandFrequency;
 private final double passbandFrequency;
 private final double stopbandFrequency2;
 private final double passbandFrequency2;
 private final double stopbandAttenuation;
 private final double passbandRipple;
 private double alpha;
 private double beta;
 private int filterOrder;
 private Map<String, Double> additionalParameters = new HashMap<>();

 private final BuildIirRequest requirements;

 public FilterBuilder(BuildIirRequest requirements) {
  this.requirements = requirements;

  passbandFrequency = requirements.getPassbandFrequencies().getValues().get(0) /
requirements.getSampleRate();
  passbandRipple = requirements.getPassbandRipple();
  if ((requirements.getIirType() == FilterType.BAND_PASS) ||
requirements.getIirType() == FilterType.BAND_STOP) {
   passbandFrequency2 = requirements.getPassbandFrequencies().getValues().get(1)
/ requirements.getSampleRate();
  } else {
   passbandFrequency2 = 0.0;
  }

  if ((requirements.getRequestType() == RequestType.STOPBAND) ||
(requirements.getApproxType() == FunctionType.ELLIPTIC)) {
   stopbandFrequency = requirements.getStopbandFrequencies().getValues().get(0)
/ requirements.getSampleRate();
   stopbandAttenuation = -requirements.getStopbandAttenuation();
   if ((requirements.getIirType() == FilterType.BAND_PASS) ||
requirements.getIirType() == FilterType.BAND_STOP) {
    stopbandFrequency2 =
requirements.getStopbandFrequencies().getValues().get(1) /
requirements.getSampleRate();
   } else {
    stopbandFrequency2 = 0.0;
   }
  } else {
   stopbandFrequency = 0.0;
   stopbandAttenuation = 0.0;
   stopbandFrequency2 = 0.0;
  }
 }

 public BuildIirResponse build() throws Exception {
  switch(requirements.getRequestType()) {
   case STOPBAND:
    double wa = prototypeFrequency();
    filterOrder = filterOrder(wa);
    break;
   case ORDER:
    filterOrder = requirements.getFilterOrder();
    switch (requirements.getIirType()) {
     case BAND_PASS:
      alpha = 1.0 / Math.tan(Math.PI * (passbandFrequency2 -
passbandFrequency));
      beta = Math.sin(2 * Math.PI * (passbandFrequency + passbandFrequency2)) /
```

```
        (Math.sin(2 * Math.PI * passbandFrequency) + Math.sin(2 * Math.PI *
passbandFrequency2));
       break;
      case BAND_STOP:
       alpha = (Math.cos(2 * Math.PI * passbandFrequency) - Math.cos(2 * Math.PI
* passbandFrequency2)) / (Math.sin(2 * Math.PI * passbandFrequency) + Math.sin(2
* Math.PI * passbandFrequency2));
       beta = Math.sin(2 * Math.PI * (passbandFrequency + passbandFrequency2)) /
(Math.sin(2 * Math.PI * passbandFrequency) + Math.sin(2 * Math.PI *
passbandFrequency2));
       break;
     }
     if (requirements.getApproxType() == FunctionType.ELLIPTIC) {
      wa = prototypeFrequency();
      double k = 1.0 / wa;
      double kk = Math.sqrt(Math.sqrt(1.0 - k * k));
      double q0 = (1 - kk) / (2 * (1 + kk));
      double q = q0 + 2 * Math.pow(q0, 5) + 15 * Math.pow(q0, 9) + 150 *
Math.pow(q0, 13);
      additionalParameters.put("k", k);
      additionalParameters.put("q", q);
     }
     break;
   }

   FilterZeroPoles zps = digitalize(prototypeZeroPoles());
   BuildIirResponse.TransferFunction tf = transferFunction(zps);

   List<Complex> zeroes = new ArrayList<>();
   List<Complex> poles = new ArrayList<>();

   for (ZPSection sec : zps.getSections()) {
    switch(sec.getZeroType()) {
     case REAL:
      zeroes.add(new Complex(sec.getRealZero(), 0.0));
      break;
     case DOUBLE_REAL:
      zeroes.add(new Complex(sec.getRealZero(), 0.0));
      zeroes.add(new Complex(sec.getRealZero2(), 0.0));
      break;
     case COMPLEX:
      zeroes.add(sec.getComplexZero());
      zeroes.add(new Complex(
        sec.getComplexZero().getRe(),
        -sec.getComplexZero().getIm()
      ));
      break;
    }
    switch(sec.getPoleType()) {
     case REAL:
      poles.add(new Complex(sec.getRealPole(), 0.0));
      break;
     case DOUBLE_REAL:
      poles.add(new Complex(sec.getRealPole(), 0.0));
      poles.add(new Complex(sec.getRealPole2(), 0.0));
      break;
     case COMPLEX:
      poles.add(sec.getComplexPole());
      poles.add(new Complex(
        sec.getComplexPole().getRe(),
        -sec.getComplexPole().getIm()
      ));
      break;
```

```
   }
  }

  return new BuildIirResponse(
    new ComplexList(zeroes),
    new ComplexList(poles),
    tf,
    new DoubleList(amplitude(tf)),
    new DoubleList(logAmplitude(tf)),
    new DoubleList(phase(tf)),
    new DoubleList(pulse(tf)),
    new DoubleList(step(tf))
  );
 }

 private List<Double> logAmplitude(BuildIirResponse.TransferFunction tf) {
  double minFreq = 0.0;
  double maxFreq = 0.5;
  double step = requirements.getFrequencyStep() / requirements.getSampleRate();

  List<Double> res = new ArrayList<>();

  for (double f = minFreq; f <= maxFreq; f+= step) {
   double y = 20 * Math.log10(tf.getFactor());
   for (Section sec : tf.getSections().getValues()) {
    double A1 = sec.getNumerator().getValues().get(1);
    double A2;
    if (sec.getNumerator().getValues().size() > 2) {
     A2 = sec.getNumerator().getValues().get(2);
    } else {
     A2 = 0.0;
    }
    double B1 = sec.getDenominator().getValues().get(1);
    double B2;
    if (sec.getDenominator().getValues().size() > 2) {
     B2 = sec.getDenominator().getValues().get(2);
    } else {
     B2 = 0.0;
    }

    y += 20 * Math.log10(Math.sqrt(
      (
        Math.pow(1 + A1 * Math.cos(2 * Math.PI * f) + A2 * Math.cos(4 * Math.PI
* f), 2) +
          Math.pow(A1 * Math.sin(2 * Math.PI * f) + A2 * Math.sin(4 * Math.PI *
f), 2)
      ) / (
        Math.pow(1 + B1 * Math.cos(2 * Math.PI * f) + B2 * Math.cos(4 * Math.PI
* f), 2) +
          Math.pow(B1 * Math.sin(2 * Math.PI * f) + B2 * Math.sin(4 * Math.PI *
f), 2)
      )
    ));
   }
   res.add(y);
  }

  return res;
 }

 private List<Double> amplitude(BuildIirResponse.TransferFunction tf) {
  double minFreq = 0.0;
  double maxFreq = 0.5;
```

```
  double step = requirements.getFrequencyStep() / requirements.getSampleRate();

  List<Double> res = new ArrayList<>();

  for (double f = minFreq; f <= maxFreq; f+= step) {
   double y = tf.getFactor();
   for (Section sec : tf.getSections().getValues()) {
    double A1 = sec.getNumerator().getValues().get(1);
    double A2;
    if (sec.getNumerator().getValues().size() > 2) {
     A2 = sec.getNumerator().getValues().get(2);
    } else {
     A2 = 0.0;
    }
    double B1 = sec.getDenominator().getValues().get(1);
    double B2;
    if (sec.getDenominator().getValues().size() > 2) {
     B2 = sec.getDenominator().getValues().get(2);
    } else {
     B2 = 0.0;
    }

    y *= Math.sqrt(
      (
        Math.pow(1 + A1 * Math.cos(2 * Math.PI * f) + A2 * Math.cos(4 * Math.PI
* f), 2) +
        Math.pow(A1 * Math.sin(2 * Math.PI * f) + A2 * Math.sin(4 * Math.PI *
f), 2)
      ) / (
        Math.pow(1 + B1 * Math.cos(2 * Math.PI * f) + B2 * Math.cos(4 * Math.PI
* f), 2) +
        Math.pow(B1 * Math.sin(2 * Math.PI * f) + B2 * Math.sin(4 * Math.PI *
f), 2)
      )
    );
   }
   res.add(y);
  }

  return res;
 }

 private List<Double> phase(BuildIirResponse.TransferFunction tf) {
  double minFreq = 0.0;
  double maxFreq = 0.5;
  double step = requirements.getFrequencyStep() / requirements.getSampleRate();

  List<Double> res = new ArrayList<>();

  for (double f = minFreq; f <= maxFreq; f+= step) {
   double y = 0.0;
   for (Section sec : tf.getSections().getValues()) {
    double A1 = sec.getNumerator().getValues().get(1);
    double A2;
    if (sec.getNumerator().getValues().size() > 2) {
     A2 = sec.getNumerator().getValues().get(2);
    } else {
     A2 = 0.0;
    }
    double B1 = sec.getDenominator().getValues().get(1);
    double B2;
    if (sec.getDenominator().getValues().size() > 2) {
     B2 = sec.getDenominator().getValues().get(2);
```

```
    } else {
     B2 = 0.0;
    }

    y += Math.atan2(
      B1 * Math.sin(2 * Math.PI * f) + B2 * Math.sin(4 * Math.PI * f),
      1 + B1 * Math.cos(2 * Math.PI * f) + B2 * Math.cos(4 * Math.PI * f)
    ) - Math.atan2(
      A1 * Math.sin(2 * Math.PI * f) + A2 * Math.sin(4 * Math.PI * f),
      1 + A1 * Math.cos(2 * Math.PI * f) + A2 * Math.cos(4 * Math.PI * f)
    );
   }
   res.add(y);
  }

  return res;
 }

 private List<Double> pulse(BuildIirResponse.TransferFunction tf) {
  List<Double> res = new ArrayList<>();
  List<Section> sec = tf.getSections().getValues();

  double[] x = new double[sec.size() + 1];
  double[] x1 = new double[sec.size() + 1];
  double[] x2 = new double[sec.size() + 1];

  x[0] = 1.0;
  x1[0] = 0.0;
  x2[0] = 0.0;

  for (int i = 0; i < requirements.getTimeCounts(); i++) {
   for (int j = 0; j < sec.size(); j++) {
    double A1 = sec.get(j).getNumerator().getValues().get(1);
    double A2;
    if (sec.get(j).getNumerator().getValues().size() > 2) {
     A2 = sec.get(j).getNumerator().getValues().get(2);
    } else {
     A2 = 0.0;
    }
    double B1 = sec.get(j).getDenominator().getValues().get(1);
    double B2;
    if (sec.get(j).getDenominator().getValues().size() > 2) {
     B2 = sec.get(j).getDenominator().getValues().get(2);
    } else {
     B2 = 0.0;
    }
    x[j + 1] = x[j] + A1 * x1[j] + A2 * x2[j] - B1 * x1[j + 1] - B2 * x2[j + 1];
   }
   double y = x[sec.size()] * tf.getFactor();
   res.add(y);
   x2 = x1;
   x1 = x;
   x = new double[sec.size() + 1];
   x[0] = 0.0;
  }

  return res;
 }

 private List<Double> step(BuildIirResponse.TransferFunction tf) {
  List<Double> res = new ArrayList<>();
  List<Section> sec = tf.getSections().getValues();
```

```
  double[] x = new double[sec.size() + 1];
  double[] x1 = new double[sec.size() + 1];
  double[] x2 = new double[sec.size() + 1];

  x[0] = 1.0;
  x1[0] = 0.0;
  x2[0] = 0.0;

  for (int i = 0; i < requirements.getTimeCounts(); i++) {
   for (int j = 0; j < sec.size(); j++) {
    double A1 = sec.get(j).getNumerator().getValues().get(1);
    double A2;
    if (sec.get(j).getNumerator().getValues().size() > 2) {
     A2 = sec.get(j).getNumerator().getValues().get(2);
    } else {
     A2 = 0.0;
    }
    double B1 = sec.get(j).getDenominator().getValues().get(1);
    double B2;
    if (sec.get(j).getDenominator().getValues().size() > 2) {
     B2 = sec.get(j).getDenominator().getValues().get(2);
    } else {
     B2 = 0.0;
    }
    x[j + 1] = x[j] + A1 * x1[j] + A2 * x2[j] - B1 * x1[j + 1] - B2 * x2[j + 1];
   }
   double y = x[sec.size()] * tf.getFactor();
   res.add(y);
   x2 = x1;
   x1 = x;
   x = new double[sec.size() + 1];
   x[0] = 1.0;
  }

  return res;
 }

 private BuildIirResponse.TransferFunction transferFunction(FilterZeroPoles zps)
throws Exception {
  BuildIirResponse.TransferFunction res = new
BuildIirResponse.TransferFunction(zps.getFactor(),
    new SectionList(new ArrayList<Section>()));

  List<Section> sections = res.getSections().getValues();

  for (ZPSection zp : zps.getSections()) {
   Section sec = new Section();
   switch(zp.getZeroType()) {
    case REAL:
     sec.setNumerator(new DoubleList(Arrays.asList(1.0, -zp.getRealZero())));
     break;
    case DOUBLE_REAL:
     sec.setNumerator(new DoubleList(Arrays.asList(1.0, -(zp.getRealZero() +
zp.getRealZero2()),
       zp.getRealZero() * zp.getRealZero2())));
     break;
    case COMPLEX:
     double x = zp.getComplexZero().getRe();
     double y = zp.getComplexZero().getIm();
     sec.setNumerator(new DoubleList(Arrays.asList(1.0, -2 * x, x * x + y *
y)));
     break;
    case NONE:
```

```
     sec.setNumerator(new DoubleList(Collections.singletonList(1.0)));
     break;
    default:
     throw new Exception("Unhandled zero type");
   }
   switch(zp.getPoleType()) {
    case REAL:
     sec.setDenominator(new DoubleList(Arrays.asList(1.0, -zp.getRealPole())));
     break;
    case DOUBLE_REAL:
     sec.setDenominator(new DoubleList(Arrays.asList(1.0, -(zp.getRealPole() +
zp.getRealPole2()),
       zp.getRealPole() * zp.getRealPole2())));
     break;
    case COMPLEX:
     double x = zp.getComplexPole().getRe();
     double y = zp.getComplexPole().getIm();
     sec.setDenominator(new DoubleList(Arrays.asList(1.0, -2 * x, x * x + y *
y)));
     break;
    case NONE:
     sec.setDenominator(new DoubleList(Collections.singletonList(1.0)));
     break;
    default:
     throw new Exception("Unhandled pole type");
   }

   sections.add(sec);
  }

  return res;
 }

 private FilterZeroPoles digitalize(FilterZeroPoles prototypeSections) {
  List<ZPSection> res = new ArrayList<>();
  double factor = prototypeSections.getFactor();

  for (ZPSection sec : prototypeSections.getSections()) {
   switch(sec.getPoleType()) {
    case REAL:
     switch(sec.getZeroType()) {
      case NONE:
       double c, D;
       Double s = sec.getRealPole();

       switch(requirements.getIirType()) {
        case LOW_PASS:
         c = 1.0 / Math.tan(Math.PI * passbandFrequency);
         res.add(new ZPSection(-1.0, (c + s) / (c - s)));
         factor /= c - s;
         break;
        case HIGH_PASS:
         c = Math.tan(Math.PI * passbandFrequency);
         res.add(new ZPSection(1.0, (s + c) / (s - c)));
         factor /= c - s;
         break;
        case BAND_PASS:
         D = Math.pow(alpha * beta, 2) - Math.pow(alpha, 2) + Math.pow(s, 2);
         if (D >= 0) {
          double d = Math.sqrt(D);
          res.add(new ZPSection(
            1.0, -1.0,
            (s + alpha) / (alpha * beta + d),
```

```
            (s + alpha) / (alpha * beta - d)
           ));
          } else {
           double d = Math.sqrt(-D);
           res.add(new ZPSection(
             1.0, -1.0,
             new Complex(
               alpha * beta * (s + alpha) / (alpha * alpha * beta * beta + d *
d),
               d * (s + alpha) / (alpha * alpha * beta * beta + d * d)
             )
           ));
          }
          factor /= alpha - s;
          break;
         case BAND_STOP:
          D = Math.pow(s * beta, 2) - Math.pow(s, 2) + Math.pow(alpha, 2);
          if (D >= 0) {
           double d = Math.sqrt(D);
           res.add(new ZPSection(
             new Complex(beta, Math.sqrt(1 - beta * beta)),
             (s + alpha) / (s * beta + d),
             (s + alpha) / (s * beta - d)
           ));
          } else {
           double d = Math.sqrt(-D);
           res.add(new ZPSection(
             new Complex(beta, Math.sqrt(1 - beta * beta)),
             new Complex(
               s * beta * (s + alpha) / (s * s * beta * beta + d * d),
               d * (s + alpha) / (s * s * beta * beta + d * d)
             )
           ));
          }
          factor /= alpha - s;
          break;
         default:
          throw new IllegalArgumentException("Illegal filter type");
        }
        break;
      }
      break;
     case COMPLEX:
      switch(sec.getZeroType()) {
       case NONE:
        double c, d;
        double x = sec.getComplexPole().getRe();
        double y = sec.getComplexPole().getIm();

        switch(requirements.getIirType()) {
         case LOW_PASS:
          c = 1.0 / Math.tan(Math.PI * passbandFrequency);
          d = ((c + x) * (c + x) + y * y);
          res.add(new ZPSection(new Complex(-1.0, 0.0), new Complex(
            (c * c - x * x - y * y) / d,
            -2 * c * y / d
          )));
          factor /= x * x + y * y + 2 * x * c + c * c;
          break;
         case HIGH_PASS:
          c = Math.tan(Math.PI * passbandFrequency);
          d = (Math.pow(c + x, 2) + y * y);
          res.add(new ZPSection(new Complex(1.0, 0.0), new Complex(
```

```
            (-c * c + x * x + y * y) / d,
            2 * c * y / d
          )));
          factor /= x * x + y * y + 2 * x * c + c * c;
          break;
         case BAND_PASS:
          Complex D = new Complex(
            Math.pow(alpha * beta, 2) - Math.pow(alpha, 2) + Math.pow(x, 2) -
Math.pow(y, 2),
            2 * x * y
          );
          Complex dq = sqrt(D);
          double xq = dq.getRe();
          double yq = dq.getIm();
          double p = Math.pow(alpha + x, 2) + Math.pow(y, 2);
          res.add(new ZPSection(new Complex(1.0, 0.0), new Complex(
            (alpha * alpha * beta + alpha * beta * x + xq * alpha + xq * x + yq *
y) / p,
            (-alpha * beta * y - xq * y + alpha * yq + x * yq) / p
          )));
          res.add(new ZPSection(new Complex(-1.0, 0.0), new Complex(
            (alpha * alpha * beta + alpha * beta * x - xq * alpha - xq * x - yq *
y) / p,
            (-alpha * beta * y + xq * y - alpha * yq - x * yq) / p
          )));
          factor /= (x * x + y * y + 2 * x * alpha + alpha * alpha);
          break;
         case BAND_STOP:
          D = new Complex(
            (x * x - y * y) * (beta * beta - 1) + alpha * alpha,
            2 * x * y * (beta * beta - 1)
          );
          dq = sqrt(D);
          xq = dq.getRe();
          yq = dq.getIm();
          p = Math.pow(alpha + x, 2) + Math.pow(y, 2);
          res.add(new ZPSection(new Complex(beta, Math.sqrt(1 - beta * beta)),
new Complex(
            (alpha * beta * x + beta * x * x + alpha * xq + x * xq + beta * y * y
+ y * yq) / p,
            (alpha * beta * y - xq * y + x * yq + alpha * yq) / p
          )));
          res.add(new ZPSection(new Complex(beta, Math.sqrt(1 - beta * beta)),
new Complex(
            (alpha * beta * x + beta * x * x - alpha * xq - x * xq + beta * y * y
- y * yq) / p,
            (alpha * beta * y + xq * y - x * yq - alpha * yq) / p
          )));
          factor /= (x * x + y * y + 2 * x * alpha + alpha * alpha);
          break;
         default:
          throw new IllegalArgumentException("Illegal filter type");
        }
        break;
       case COMPLEX:
        double d2;
        x = sec.getComplexPole().getRe();
        y = sec.getComplexPole().getIm();
        double x2 = sec.getComplexZero().getRe();
        double y2 = sec.getComplexZero().getIm();

        switch(requirements.getIirType()) {
         case LOW_PASS:
```

```
          c = 1.0 / Math.tan(Math.PI * passbandFrequency);
          d = (Math.pow(c + x, 2) + y * y);
          d2 = (Math.pow(c + x2, 2) + y2 * y2);
          res.add(new ZPSection(
            new Complex(
              (c * c - x2 * x2 - y2 * y2) / d2,
              2 * c * y2 / d2
            ),
            new Complex(
              (c * c - x * x - y * y) / d,
              2 * c * y / d
            )
          ));
          factor *= (x2 * x2 + y2 * y2 + 2 * x2 * c + c * c) / (x * x + y * y + 2
* x * c + c * c);
          break;
        case HIGH_PASS:
          c = Math.tan(Math.PI * passbandFrequency);
          d = (Math.pow(c + x, 2) + y * y);
          d2 = (Math.pow(c + x2, 2) + y2 * y2);
          res.add(new ZPSection(
            new Complex(
              -(c * c - x2 * x2 - y2 * y2) / d2,
              -2 * c * y2 / d2
            ),
            new Complex(
              -(c * c - x * x - y * y) / d,
              -2 * c * y / d
            )
          ));
          factor *= (x2 * x2 + y2 * y2 + 2 * x2 * c + c * c) / (x * x + y * y + 2
* x * c + c * c);
          break;
        case BAND_PASS:
          Complex D = new Complex(
            Math.pow(alpha * beta, 2) - Math.pow(alpha, 2) + Math.pow(x, 2) -
Math.pow(y, 2),
            2 * x * y
          );
          Complex dq = sqrt(D);
          double xq = dq.getRe();
          double yq = dq.getIm();
          double p = Math.pow(alpha + x, 2) + Math.pow(y, 2);
          Complex D2 = new Complex(
            Math.pow(alpha * beta, 2) - Math.pow(alpha, 2) + Math.pow(x2, 2) -
Math.pow(y2, 2),
            2 * x2 * y2
          );
          Complex dq2 = sqrt(D2);
          double xq2 = dq2.getRe();
          double yq2 = dq2.getIm();
          double p2 = Math.pow(alpha + x2, 2) + Math.pow(y2, 2);
          res.add(new ZPSection(
            new Complex(
              (alpha * alpha * beta + alpha * beta * x2 + xq2 * alpha + xq2 * x2
+ yq2 * y2) / p2,
              (-alpha * beta * y2 - xq2 * y2 + alpha * yq2 + x2 * yq2) / p2
            ), new Complex(
              (alpha * alpha * beta + alpha * beta * x + xq * alpha + xq * x + yq
* y) / p,
              (-alpha * beta * y - xq * y + alpha * yq + x * yq) / p
            )
          ));
```

```
          res.add(new ZPSection(
            new Complex(
              (alpha * alpha * beta + alpha * beta * x2 - xq2 * alpha - xq2 * x2
- yq2 * y2) / p2,
              (-alpha * beta * y2 + xq2 * y2 - alpha * yq2 - x2 * yq2) / p2
            ), new Complex(
              (alpha * alpha * beta + alpha * beta * x - xq * alpha - xq * x - yq
* y) / p,
              (-alpha * beta * y + xq * y - alpha * yq - x * yq) / p
            )
          ));
          factor *= (x2 * x2 + y2 * y2 + 2 * x2 * alpha + alpha * alpha) /
            (x * x + y * y + 2 * x * alpha + alpha * alpha);
          break;
         case BAND_STOP:
          D = new Complex(
            (x * x - y * y) * (beta * beta - 1) + alpha * alpha,
            2 * x * y * (beta * beta - 1)
          );
          dq = sqrt(D);
          xq = dq.getRe();
          yq = dq.getIm();
          p = Math.pow(alpha + x, 2) + Math.pow(y, 2);
          D2 = new Complex(
            (x2 * x2 - y2 * y2) * (beta * beta - 1) + alpha * alpha,
            2 * x2 * y2 * (beta * beta - 1)
          );
          dq2 = sqrt(D2);
          xq2 = dq2.getRe();
          yq2 = dq2.getIm();
          p2 = Math.pow(alpha + x2, 2) + Math.pow(y2, 2);
          res.add(new ZPSection(
            new Complex(
              (alpha * beta * x2 + beta * x2 * x2 + alpha * xq2 + x2 * xq2 + beta
* y2 * y2 + y2 * yq2) / p2,
              (alpha * beta * y2 - xq2 * y2 + x2 * yq2 + alpha * yq2) / p2
            ), new Complex(
              (alpha * beta * x + beta * x * x + alpha * xq + x * xq + beta * y *
y + y * yq) / p,
              (alpha * beta * y - xq * y + x * yq + alpha * yq) / p
            )
          ));
          res.add(new ZPSection(
            new Complex(
              (alpha * beta * x2 + beta * x2 * x2 - alpha * xq2 - x2 * xq2 + beta
* y2 * y2 - y2 * yq2) / p2,
              (alpha * beta * y2 + xq2 * y2 - x2 * yq2 - alpha * yq2) / p2
            ), new Complex(
              (alpha * beta * x + beta * x * x - alpha * xq - x * xq + beta * y *
y - y * yq) / p,
              (alpha * beta * y + xq * y - x * yq - alpha * yq) / p
            )
          ));
          factor *= (x2 * x2 + y2 * y2 + 2 * x2 * alpha + alpha * alpha) /
            (x * x + y * y + 2 * x * alpha + alpha * alpha);
          break;
         default:
          throw new IllegalArgumentException("Illegal filter type");
        }
        break;
      }
      break;
    }
```

```
  }

  return new FilterZeroPoles(res, factor);
 }

 private Complex sqrt(Complex z) {
  double a = z.getRe();
  double b = z.getIm();
  return new Complex(
    Math.sqrt((Math.sqrt(a * a + b * b) + a) / 2),
    (b == 0 ? 1 : Math.signum(b)) * Math.sqrt((Math.sqrt(a * a + b * b) - a) /
2)
  );
 }

 private int checkOrderParity(int odd, int even) {
  return (filterOrder % 2 != 0 ? odd : even);
 }

 private FilterZeroPoles prototypeZeroPoles() {
  List<ZPSection> zps = new ArrayList<>();
  double factor = 1.0;

  double sigma0;
  double Omega0;
  double Lambda;

  switch (requirements.getApproxType()) {
   case BUTTERWORTH:
    sigma0 = Math.pow(Math.pow(10, passbandRipple / 10) - 1, -1.0 / (2 *
filterOrder));
    if (filterOrder % 2 != 0) {
     factor = sigma0;
     zps.add(new ZPSection(-sigma0));
    }
    for (int m = 1; m <= checkOrderParity((filterOrder - 1) / 2,
      filterOrder / 2); m++) {
     double re = sigma0 * Math.sin(Math.PI * (2 * m + checkOrderParity(1, -1)) /
(2 * filterOrder));
     double im = sigma0 * Math.cos(Math.PI * (2 * m + checkOrderParity(1, -1)) /
(2 * filterOrder));
     zps.add(new ZPSection(new Complex(re, im)));
     factor *= (re * re + im * im);
    }
    break;
   case CHEBYSHEV:
    Lambda = Math.log((Math.pow(10, passbandRipple / 20) + 1) / (Math.pow(10,
passbandRipple / 20) - 1))
      / (2 * filterOrder);
    sigma0 = Math.sinh(Lambda);
    Omega0 = Math.cosh(Lambda);
    if (filterOrder % 2 != 0) {
     factor = sigma0;
     zps.add(new ZPSection(-sigma0));
    } else {
     factor = Math.pow(10, -passbandRipple / 20);
    }
    for (int m = 1; m <= checkOrderParity((filterOrder - 1) / 2,
      filterOrder / 2); m++) {
     double re = sigma0 * Math.sin(Math.PI * (2 * m - 1) / (2 * filterOrder));
     double im = Omega0 * Math.cos(Math.PI * (2 * m - 1) / (2 * filterOrder));
     zps.add(new ZPSection(new Complex(re, im)));
     factor *= re * re + im * im;
```

```
    }
    break;
   case ELLIPTIC:
    double k = additionalParameters.get("k");
    double q = additionalParameters.get("q");
    Lambda = Math.log((Math.pow(10, passbandRipple / 20) + 1) / (Math.pow(10,
passbandRipple / 20) - 1))
      / (2 * filterOrder);
    double f = 2 * Math.pow(q, 0.25);
    sigma0 = f * (Math.sinh(Lambda) - q * q * Math.sinh(3 * Lambda)) / (1 - 2 *
q * Math.cosh(2 * Lambda));
    double W = Math.sqrt((1 + k * sigma0 * sigma0) * (1 + sigma0 * sigma0 / k) /
k);
    if (filterOrder % 2 != 0) {
     factor = sigma0 / Math.sqrt(k);
     zps.add(new ZPSection(-sigma0 / Math.sqrt(k)));
    } else {
     factor = Math.pow(10, -passbandRipple / 20);
    }
    for (int m = 1; m <= checkOrderParity((filterOrder - 1) / 2,
      filterOrder / 2); m++) {
     double l;
     if (filterOrder % 2 != 0) {
      l = (double)m;
     } else {
      l = m - .5;
     }

     double p = Math.PI * l / filterOrder;
     double Omega = f * (Math.sin(p) - q * q * Math.sin(3 * p)) / (1 - 2 * q *
Math.cos(2 * p));
     double V = Math.sqrt((1 - k * Omega * Omega) * (1 - Omega * Omega / k) /
k);
     double re = sigma0 * V / (1 + Math.pow(sigma0 * Omega, 2));
     double im = Omega * W / (1 + Math.pow(sigma0 * Omega, 2));

     zps.add(new ZPSection(new Complex(0.0, 1.0 / (Omega * Math.sqrt(k))), new
Complex(re, im)));

     factor *= (re * re + im * im) * (Omega * Omega * k);
    }
    break;
   default:
    throw new IllegalArgumentException("Illegal approximation type");
  }

  return new FilterZeroPoles(zps, factor);
 }

 private int filterOrder(double wa) {
  double D = (Math.pow(10, stopbandAttenuation / 10) - 1) / (Math.pow(10,
passbandRipple / 10) - 1);

  switch (requirements.getApproxType()) {
   case BUTTERWORTH:
    return ((int) Math.ceil(Math.log(D) / (2 * Math.log(wa))));
   case CHEBYSHEV:
    return ((int) Math.ceil(Math.log(4 * D) / (2 * Math.log(wa + Math.sqrt(wa *
wa - 1)))));
   case ELLIPTIC:
    double k = 1.0 / wa;
    double kk = Math.sqrt(Math.sqrt(1.0 - k * k));
    double q0 = (1 - kk) / (2 * (1 + kk));
```

```
    double q = q0 + 2 * Math.pow(q0, 5) + 15 * Math.pow(q0, 9) + 150 *
Math.pow(q0, 13);
    additionalParameters.put("k", k);
    additionalParameters.put("q", q);
    return ((int) Math.ceil(Math.log(16 * D) / Math.log(1.0 / q)));
   default:
    throw new IllegalArgumentException("Illegal approximation type");
  }
 }

 private double prototypeFrequency() {
  double wa;
  double max1, max2;

  switch (requirements.getIirType()) {
   case LOW_PASS:
    wa = Math.tan(Math.PI * stopbandFrequency) / Math.tan(Math.PI *
passbandFrequency);
    break;
   case HIGH_PASS:
    wa =  Math.tan(Math.PI * passbandFrequency) / Math.tan(Math.PI *
stopbandFrequency);
    break;
   case BAND_PASS:
    alpha = 1.0 / Math.tan(Math.PI * (passbandFrequency2 - passbandFrequency));
    beta = Math.sin(2 * Math.PI * (passbandFrequency + passbandFrequency2)) /
      (Math.sin(2 * Math.PI * passbandFrequency) + Math.sin(2 * Math.PI *
passbandFrequency2));
    max1 = -alpha * bandFilterHelper(stopbandFrequency, beta);
    max2 = alpha * bandFilterHelper(stopbandFrequency2, beta);
    wa = (max1 > max2 ? max2 : max1);
    break;
   case BAND_STOP:
    alpha = (Math.cos(2 * Math.PI * passbandFrequency) - Math.cos(2 * Math.PI *
passbandFrequency2)) / (Math.sin(2 * Math.PI * passbandFrequency) + Math.sin(2 *
Math.PI * passbandFrequency2));
    beta = Math.sin(2 * Math.PI * (passbandFrequency + passbandFrequency2)) /
(Math.sin(2 * Math.PI * passbandFrequency) + Math.sin(2 * Math.PI *
passbandFrequency2));
    max1 = -alpha / bandFilterHelper(stopbandFrequency, beta);
    max2 = alpha / bandFilterHelper(stopbandFrequency2, beta);
    wa = (max1 > max2 ? max2 : max1);
    break;
   default:
    throw new IllegalArgumentException("Illegal filter type");
  }

  return wa;
 }

 private static double bandFilterHelper(double f, double beta) {
  return (beta - Math.cos(2 * Math.PI * f)) / Math.sin(2 * Math.PI * f);
 }
}
```

Лістинг файлу «FilterEndpoint.java»:

```
package calculations;

import autogenerated.BuildIirRequest;
import autogenerated.BuildIirResponse;
import autogenerated.DoubleList;
```

```
import org.springframework.ws.client.WebServiceFaultException;
import org.springframework.ws.server.endpoint.annotation.Endpoint;
import org.springframework.ws.server.endpoint.annotation.PayloadRoot;
import org.springframework.ws.server.endpoint.annotation.RequestPayload;
import org.springframework.ws.server.endpoint.annotation.ResponsePayload;

@Endpoint
public class FilterEndpoint {
    @PayloadRoot(namespace = "digifilter-server", localPart = "buildIirRequest")
    @ResponsePayload
    public BuildIirResponse buildIir(@RequestPayload BuildIirRequest request) {
        try {
            return new FilterBuilder(request).build();
        } catch (Exception e) {
            e.printStackTrace();
            throw new WebServiceFaultException(e.getClass().getName() + ": " +
e.getMessage());
        }
    }
}
```

Лістинг файлу «FilterZeroPoles.java»:

```
package calculations;

import java.util.List;

public class FilterZeroPoles {
    private List<ZPSection> sections;
    private double factor;

    public FilterZeroPoles(List<ZPSection> sections, double factor) {
        this.sections = sections;
        this.factor = factor;
    }

    public List<ZPSection> getSections() {
        return sections;
    }

    public double getFactor() {
        return factor;
    }
}
```

Лістинг файлу «SectionType.java»:

```
package calculations;

public enum SectionType {
    REAL,
    COMPLEX,
    NONE,
    DOUBLE_REAL
}
```

Лістинг файлу «WebServiceConfig.java»:

```
package calculations;
```

```
import org.springframework.boot.web.servlet.ServletRegistrationBean;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.core.io.ClassPathResource;
import org.springframework.ws.config.annotation.EnableWs;
import org.springframework.ws.config.annotation.WsConfigurerAdapter;
import org.springframework.ws.transport.http.MessageDispatcherServlet;
import org.springframework.ws.wsdl.wsdl11.DefaultWsdl11Definition;
import org.springframework.xml.xsd.SimpleXsdSchema;
import org.springframework.xml.xsd.XsdSchema;

@EnableWs
@Configuration
public class WebServiceConfig extends WsConfigurerAdapter {
    @Bean
    public ServletRegistrationBean messageDispatcherServlet(ApplicationContext
applicationContext) {
        MessageDispatcherServlet servlet = new MessageDispatcherServlet();
        servlet.setApplicationContext(applicationContext);
        servlet.setTransformWsdlLocations(true);
        return new ServletRegistrationBean(servlet, "/iir/*");
    }

    @Bean(name = "iir")
    public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema iirSchema)
{
        DefaultWsdl11Definition wsdl11Definition = new
DefaultWsdl11Definition();
        wsdl11Definition.setPortTypeName("FiltersPort");
        wsdl11Definition.setLocationUri("/iir");
        wsdl11Definition.setTargetNamespace("digifilter-server");
        wsdl11Definition.setSchema(iirSchema);
        return wsdl11Definition;
    }

    @Bean
    public XsdSchema iirSchema() {
        return new SimpleXsdSchema(new ClassPathResource("iir.xsd"));
    }
}
```

Лістинг файлу «ZPSection.java»:

```
package calculations;

import autogenerated.Complex;

public class ZPSection {
    private Double realZero;
    private Double realZero2;
    private Double realPole;
    private Double realPole2;
    private Complex complexZero;
    private Complex complexPole;
    private final SectionType zeroType;
    private final SectionType poleType;

    public ZPSection(Double realPole) {
        this.realPole = realPole;
        this.zeroType = SectionType.NONE;
```

```
        this.poleType = SectionType.REAL;
    }

    public ZPSection(Complex complexPole) {
        this.complexPole = complexPole;
        this.zeroType = SectionType.NONE;
        this.poleType = SectionType.COMPLEX;
    }


    public ZPSection(Double realZero, Double realPole) {
        this.realZero = realZero;
        this.realPole = realPole;
        this.zeroType = SectionType.REAL;
        this.poleType = SectionType.REAL;
    }

    public ZPSection(Complex complexZero, Complex complexPole) {
        this.complexZero = complexZero;
        this.complexPole = complexPole;
        this.zeroType = SectionType.COMPLEX;
        this.poleType = SectionType.COMPLEX;
    }

    public ZPSection(Double realZero, Double realZero2, Complex complexPole) {
        this.realZero = realZero;
        this.realZero2 = realZero2;
        this.complexPole = complexPole;
        this.zeroType = SectionType.DOUBLE_REAL;
        this.poleType = SectionType.COMPLEX;
    }

    public ZPSection(Double realZero, Double realZero2, Double realPole, Double
realPole2) {
        this.realZero = realZero;
        this.realZero2 = realZero2;
        this.realPole = realPole;
        this.realPole2 = realPole2;
        this.zeroType = SectionType.DOUBLE_REAL;
        this.poleType = SectionType.DOUBLE_REAL;
    }

    public ZPSection(Complex complexZero, Double realPole, Double realPole2) {
        this.realPole = realPole;
        this.realPole2 = realPole2;
        this.complexZero = complexZero;
        this.zeroType = SectionType.COMPLEX;
        this.poleType = SectionType.DOUBLE_REAL;
    }

    public SectionType getZeroType() {        return zeroType;    }

    public SectionType getPoleType() {        return poleType;    }

    public Double getRealZero() {        return realZero;    }

    public Double getRealPole() {        return realPole;    }

    public Complex getComplexZero() {        return complexZero;    }

    public Complex getComplexPole() {        return complexPole;    }

    public Double getRealZero2() {        return realZero2;    }
```

```
    public Double getRealPole2() {        return realPole2;    }
}
```