Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет «Запорізька політехніка»

Факультет інформаційної безпеки та електронних комунікацій
(повне найменування інституту, назва факультету)

Кафедра інформаційної безпеки та наноелектроніки
(повна назва кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
магістр
(ступінь вищої освіти)

на тему Дослідження і реалізація квантового вейвлет-перетворення Добеші
(назва теми)

Виконала: студентка 2 курсу, групи БК-813м
Спеціальності 125 Кібербезпека та захист інформації
(код і найменування спеціальності)
Освітня програма (спеціалізація)
Безпека інформаційних і комунікаційних систем
ЗАЙЦЕВА А.О.
(ПРІЗВИЩЕ та ініціали)

Керівник НЕЛАСА Г.В.
(ПРІЗВИЩЕ та ініціали)

Рецензент САМОЙЛИК С.С.
(ПРІЗВИЩЕ та ініціали)

2024

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Інформаційної безпеки та електронних комунікацій
Кафедра Інформаційна безпека та наноелектроніка
Ступінь вищої освіти магістр
Спеціальність 125 Кібербезпека та захист інформації
(код і найменування)
Освітня програма (спеціалізація) безпека інформаційних і комунікаційних систем
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІБтаН, к. ф.-м. н., доцент
_____ Андрій КОРОТУН
“____” _____2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТКИ

ЗАЙЦЕВОЇ Аніти Олександрівни
(ПРІЗВИЩЕ, ім’я, по батькові)

1. Тема проєкту (роботи) Дослідження і реалізація квантового вейвлет-перетворення Добеші (Research and implementation of the Daubechies quantum wavelet transform)

керівник проєкту (роботи) к.т.н., доцент НЕЛАСА Ганна Вікторівна,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім’я, по батькові)
затверджені наказом закладу вищої освіти від “05” грудня 2024 року №507

2. Строк подання студентом проєкту (роботи) 24 грудня 2024

3. Вихідні дані до проєкту (роботи) реалізувати квантове вейвлет-перетворення Добеші для обробки сигналу та порівняти з квантовим перетворенням Фур’є

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Теорія швидкого перетворення Фур'є. Теоретичні аспекти вейвлетного аналізу. Квантова обробка сигналу: реалізація та порівняння

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
Презентація PowerPoint (18 слайдів)

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
Розділи 1-3	НЕЛАСА Г.В., к.т.н., доцент кафедри ІБтаН	 01.10.2024	 13.12.2024
Нормоконтроль	КОРОЛЬКОВ Р.Ю., к.т.н., доцент кафедри ІБтаН	19.12.2024	19.12.2024

7. Дата видачі завдання «01» жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	01.10.2024	Виконано
2	Аналіз історичних відомостей розвитку швидкого перетворення Фур'є.	01.10-06.10.2024	Виконано
3	Дослідження перетворення Фур'є, зокрема дискретного.	07.10-15.10.2024	Виконано
4	Аналіз алгоритму Кулі-Тьюкі ШПФ.	16.10-27.10.2024	Виконано
5	Дослідження вейвлетів та їх перетворень.	28.10-10.11.2024	Виконано
6	Аналіз вейвлета Добеші.	11.11-17.11.2024	Виконано
7	Кодова квантова реалізація перетворення вейвлета Добеші.	18.11-01.12.2024	Виконано
8	Порівняльний аналіз квантових перетворень Добеші та Фур'є.	02.12-08.12.2024	Виконано
9	Оформлення пояснювальної записки та відповідної документації.	09.12-15.12.2024	Виконано
10	Нормоконтроль та рецензування.	16.12-19.12.2024	Виконано
11	Захист дипломної роботи.	24.12.2024	Виконано

Студентка  (підпис) Аніта ЗАЙЦЕВА (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)  (підпис) Ганна НЕЛАСА (Ім'я ПРІЗВИЩЕ)

АНОТАЦІЯ

Пояснювальна записка до магістерської роботи: 69 с., 3 табл., 16 рис., 2 дод., 40 джерел.

АЛГОРИТМ КУЛІ-ТЬЮКІ, ВЕЙВЛЕТ ДОБЕШІ, ВЕЙВЛЕТ-ПЕРЕТВОРЕННЯ, ШВИДКЕ ПЕРЕТВОРЕННЯ ФУР’Є

Об'єктом дослідження є квантові методи обробки сигналів.

Предметом дослідження є порівняння квантових алгоритмів вейвлет-перетворення Добеші та перетворення Фур'є.

Метою роботи є дослідження та реалізація квантового вейвлет-перетворення Добеші для обробки сигналів та подальше порівняння з квантовим перетворенням Фур’є.

Квантове вейвлет-перетворення Добеші має практичну цінність в обробці сигналів та зображень, дозволяючи ефективно стискати дані та виділяти важливі характеристики.

У даній роботі було досліджено вейвлет-перетворення Добеші. Було використано готовий інструмент квантового перетворення Фур’є з бібліотеки pennylane. Було реалізовано квантове вейвлет-перетворення Добеші D4 за допомогою бібліотеки програмного забезпечення для квантового машинного навчання та квантових обчислень – PennyLane, а також хмарного середовища Google Colab для роботи з Jupyter Notebook. В кінці було проведено порівняльний аналіз даних квантових перетворень.

ABSTRACT

Explanatory note to the master's thesis: 69 p., 3 tables, 16 figure, 2 appendix, 40 sources.

COULEY-TUKEY ALGORITHM, DAUBECHIES WAVELET, FAST FOURIER TRANSFORM, WAVELET TRANSFORM

The object of research is quantum methods of signal processing.

The subject of the research is the comparison of the quantum algorithms of Daubechies wavelet transform and Fourier transform.

The aim of the work is to research and implement the quantum Daubechies wavelet transform for signal processing and further comparison with the quantum Fourier transform.

Daubechies quantum wavelet transform has practical value in signal and image processing, allowing efficient data compression and extracting important features.

The Daubechies wavelet transform was investigated in this paper. The out-of-the-box quantum Fourier transform tool from the pennylane library was used. A Daubechies D4 quantum wavelet transform was implemented using PennyLane, a software library for quantum machine learning and quantum computing, as well as the Google Colab cloud environment for working with Jupyter Notebook. At the end, a comparative analysis of the quantum transform data was performed.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БВП – безперервне вейвлет-перетворення;

ДВП – дискретне вейвлет-перетворення;

ДПФ – дискретного перетворення Фур'є;

ЕЕГ– електроенцефалографія;

ЕКГ – електрокардіографія;

КПФ – квантове перетворення Фур’є;

ПФДЧ – перетворення Фур'є з дискретним часом;

ШПФ – швидке перетворення Фур'є

ВСТУП

Квантові обчислення та обробка інформації на основі квантових принципів відкривають нові горизонти в багатьох наукових і практичних сферах. Однією з основних задач у цій галузі є ефективне представлення та аналіз даних, що стало поштовхом для розробки численних алгоритмів. Серед них особливе місце займають квантове перетворення Фур'є та квантове вейвлет-перетворення. Обидва методи пропонують різні підходи до обробки сигналів, дозволяючи отримувати інформацію з високою точністю та швидкістю.

Квантове перетворення Фур'є стало революційним відкриттям, яке дозволяє зменшити складність обчислень у порівнянні з класичними алгоритмами, зокрема, у задачах факторизації та ін. На іншому фронті, квантове вейвлет-перетворення забезпечує локалізацію інформації як у часовій, так і в частотній областях, що робить його надзвичайно корисним у аналізі сигналів з мінливою характеристикою.

У даній роботі буде здійснено реалізацію та порівняння цих двох підходів, з акцентом на їх теоретичні основи, алгоритмічні реалізації та практичні застосування. Аналіз дозволить виявити сильні та слабкі сторони кожного з методів, а також визначити, в яких умовах їх використання є найбільш доцільним.

Тема даної дипломної роботи актуальна оскільки обидва методи мають різні підходи до обробки інформації в квантових системах. Квантове перетворення Фур'є широко використовується в алгоритмах, тоді як вейлет-перетворення забезпечує кращу локалізацію сигналів у часі та частоті.

Метою цієї дипломної роботи є дослідження та реалізація квантового вейвлет-перетворення Добеші для обробки сигналів та подальше порівняння з квантовим перетворенням Фур’є.

Завданнями роботи є:

– розглянути перетворення Фур’є;

– розглянути вейвлет Добеші;

– реалізувати квантове перетворення Добеші та застосувати готовий інструмент квантового перетворення Фур’є;

– зробити порівняльний аналіз перетворень.

Методи дослідження включають аналітичний огляд наукової літератури та нормативних документів, теоретичний аналіз перетворення Фур’є, на прикладі алгоритму ШПФ Кулі-Тьюкі, а також вейвлет-перетворення, на прикладі вейвлета Добеші, та програмну реалізацію даних перетворень на квантах для подальшого порівняння.

У цій дипломній роботі висвітлені теоретичні основи перетворення Фур’є та вейвлет-перетворення, програмна реалізація вейвлет-перетворення Добеші та застосування готового інструменту квантового перетворення Фур’є для подальшого порівняльного аналізу.

Структура дипломної роботи включає такі розділи: вступ, теорія швидкого перетворення Фур’є, теоретичні аспекти вейвлетного аналізу, квантова обробка сигналів: реалізація та порівняння та висновки.

В ході виконання роботи представлено квантову реалізацію перетворень Фур’є та вейвлетів Добеші, а також порівняльний аналіз як підсумок дослідження.

Результати проведених досліджень можуть бути впроваджені у різних галузях, таких як обробка зображень, телекомунікації та штучний інтелект.

1 ТЕОРІЯ ШВИДКОГО ПЕРЕТВОРЕННЯ ФУР’Є

1.1 Історія швидкого перетворення Фур'є

Історія швидких алгоритмів для дискретного перетворення Фур'є (далі – ДПФ) має свої коріння у неопублікованій роботі Карла Фрідріха Гауса 1805 року щодо орбіт астероїдів Паллади та Юнони. Гаусс прагнув інтерполювати орбіти на основі обмежених спостережень, і його метод сильно нагадував той, що був опублікований у 1965 році Джеймсом Кулі та Джоном Тьюкі [1], яким, як правило, приписують створення сучасного загального алгоритму ШПФ. Хоча роботи Гаусса передували навіть досягненням Джозефа Фур’є у 1822 році, він не досліджував складність методу і, нарешті, використовував інші підходи для досягнення тих самих результатів.

Між 1805 та 1965 роками різними авторами були опубліковані різні версії швидких перетворень Фур'є (далі – ШПФ). У 1932 році Френк Єйтс розробив свій варіант під назвою "Алгоритм взаємодії", який ефективно обчислював перетворення Адамара та Уолша [2]. Цей алгоритм досі використовується в галузі статистичного проектування та аналізу експериментів. У 1942 році Г. К. Даніельсон та Корнеліус Ланцос представили свою версію обчислення ДПФ для рентгенівської кристалографії [3], де виявили, що можна використовувати "періодичність" і застосовувати "трюк подвоєння".

В матричній схемі Ланцоса першим кроком зводяться дані парних і непарних компонентів функції. Таким чином можна буде обчислити дійсні перетворення косинуса і синуса. Інша частина процесу використовує симетрії синусів і косинусів, подібно до методів Рунге [4]. Після цього використовується алгоритм подвоєння. Другим кроком є множення обертового коефіцієнта і третім кроком виконується обчислення метелика. Але варто спостерігати за точністю, порівнюючи два вхідних даних: коефіцієнти Фур'є підряду. Якщо вони однакові, високочастотна половина

нового перетворення дорівнює нулю, і, припускаючи, що немає частотних компонентів за межами цього, частота дискретизації достатня і помилок не виникло [4]. Таким чином, Ланцос мав алгоритм ШПФ і як би у нього був електронний комп’ютер, він би скоріше за все написав програму алгоритму.

У 1965 році Джеймс Кулі та Джон Тьюкі, незалежно один від одного, відкрили знову вищезгадані алгоритми та опублікували більш загальний алгоритм швидкого перетворення Фур'є. Немалу роль у цього зіграв Річард Гарвін. Коли Кулі працював над власним проектом, до нього завітав Гарвін з деякими нотатками, які він зробив під час зустрічі з Джоном Тьюкі на засіданні Науково-консультативного комітету президента Кеннеді, членами якого вони обидва були [4]. Річард Гарвін розумів, що розрахунки Тьюкі важливі, тому він хотів, щоб ідея була реалізована.

Під час дискусії з Тьюкі, Гарвін визнав, що алгоритм ШПФ можна застосувати не лише до проблем національної безпеки, а й до широкого кола проблем. Його цікавило саме визначення періодичності орієнтації спінів у тривимірному кристалі гелію-3.

Сам Кулі писав, що його цікавило тільки його дослідження, але після того, як дізнався репутацію Гарвіна, створив тривимірну програму ШПФ. До розробки алгоритму було докладено зусиль, щоб заощадити пам’ять і адресацію шляхом перезапису даних [4].

Закінчимо цю історію двома концепціями в чисельних алгоритмах, які вивів Кулі у своїй статті. Доречи, кажуть, що вони з’явились вперше саме в алгоритмі швидкого перетворення Фур’є.

Перша концепція. Підхід «розділяй і володарюй» (divide-and-conquer). Якщо велика проблема розміру N вимагає зусиль, які зростають як деякий ступінь N, тоді можна, загалом, скоротити обчислення, розділивши задачу на частини, що мають однакову структуру [4].

Друга концепція. Асимптотична поведінка (asymptotic behavior) кількості операцій. Асимптотичний аналіз в даному випадку використовується для розгляду алгоритму, що застосовує великі набори

вхідних даних. Важливість ранніх форм алгоритму ШПФ не була помічена, так як ця поведінка не була істотна для малих N.

1.2 Визначення перетворення Фур'є

Почнемо з ряду Фур'є, який розбиває періодичну функцію на суму синусоїдальних функцій [5]. Періодичною є функція з основним періодом *T*, якщо наступне вірно для всіх *t*:

$$f(t+T) = f(t). \tag{1.1}$$

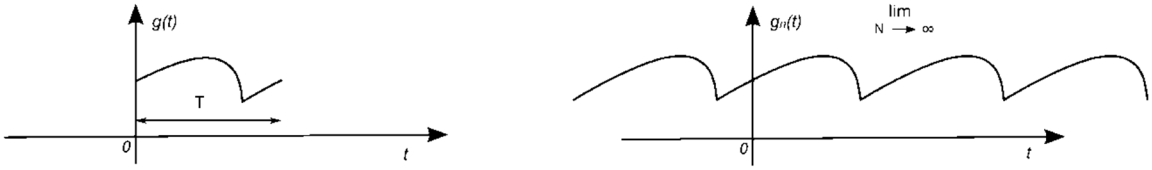
Простіше кажучи [5], функція часу з періодом *T* буде мати те саме значення в *T* секунд, що і зараз, незалежно від того, коли спостерігаємо функцію. Основний період *T* – значення, яке більше нуля, яке є найменшим можливим *T* і для якого рівняння (1.1) завжди вірне.

Рядом Фур'є з періодом *T* [5] є нескінченна сума синусоїдальних функцій (косинус і синус), кожна з яких має частоту, кратну 1/*T*. Ряд має константи a_m, b_n, які є коефіцієнтами ряду, та визначають відносні ваги для кожної з синусоїд. Формула виглядає так:

$$g(t) = a_0 + \sum_{m=1}^{\infty} a_m \cos\left(\frac{2\pi m t}{T}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{2\pi n t}{T}\right) =$$

$$\sum_{m=0}^{\infty} a_m \cos(\frac{2\pi m t}{T}) + \sum_{n=1}^{\infty} b_n \sin(\frac{2\pi n t}{T}). \tag{1.2}$$

Для загального розуміння розглянемо рисунок 1.1. На верхньому графіку показано неперіодичну функцію *g(t)*, яка визначена в інтервалі від 0 до *T*. Проаналізувавши функцію на цьому інтервалі, створено ряд Фур'є (рис. 1.1 б). Ряд Фур'є є періодичною функцією, навіть якщо вихідна функція *g(t)* нею не є.



а) б)

а - графік неперіодичної функції; б - графік ряду Фур’є

Рисунок 1.1 – Наближення неперіодичної функції за допомогою ряду Фур'є

Перетворення Фур’є – математичний інструмент, який розкладає будь-яку функцію на суму синусоїдальних базисних функцій, кожна з яких є комплексною експонентою з різною частотою.

Перетворення Фур’є функції *g(t)* визначається за формулою:

$$F\{g(t)\} = G(f) = \int_{-\infty}^{\infty} g(t)e^{-2\pi ift}\,dt. \quad (1.3)$$

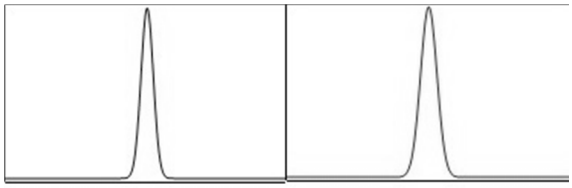
G(f), яку часто називають спектром *g*, показує, скільки потужності *g(f)* на частоті *f*. За допомогою зворотного перетворення Фур’є можна отримати *g* з *G*:

$$F^{-1}\{G(f)\} = \int_{-\infty}^{\infty} G(f)e^{2\pi ift}\,df = g(t). \quad (1.4)$$

В результаті *g(t)* і *G(f)* утворюють пару Фур’є: вони є різними представленнями однієї базової ідентичності [6].

В математиці дискретне перетворення Фур'є перетворює скінченну послідовність рівновіддалених вибірок функції в однакову довжину послідовності рівновіддалених вибірок перетворення Фур'є з дискретним часом, яке є комплекснозначною функцією частоти [7]. Інтервал, на якому перетворення Фур'є з дискретним часом дискретизується, є зворотним значенням тривалості вхідної послідовності [8].

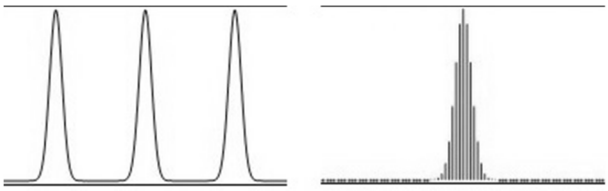
Якщо вихідна послідовність охоплює всі ненульові значення функції, її ПФДЧ є безперервним (і періодичним), а ДПФ забезпечує дискретні вибірки одного циклу. Якщо вихідна послідовність є одним циклом періодичної функції, ДПФ забезпечує всі ненульові значення одного циклу ПФДЧ. Далі серією графіків (рис. 1.2 – 1.5) буде продемонстровано зв’язок між безперервним перетворенням Фур’є та дискретним перетворенням Фур’є.



а) б)

а – графік неперервної функції; б - графік перетворення Фур’є неперервної функції

Рисунок 1.2 – Перетворення Фур’є неперервної функції

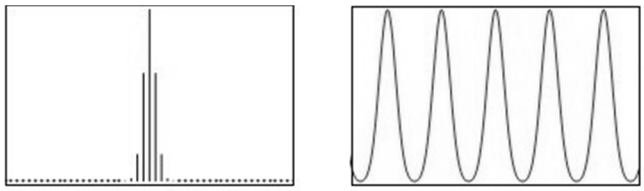


а) б)

а - Графік періодичного підсумовування вихідної функції; б - графік перетворення Фур’є

Рисунок 1.3 – Періодичне підсумовування функції

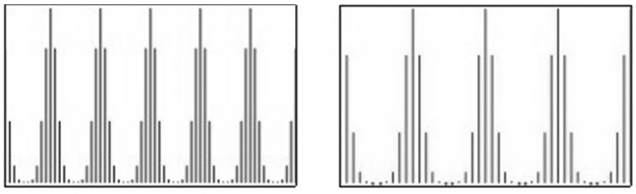
Треба уточнити, що рис. 1.3 б перетворення Фур’є дорівнює нулю, за винятком дискретних точок. Зворотне перетворення – сума синусоїд, що є рядами Фур'є.



а) б)

а – графік вихідної дискретизованої функції; б - графік перетворення Фур’є

Рисунок 1.4 – Дискретизація функції



а) б)

а – графік зворотного ДПФ; б – графік ДПФ

Рисунок 1.5 – Дискретне перетворення Фур’є

Дискретне перетворення Фур'є перетворює [8] послідовність з N комплексних чисел $\{\mathbf{x}_n\} := x_0, x_1, \ldots, x_{N-1}$ в іншу послідовність комплексних чисел, $\{\mathbf{X}_n\} := X_0, X_1, \ldots, X_{N-1}$, який визначається:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi \frac{k}{N} n}. \quad (1.5)$$

Для того, щоб визначити ДПФ дискретного сигналу x_n, потрібно множити кожне його значення e на деяку функцію n. Потім підсумувати результати, які отримали для заданого n. Якби використовували комп’ютер для обчислення ДПФ сигналу, потрібно було б виконувати операції N (множення) $\times$ N (додавання) = $O(N^2)$.

Рівняння 1.5 можна оцінити за межами $k \in [0, N-1]$, і ця розширена послідовність є *N*-періодичною. Відповідно, іноді використовуються інші послідовності з *N* індексів, наприклад $[-\frac{N}{2}, \frac{N}{2}-1]$ (якщо *N* парне) і $[-\frac{N-1}{2}, \frac{N-1}{2}]$ (якщо *N* непарне), що означає заміну лівої та правої половин результату перетворення [9].

Рівняння зворотного ДПФ наведено нижче [7]:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k \cdot e^{i2\pi\frac{k}{N}n}. \quad (1.6)$$

Швидке перетворення Фур'є – алгоритм, який визначає дискретне перетворення Фур'є об'єкта швидше, ніж обчислює його. На жаргоні інформатики ШПФ зменшує кількість обчислень, необхідних для завдання розміром *N* від $O(N^2)$ до $O(N\log N)$ [10]. Можливо здається це все неважливим, але коли *N* достатньо великий, він може змінити ситуацію. Наглядно це можна побачити в таблиці 1.1.

Таблиця 1.1 – Масштабування часу ДПФ для різних розмірів N

N	10^3	10^6	10^9
N^2	10^6	10^{12}	10^{18}
$N\log_2 N$	10^4	$20x10^6$	$30x10^9$

Найвідоміші алгоритми ШПФ залежать від факторизації *n*, але є з $O(N\log N)$ складністю для всіх, навіть простих *n*. Багато алгоритмів ШПФ залежать лише від того, що $e^{-2\pi i/n}$ є *n*-им примітивним коренем з одиниці, і тому може бути застосований до аналогічних перетворень над будь-яким скінченним полем. Оскільки зворотне ДПФ таке ж, як і ДПФ, але з протилежним знаком у експоненті та коефіцієнтом *1/n*, будь-який алгоритм ШПФ можна легко адаптувати для нього [11].

Алгоритми швидкого перетворення Фур'є можуть зазнавати помилок при використанні арифметики з плаваючою комою з обмеженою точністю, проте ці помилки зазвичай є досить малими. Більшість алгоритмів, таких як алгоритм Кулі-Тьюкі, демонструють відмінні числові властивості завдяки структурі попарного підсумовування. Верхня межа відносної похибки для алгоритму Кулі-Тьюкі становить $O(\varepsilon \log n)$, порівняно з $O(\varepsilon n^{3/2})$ для наївної формули ДПФ [12], де ε - відносна точність машини з плаваючою комою. Фактично, середньоквадратичні помилки набагато кращі, ніж ці верхні межі, будучи лише $O(\varepsilon\sqrt{\log n})$ для Кулі-Тьюкі і $O(\varepsilon\sqrt{n})$ для наївного ДПФ [13]. Проте ці результати дуже чутливі до точності обертових факторів, які застосовуються в алгоритмах ШПФ (тобто значень тригонометричних функцій). Нерідко недбалі реалізації можуть призводити до значно гіршої точності, наприклад, якщо вони використовують неточні рекурентні формули для тригонометричних функцій. Деякі алгоритми швидкого перетворення Фур'є, крім Кулі-Тьюкі, такі як алгоритм Рейдера-Бреннера, в цілому є менш стабільними.

В арифметиці з фіксованою комою помилки кінцевої точності, накопичені алгоритмами ШПФ, є гіршими, а середньоквадратичні помилки зростають як $O(\sqrt{n})$ для алгоритму Кулі-Тьюкі [14]. Для досягнення такої точності необхідно уважно підходити до масштабування, щоб зменшити втрати точності. Алгоритми ШПФ з фіксованою точкою, такі як алгоритм Кулі-Тьюкі, включають зміну масштабу на кожному проміжному етапі розкладання.

Щоб перевірити правильність реалізації ШПФ, можна отримати суворі гарантії $O(n \log n)$ часу за допомогою простої процедури перевірки лінійності, імпульсної реакції та властивостей зсуву в часі перетворення на випадкових входах [15].

1.3 Алгоритм Кулі–Тьюкі

Проблема з використанням стандартного ДПФ полягає в тому, що воно вимагає великих матриць множень і сум над усіма елементами, які є надзвичайно складними операціями [16]. Алгоритм Кулі-Тьюкі робить це без множення матриці та з меншою кількістю підсумовувань. Хитрість алгоритму Кулі-Тьюкі полягає в рекурсії [16]. На початку розбиваємо матрицю, над якою хочемо виконати швидке перетворення Фур'є, на дві частини: перша для всіх елементів з парним індексом, друга для всіх з непарним. Після цього продовжуємо розбивати масив, поки не отримаємо керований розмір масиву для виконання ДПФ. За допомогою рекурсії зменшується складність до $\sim O(n\log n)$, а це вже є можливою операцією.

Загалом, алгоритми Кулі-Тьюкі рекурсивно перевиражають ДПФ складеного розміру $N = N_1 N_2$ як [17] :

– виконайте N_1 ДПФ розміром N_2;

– помножте на комплексні корені з одиниці (їх ще називають факторами обертання);

– виконайте N_2 ДПФ розміром N_1.

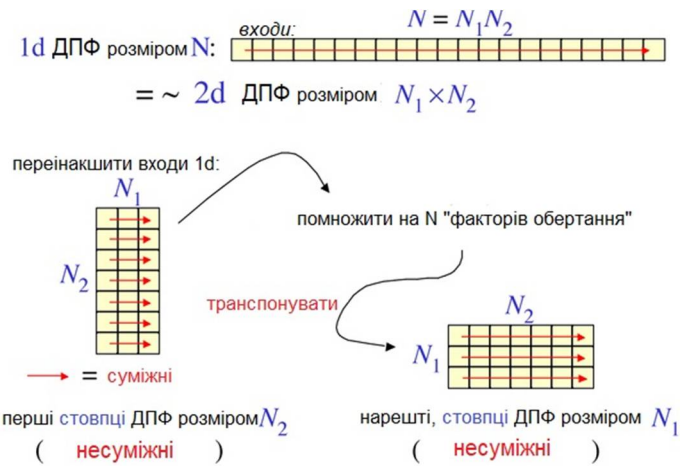


Рисунок 1.6 – Основний крок ШПФ Кулі–Тьюкі
для загального розкладання на множники

Або N_1, або N_2 – малий множник (не обов'язково простий), який називається основою (вона може відрізнятися між етапами рекурсії). Якщо N_1 є основою – це алгоритм децимації за часом (тобто розбиття сигналу в часовій області на більш дрібні сигнали), якщо N_2 є основою – це децимація за частотою (процес зменшення частоти дискретизації сигналу до нижчої частоти дискретизації, що відрізняється від вихідної частоти на ціле число; так званий алгоритмом Санде–Тьюкі). Версія, яка була представлена вище, була алгоритмом radix-2 децимації за часом. У кінцевому виразі фаза множення непарного перетворення виступає як коефіцієнт обертання. Комбінація +/- (метелик) парного та непарного перетворень формує розмір 2 для дискретного перетворення Фур'є.

Алгоритм radix-2 можна описати як $\log_2(N)$ ітерацій, кожна з яких проходить через дані, виконуючи двоточкове перетворення Фур'є [4].

$$X_1(k) = X_0(k) + X_0(k+d)W^e, \tag{1.7}$$

$$X_1(k+d) = X_0(k) - X_0(k+d)W^e, \tag{1.8}$$

де показник степеня e – проста функція індексу k. Переміщення d або починається з 2 і подвоюється на кожному кроці, або починається з $N/2$ і зменшується вдвічі на кожному кроці. Це можна описати як алгоритм подвоєння. На першому кроці отримуємо набір 2-точкових перетворень Фур'є. На другому кроці коректують половину точок за допомогою фазового коефіцієнта та створюють набір 4-точкових перетворень Фур'є. Це повторюється, створюючи 8-точкові перетворення тощо. Наведене вище обчислення називають «метеликом» через його зовнішній вигляд, коли описується графіком потоку сигналу, як на рис. 1.7 [4].

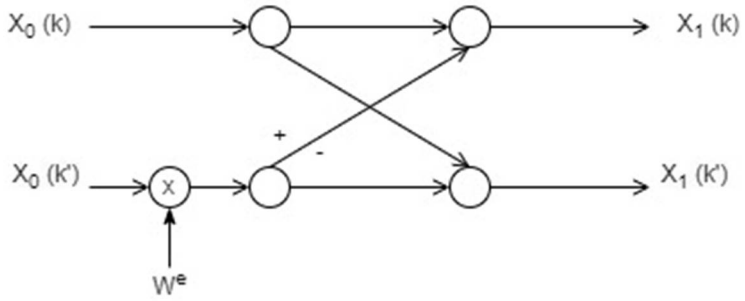


Рисунок 1.7 – Графік потоку сигналу для алгоритму radix-2
з децимацією за часом

Студент Тьюкі, Гордон Санде, вивів іншу формулу алгоритму [4] :

$$X_1(k) = X_0(k) + X_0(k+d), \tag{1.9}$$

$$X_1(k+d) = [X_0(k) - X_0(k+d)]W^e. \tag{1.10}$$

Рівняння 1.9-1.10 відомі як форма ШПФ Санде-Тьюкі та описується графіком потоку сигналу на рис. 1.8. Санде назвав комплексну фазову функцію W^e – «фактором обертання» [4].

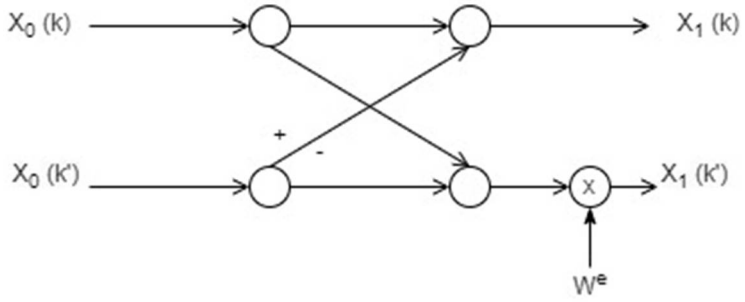


Рисунок 1.8 – Графік потоку сигналу для алгоритму radix-2
з децимацією по частоті (Санде-Тьюкі)

Наостанок розглянемо алгоритм Кулі-Тьюкі на базі простого псевдокоду, який буде наведено на рис. 1.9 [18]. Псевдокод реалізує алгоритм за методикою "розділяй і володарюй".

$x_{0,\dots,N-1}$←**ditfft2**(x, N, s):
if N=1 **then**
X_0←x_0
else
$X_{0,\dots,N/2-1}$←**ditfft2**(x, N/2, 2s)
$X_{N/2,\dots,N-1}$←**ditfft2**(x+s, N/2, 2s)
for k=0 to N/2-1 **do**
p←X_k
q←exp(-2πi/N k) $X_{k+N/2}$
X_k←p+q
$X_{k+N/2}$←p-q
end for
end if
	ДПФ (x_0, x_s, x_{2s}, …, $x_{(N-1)s}$):
	Простий базовий варіант ДПФ розміром 1.
	
	ДПФ (x_0, x_{2s}, x_{4s}, …, $x_{(N-2)s}$)
	ДПФ (x_s, x_{s+2s}, x_{s+4s}, …, $x_{(N-1)s}$)
	Поєднання ДПФ даох половинок у повний ДПФ:

Рисунок 1.9 – Псевдокод для алгоритму radix-2

Вхідні параметри:

- *x* масив даних (сигнал), який потрібно перетворити;
- *N* кількість точок, які потрібно обробити (*N* - ступінь двійки);
- *s* зсув для вибору елементів з *x*.

Якщо *N*=1, то результат дорівнює x_0. Це базовий випадок, оскільки ДПФ для одного елемента дорівнює цьому елементі. При *N*>1 алгоритм розбиває дані на дві частини: елементи з парними індексами (x_0, x_{2s}, x_{4s}, ..., $x_{(N-2)s}$) та елементи з непарними (x_s, $x_{s+2}s$, ..., $x_{(N-1)s}$). Обидві половини обробляються повторно за допомогою того ж алгоритму. За допомогою циклу, результати з двох частин комбінуються в остаточний масив *X*. Для кожного *k* виконується комбінація: *p* – значення з першої половини; *q* – експоненційний множник, який залежить від *k* (порядок елемента) і *N* (розмір). Це значення допомагає обчислити взаємодію між половинами. X_k і $X_{k\ +\ N/2}$ отримують нові значення на основі *p* та *q*, що дозволяє сформувати остаточний ДПФ. Шляхом повторного розділення та комбінування зменшується складність алгоритму з $O(N^2)$ до $O(N\log N)$.

2 ТЕОРЕТИЧНІ АСПЕКТИ ВЕЙВЛЕТНОГО АНАЛІЗУ

2.1 Теорія вейвлетів

Наприкінці 1970-х років інженер-геофізик Дж. Морлет зіткнувся з проблемою аналізу сигналів, які містять високочастотні компоненти з короткими проміжками часу, а також низькочастотні компоненти з великими проміжками часу. Короткочасне перетворення Фур'є змогло проаналізувати або високочастотні компоненти за допомогою вузьких вікон (широкосмуговий частотний аналіз), або низькочастотні компоненти за допомогою широких вікон (вузькосмуговий частотний аналіз), але не обидва разом [19,20]. Він розробив метод використання різних віконних функцій для аналізу різних діапазонів частот. Ці вікна в свою чергу були згенеровані розширенням або стисненням прототипу Гауса та мали компактну підтримку і за часом, і за частотою. Через «малу та коливальну» природу цих віконних функцій Морлет назвав свої основні функції вейвлетами постійної форми. Однак, як і Фур'є у свій час, зазнав критики з боку колег. У пошуках підтримки, Морлет зустрів фізика-теоретика квантової механіки А. Гроссмана, який формалізував перетворення та розробив зворотне перетворення. Проте вони не знали, що їх відкриття було повторним і дещо іншою інтерпретацією роботи Альберто Кальдерона з гармонічного аналізу 1964 року.

В 1984 році французький математик Ів Мейєр помітивши подібність між роботами Морлета та Кальдерона, почав працювати над розробкою вейвлетів із кращими властивостями локалізації. Через рік побудував ортогональні вейвлет-базисні функції, в яких були покращені часова та частотна локалізації. Але, на іронію долі, виявилося, що інший аналітик Дж.О. Стромберг вже відкрив ті ж самі вейвлети близько п'яти років тому. Але треба зауважити, що першовідкривачем був німецький математик Альфред Гаар у 1909 році. Незважаючи на те, що вейвлети Хаара є першими

та найпростішими ортонормальними вейвлетами, вони не мають практичного застосування через їх погану частотну локалізацію. Розробки ортонормованих базисних функцій математика були розширені в 1930-х роках Полом Леві, коли він вивчав випадкові сигнали броунівського руху, і незалежно Літлвудом і Пейлі, які працювали над локалізацією вкладених енергій функції.

Тим часом Інґрід Добеші, колишня аспірантка Гроссмана, розробила вейвлет-фрейми для дискретизації параметрів часу та масштабу вейвлет-перетворення. Це надало більшу свободу у виборі базисних функцій. А у 1986 році Маллат спільно з Мейєром розвинув ідею аналізу з різною роздільною здатністю для дискретного вейвлет-перетворення. Основна ідея полягала [19,20] в розкладанні дискретного сигналу на його подвійні смуги частот за допомогою серії фільтрів низьких і високих частот для обчислення ДВП з наближень у різних масштабах. Таким чином, Добеші разом зі Стефаном Маллатом приписують розвиток переходу від безперервного до дискретного аналізу сигналів. Також у 1988 році, завдяки розвитку ортонормованих базисів Добеші для вейвлетів з компактною підтримкою, було закладено основи сучасної теорії вейвлетів.

Вейвлет – хвилеподібне коливання з амплітудою, яка починається з нуля, збільшується або зменшується, а потім повертається до нуля один або кілька разів. Слово вейвлет еквівалент французького слова ondelette, що означає «мала хвиля». Вейвлет або сімейство вейвлетів визначають такими способами.

Фільтр масштабування – фільтр низьких частот із кінцевою імпульсною характеристикою, довжина якої *2N* та сума – 1. Цим методом можна визначити вейвлети Добеші та Симлета.

Вейвлет функція *ψ(t)* – вейвлет, який представляється лише в часовій області. Прикладом є вейвлети мексиканського капелюха.

Функція масштабування. Вейвлети визначаються вейвлет-функцією ψ(t) (тобто материнським вейвлетом) і функцією масштабування *φ(t)* (також

називається батьківським вейвлетом) у часовій області. Вейвлет-функція фактично є смуговим фільтром і масштабуванням, що для кожного рівня вдвічі зменшує його пропускну здатність. Це створює проблему: щоб охопити весь спектр, потрібна нескінченна кількість рівнів. Функція масштабування фільтрує найнижчий рівень перетворення та забезпечує охоплення всього спектру [21]. Цим методом можуть бути визначені вейвлети Мейєра.

Узагальнемо. Вейвлет є математичною функцією, яку використовують для поділу заданої функції або сигналу безперервного часу на компоненти масштабу. Зазвичай кожному з компонентів шкали можна призначати діапазон частот, після чого можна вивчати з роздільною здатністю, яка відповідає масштабу компонента. Вейвлети є масштабованими та трансльованими копіями (відомими як «дочірні вейвлети») коливальної форми сигналу кінцевої довжини або швидкозгасаючих (відомих як «материнський вейвлет») [21] .

Представленням функції вейвлетами є вейвлет-перетворення. Класифікуються на дискретне та безперервне. Треба зауважити, що дискретне вейвлет-перетворення є безперервним (аналоговим) перетворенням як і безперервне вейвлет-перетворення. Безперервне працює над усім можливим масштабом і трансляцією, дискретне, в свою чергу, використовує певну підмножину значень масштабу та трансляції або сітку представлення.

2.2 Від безперервного до дискретного перетворення

Вейвлет-перетворення – це математичні інструменти для аналізу даних, де характеристики змінюються в різних масштабах [22]. Сигнали

можуть характеризуватися змінами частот, перехідними процесами або повільно варіативними трендами.

Аналіз Фур'є полягає в розкладанні сигналу на синусоїди певних частот, вейвлет-аналіз базується на розкладанні сигналів на зміщені та масштабовані версії вейвлета [22]. Вейвлет, на відміну від синусоїди, є швидко затухаючим хвилеподібним коливанням, що дозволяє їм представляти дані на різних масштабах.

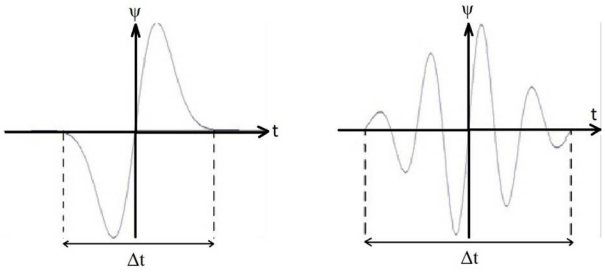
Ідея вейвлет-перетворення полягає в наступному: перетворення дозволяє зміни лише в розширенні часу, а не в формі. Таким чином, накладає обмеження на вибір відповідних базових функцій. Очікується, що зміни в розширенні часу відповідатимуть відповідній частоті аналізу базисної функції.

На основі принципу невизначеності обробки сигналів [23],

$$\Delta t \Delta \omega \geq \frac{1}{2}, \tag{2.1}$$

де t - час, ω - кутова частота ($\omega = 2\pi f$, f - звичайна частота).

Чим вища вимога до часової роздільної здатності, тим нижчою повинна бути роздільна здатність за частотою. Якщо вибрати більше розширення вікна аналізу, значення Δt також зросте.



а) б)

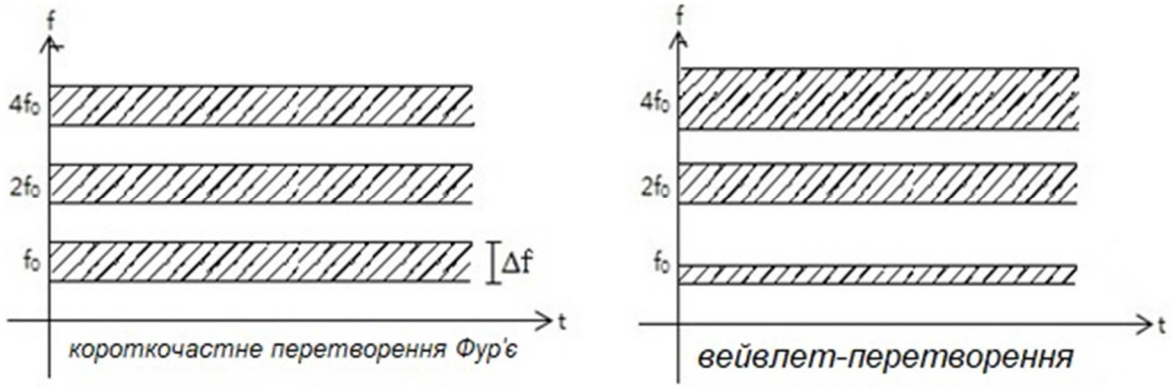
а – графік при малому Δt; б - графік при великому Δt

Рисунок 2.1 – Вимоги до часової роздільної здатності [23]

При великому Δt погана роздільна здатність часу, але хороша частотна; низька частота, великий коефіцієнт масштабування. При малому Δt хороша роздільна здатність за часом, але погана за частотою; висока частота, малий коефіцієнт масштабування.

Іншими словами, базисну функцію ψ можна сприймати як імпульсну характеристику системи, яка використовувалася для фільтрації функції *x(t)*. Перетворений сигнал надає інформацію про час і частоту. Таким чином, вейвлет-перетворення містить інформацію, подібну до короткочасного перетворення Фур'є, але з додатковими особливими властивостями вейвлетів, які проявляються в роздільній здатності в часі для вищих частот аналізу базисної функції.

Для прикладу, аналіз трьох накладених синусоїдальних сигналів $y(t)=\sin(2\pi f_0 t)+\sin(4\pi f_0 t)+\sin(8\pi f_0 t)$ [23].



а) б)

а – короткочасне перетворення Фур’є; б – вейвлет-перетворення

Рисунок 2.2 – Аналіз при різних перетвореннях

Сигнал або функцію *f(t)* часто легше аналізувати, описувати або обробляти, якщо виразити її у вигляді лінійного розкладання за

$$f(t) = \sum_l a_l \psi_l(t), \qquad (2.2)$$

де l – цілочисельний індекс для кінцевої або нескінченної суми, a_l – дійсні коефіцієнти розкладання, а $\psi_l(t)$ – набір дійсних функцій від t, який називається набором розкладання [24]. Якщо рівняння (2.2) є унікальним, то множина є базисом для класу функцій, які можна виразити в такій формі. Якщо базис є ортогональним, то

$$\langle \psi_k(t), \psi_l(t) \rangle = \int \psi_k(t)\psi_l(t)dt = 0 \quad k \neq l, \tag{2.3}$$

тоді коефіцієнти можна обчислити за скалярним добутком [24]

$$a_k = \langle f(t), \psi_k(t) \rangle = \int f(t)\psi_k(t)dt. \tag{2.4}$$

Для ряду Фур'є ортогональними базисними функціями $\psi_k(t)$ є $\sin(k\omega_\square t)$ і $\cos(k\omega_\square t)$ з частотами $k\omega_\square$. У ряді Тейлора неортогональними базисними функціями є прості одночлени t^k, а для багатьох інших розкладів використовуються різні поліноми. Також існують розширення, які застосовують сплайни і навіть фрактали.

Для вейвлет-розкладу будується двопараметрична система

$$f(t) = \sum_k \sum_j a_{j,k}\psi_{j,k}(t), \tag{2.5}$$

де і j, і k - цілі індекси, а $\psi_{j,k}(t)$ - функції вейвлет-розкладу, які зазвичай утворюють ортогональний базис [24].

Безперервне вейвлет-перетворення (БВП) в математиці є формальним інструментом, щоб забезпечити повне представлення сигналу, дозволяючи при цьому трансляції та параметру масштабу вейвлетів безперервно змінюватися. Безперервне вейвлет-перетворення функції $x(t)$ за допомогою безперервних масштабних перетворень a та переносів b материнського вейвлета виражається наступним інтегралом [25]

$$X_\omega(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} x(t)\bar{\psi}\left(\frac{t-b}{a}\right) dt, \quad (2.6)$$

де *ψ(t)* - безперервна функція у часовій та частотній області ("материнський вейвлет"), накладна лінія представляє операцію комплексного сполучення. Основною метою материнського вейвлета є забезпечення вихідної функції для генерації дочірніх вейвлетів, які в свою чергу є перекладеними та масштабованими версіями материнського. Для відновлення вихідного сигналу *x(t)* використовують перше зворотне БВП

$$x(t) = C_\psi^{-1} \int_0^{\infty} \int_{-\infty}^{\infty} X_\omega(a,b) \frac{1}{|a|^{1/2}} \tilde{\psi}\left(\frac{t-b}{a}\right) db \, \frac{da}{a^2}, \quad (2.7)$$

де $\tilde{\psi}(t)$ є подвійною функцією $\psi(t)$ [25].

Щодо коефіцієнту масштабу *a*, то він може або розширювати, або стискати сигнал. Коли коефіцієнт відносно низький, сигнал більш стиснутий, що призводить до більш детального кінцевого графіка. Недоліком є те, що низький коефіцієнт масштабування не залишається постійним протягом усього сигналу. З іншого боку, при високому коефіцієнті сигнал розтягується, що означає, що результуючий графік буде менш детальним.

Будь-яке вейвлет-перетворення, вейвлети для якого дискретно відбираються, є дискретним. Ключова перевага – часова роздільна здатність, яка в свою чергу фіксує і частоту, і інформацію про місцезнаходження в часі.

Дискретне вейвлет-перетворення сигналу *x* обчислюється, пропускаючи його через низку фільтрів (2.8). Спочатку зразки пропускають через фільтр низьких частот з заданою імпульсною характеристикою.

$$y[n] = (x \cdot g)[n] = \sum_{k=-\infty}^{\infty} x[n]g[n-k] \quad (2.8)$$

Сигнал також розкладається одночасно за допомогою фільтра високих частот *h*. Вихідні дані дають коефіцієнти деталізації (від фільтра високих частот) і коефіцієнти апроксимації (від фільтра низьких частот). Важливо,

щоб два фільтри були пов’язані один з одним, і вони відомі як квадратурний дзеркальний фільтр [26].

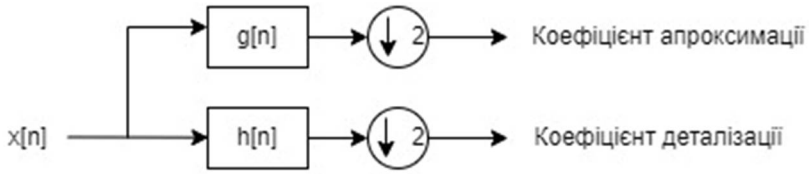


Рисунок 2.3 – Блок-схема аналізу фільтрів

Половина вибірок за правилом Найквіста може бути відкинута, так як половина частот сигналу видалена. Вихід фільтра нижніх частот g потім субдескретизується на 2 і далі обробляється, пропускаючи його через новий фільтр низьких частот g та фільтр високих частот h з половиною частоти зрізу попереднього [26]

$$y_{low}[n] = \sum_{k=-\infty}^{\infty} x[k]g[2n-k], \tag{2.9}$$

$$y_{high}[n] = \sum_{k=-\infty}^{\infty} x[k]h[2n-k]. \tag{2.10}$$

Дане розкладання зменшило вдвічі роздільну здатність за часом, так як тільки половина виходу кожного фільтра характеризує сигнал. Проте кожен з виходів має половину смуги частот входу, через що роздільна здатність частоти подвоєна.

З оператором підвибірки ↓ [26]

$$(y \downarrow k) = y[kn] \tag{2.11}$$

наведене вище можна записати стисло

$$y_{low} = (x \cdot g) \downarrow 2, \tag{2.12}$$

$$y_{high} = (x \cdot h) \downarrow 2. \tag{2.13}$$

Розкладання повторюється для подальшого збільшення частотної роздільної здатності та коефіцієнтів апроксимації, які розкладені завдяки фільтрам високих та низьких частот, після чого знижується дискретизація. Це представлено у вигляді бінарного дерева з вузлами, що представляють підпростір з іншою частотно-часовою локалізацією. Дерево відоме як банк фільтрів [26].

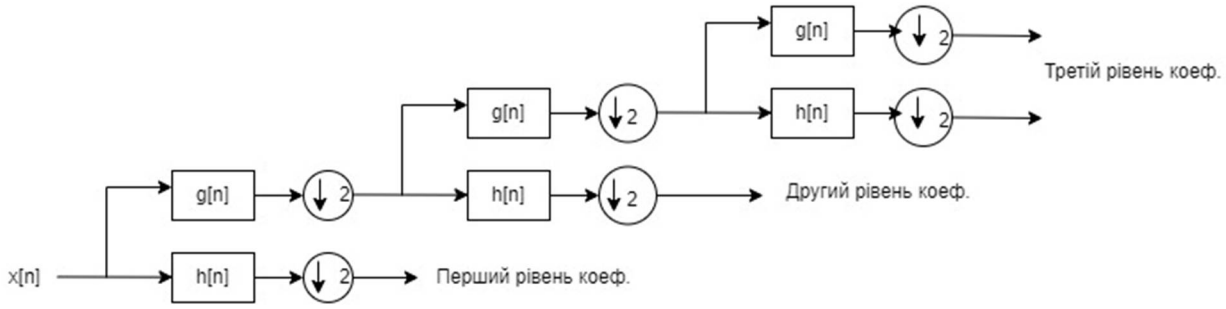


Рисунок 2.4 – 3-рівневий фільтр-банк

На кожному рівні сигнал розкладається на низькі та високі частоти. Через процес розкладання вхідний сигнал має бути кратним 2^n, де n - кількість рівнів. Наприклад, сигнал із 32 вибірками, діапазон частот від 0 до f_n і 3 рівні розкладання виробляють 4 вихідні шкали [26]:

Таблиця 2.1 – Рівні розкладання сигналу із 32 вибірками

Рівень	Частоти	Вибірки
3	0 до $f_n/8$	4
	$f_n/8$ до $f_n/4$	4
2	$f_n/4$ до $f_n/2$	8
1	$f_n/2$ до f_n	16

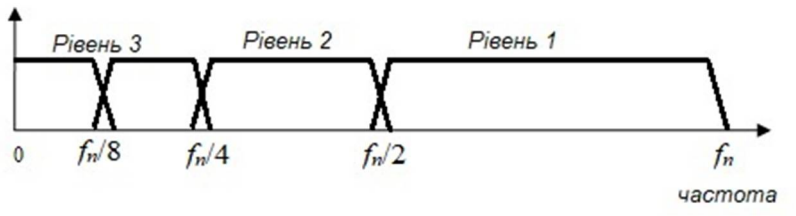


Рисунок 2.5 – Представлення ДВП у частотній області

Реалізацію банку фільтрів вейвлетів можна розглядати як обчислення вейвлет-коефіцієнтів для дискретного набору дочірніх вейвлетів, основаних на заданому материнському вейвлеті *ψ(t)*. У випадку ДВП материнський вейвлет зміщується та масштабується за ступенями двох [26]

$$\psi_{j,k}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-k2^j}{2^j}\right), \tag{2.14}$$

де *j* - параметр масштабу, *k* - параметр зсуву (обидва є цілими числами).

Реалізація фільтр-банку ДВП в деяких випадках приймає тільки *O(N)*, якщо порівнювати з *O(NlogN)* для ШПФ. Якщо довжина *g[n]* та *h[n]* не залежить від *N*, тоді *x*h* та *x*g* кожен займає *O(N)* часу. Банк вейвлет-фільтрів виконує кожну з цих двох *O(N)* згорток, після чого розділяє сигнал на дві гілки, які мають розмір *N/2*. Правда він лише поетапно розділяє тільки верхню, згорнуту з *g[n]*; ШПФ поетапно розділяє обидві гілки. Це призводить до наступного рекурентного співвідношення [26]

$$T(N) = 2N + T(\frac{N}{2}) \tag{2.15}$$

що призводить до часу *O(N)* для всієї операції.

Локальність вейвлетів разом зі складністю *O(N)* забезпечує можливість обчислення перетворення в режимі онлайн (на основі потоку). Ця властивість контрастує з швидким перетворенням Фур'є, яке вимагає доступу до всього сигналу одночасно. Це також стосується багатомасштабного перетворення та багатовимірного перетворення [27].

2.3 Вейвлет Добеші

Вейвлети Добеші є сімейством ортогональних вейвлетів, які визначають дискретне вейвлет-перетворення та мають максимальну кількість моментів зникнення для конкретно вибраної опори.

Загалом вейвлет-перетворення Добеші визначаються як і Хаара: обчисленням поточних середніх і різниць за допомогою скалярних добутків із сигналами масштабування та вейвлетами [28].

Вейвлет Хаара - найпростіший тип вейвлета. В дискретній формі вони є перетворенням Хаара, яке в свою чергу є прототипом для інших вейвлет-перетворень. Вейвлет-перетворення Хаара розкладає дискретний сигнал на два підсигнали половини його довжини: один підсигнал – поточне середнє або тренд, інший – поточна різниця або коливання [29]. Материнська вейвлет-функція $\psi(t)$ вейвлета Хаара може бути описана як [30]

$$\psi(t) = \begin{cases} 1 & 0 \le t \le \frac{1}{2}, \\ -1 & \frac{1}{2} \le t \le 1, \\ 0 & \text{інакше.} \end{cases} \tag{2.16}$$

Вейвлети Добеші вибираються таким чином, щоб мати максимальну кількість A моментів, які зникають, для заданої ширини опори (числа коефіцієнтів) $2A$ [31]. Використовують дві схеми найменування: DN – використовує довжину або ж кількість відводів, db$A$ – стосується кількості моментів, які зникають. Тобто D2 та db1 є одним і тим самим вейвлет-перетворенням.

Вейвлет-перетворення Добеші D4 має чотири коефіцієнти функції масштабування

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}. \tag{2.17}$$

Якщо вихідний набір містить *N* значень, на етапі вейвлет-перетворення функція масштабування застосовується для отримання *N/2* згладжених значень. У впорядкованому вейвлет-перетворенні ці згладжені значення розміщуються в нижній половині вхідного вектора з *N* елементами.

Нижче наведено як вирахувати та чому дорівнюють значення коефіцієнтів вейвлет-функції

$$g_k = (-1)^k h_{N-1-k}, \tag{2.18}$$

$$g_0 = h_3, g_1 = -h_2, g_2 = h_1, g_3 = -h_0. \tag{2.19}$$

Якщо вихідний набір даних містить *N* значень, вейвлет-функція застосовується для отримання *N/2* відмінностей, які відображають зміни в даних. У впорядкованому вейвлет-перетворенні коефіцієнти вейвлетів зберігаються в верхній половині вхідного вектора з *N* елементами.

3 КВАНТОВА ОБРОБКА СИГНАЛУ: РЕАЛІЗАЦІЯ ТА ПОРІВНЯННЯ

3.1 PennyLane & Google Colab

PennyLane – це бібліотека програмного забезпечення з відкритим кодом, розроблена для квантового машинного навчання та квантових обчислень. Це дозволяє користувачам створювати, тренувати та запускати квантові схеми на різних квантових пристроях і симуляторах. Основними характеристиками є:

- легке інтегрування квантових алгоритмів з класичними методами машинного навчання;
- є можливість запускати програми на різних квантових пристроях та симуляторах без зміни коду;
- зрозумілий інтерфейс, що спрощує процес створення та навчання квантових моделей;
- присутня активна спільнота, безліч посібників, документації та навчальних матеріалів, що допомагає швидко освоїти технологію;
- вбудовані можливості автоматичної диференціації полегшують оптимізацію параметрів квантових моделей;
- працює з такими бібліотеками, як PyTorch, TensorFlow, JAX і NumPy.

Google Colab – хмарне середовище для роботи з Jupyter Notebook, яке дозволяє виконувати обчислення, аналіз даних, машинне навчання та інші завдання без необхідності налаштування власної інфраструктури. Одна з основних переваг – безкоштовний доступ до графічних процесорів, що дозволяє значно прискорити виконання складних обчислень і моделей машинного навчання. Крім того, Google Colab підтримує популярні

бібліотеки Python, має зручний інтерфейс і забезпечує можливість спільної роботи над проєктами.

3.2 Кодова реалізація та пояснення

Перед початком роботи потрібно встановити та підключити бібліотеки `pennylane` та `numpy`, присвоївши їм `qml` та `np`, відповідно, для подальшої зручності.

```
!pip install pennylane --upgrade
!pip install numpy

import numpy as np
import pennylane as qml
```

Бібліотека pennylane для квантового машинного навчання та дозволяє створювати і виконувати квантові програми. Numpy для роботи з масивами та матрицями в Python. А тепер починаємо творити мистецтво.

Для початку треба визначити коефіцієнти для вейвлет-перетворення Добеші D4. Перетворення використовує коефіцієнти апроксимації (h) та коефіцієнти деталізації (g). Вони розраховуються за формулами, які прописані в другому розділі в п.п. 2.3 Вейвлет Добеші. Якщо коротко, то h спрощують сигнал, зберігаючи при цьому основні ознаки. А от g визначають, як виділити деталі сигналу, зокрема, зміни в амплітуді або частоті.

```
# Коефіцієнти для апроксимації вейвлет-перетворення D4
h0 = (1 + np.sqrt(3)) / (4 * np.sqrt(2))
h1 = (3 + np.sqrt(3)) / (4 * np.sqrt(2))
h2 = (3 - np.sqrt(3)) / (4 * np.sqrt(2))
h3 = (1 - np.sqrt(3)) / (4 * np.sqrt(2))

# Коефіцієнти для деталізації вейвлет-перетворення D4
g0 = (1 - np.sqrt(3)) / (4 * np.sqrt(2))
```

```
g1 = (-3 - np.sqrt(3)) / (4 * np.sqrt(2))
g2 = (3 - np.sqrt(3)) / (4 * np.sqrt(2))
g3 = (-1 - np.sqrt(3)) / (4 * np.sqrt(2))

approximation_filter = np.array([h0, h1, h2, h3])
detail_filter = np.array([g0, g1, g2, g3])
```

Після чого об'єднуємо коефіцієнти в масиви `approximation_filter` та `detail_filter`, відповідно.

Надалі прописуємо кількість кубітів для вейвлет-перетворення та Фур'є, а також довжину сигналу (2^n).

```
wavelet_qubits = 4
qft_qubits = 3
signal_length = 16
```

В наступних рядках коду створюємо квантові пристрої, зокрема

```
dev_qft = qml.device('default.qubit', wires=qft_qubits)
dev_d4 = qml.device("default.qubit", wires=wavelet_qubits)
```

Перший рядок створює квантовий пристрій для перетворення Фур'є. В цьому рядку функція `qml.device` використовуєься для створення квантового пристрою, що буде модулювати квантовий комп'ютер. `default.qubit` використовує симуляцію кубітів на класичному комп'ютері, а `wires=qft_qubits` вказує кількість кубітів, яка буде використовуватися для виконання квантового перетворення Фур'є.

Другий рядок по аналогії з першим створює квантовий пристрій для виконання вейвлет-перетворення D4, тільки тут кількість кубітів позначена як `wires=wavelet_qubits`.

В даному фрагменті коду визначається квантова схема для виконання Фур'є. Треба зауважити, що в pennylane квантова реалізація Фур'є вже присутня, тому будемо використовувати готові інструменти.

```
@qml.qnode(dev_qft)
def circuit_qft(basis_state):
    qml.BasisState(basis_state, wires=range(qft_qubits))
    qml.QFT(wires=range(qft_qubits))
    return qml.state()
```

Позначення `@qml.qnode(dev_qft)` є декоратором, який позначає функцію як квантову схему. Така схема буде виконуватися на віртуальному квантовому пристрої, створеному раніше (`dev_qft`), що в свою чергу містить `qft_qubits` кубітів. Визначення функції `def circuit_qft(basis_state)`, яка описує квантову схему. Вхідним параметром тут є `basis_state` – список, який задає базовий стан для квантових кубітів, на яких буде виконуватися перетворення Фур'є. У процесі роботи функції будемо задавати цей базовий стан. Операція `qml.BasisState(basis_state, wires=range(qft_qubits))` готує кубіти до заданого початкового стану. Кубіти в свою чергу можуть бути в різних станах: 0 або 1, або в суперпозиції цих станів, тобто у комбінації цих двох станів, що дозволяє йому бути в декількох станах одночасно. `wires=range(qft_qubits)` вказує з яких кубітів складається схема. `qml.QFT(wires=range(qft_qubits))` – операція квантового перетворення Фур'є. Вона перетворює базовий стан кубітів у суперпозицію всіх можливих значень за допомогою різних квантових вентилів. В `return qml.state()` команда `qml.state()` повертає квантовий стан кубітів після виконання операції квантового перетворення Фур’є. Вона дає доступ до амплітуд усіх можливих станів квантової системи (тобто до всіх можливих результатів вимірювання кубітів).

А тепер переходимо до кодової реалізації перетворення вейвлета Добеші D4. Розглянемо наступний фрагмент коду

```
@qml.qnode(dev_d4)
def d4_wavelet_transform(input_state):
    qml.MottonenStatePreparation(input_state,
wires=range(wavelet_qubits))
```

Декоратор `@qml.qnode(dev_d4)` визначає функцію як квантову схему, яка буде виконуватися на квантовому пристрої (`dev_d4`). Функція `def d4_wavelet_transform(input_state):` описує вейвлет-перетворення D4 для обробки вхідного стану `input_state`, який є вхідним параметром – масивом, що представляє собою класичний сигнал, який буде квантово підготовлений для

обробки. Операція `qml.MottonenStatePreparation(input_state, wires=range(wavelet_qubits))` готує вхідний сигнал для квантової системи за допомогою методу MottonenStatePreparation. Після застосування MottonenStatePreparation вхідний класичний сигнал `input_state` буде перетворений на квантовий стан і збережений у `wavelet_qubits`. Кількість кубітів, яка використовуватиметься для підготовки квантового стану – `wires=range(wavelet_qubits)`.

Приступаємо до першого етапу, а саме апроксимації.

```
for i in range(0, wavelet_qubits - 1, 2):
        qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
        qml.CNOT(wires=[i, i + 1])
        qml.RY(2 * np.arctan(approximation_filter[1]), wires=i + 1)


    for i in range(1, wavelet_qubits - 1, 2):
        qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
        qml.CNOT(wires=[i, i + 1])
        qml.RY(2 * np.arctan(approximation_filter[1]), wires=i + 1)
```

Перший цикл обробляє елементи з парними індексами.

Квантовий вентиль RY виконує обертання кубіта на кут `2 * np.arctan(approximation_filter[0])` навколо осі Y. `approximation_filter[0]` – це перший коефіцієнт з масиву фільтра, який застосовується для апроксимації. За допомогою функції `np.arctan` обчислюється кут обертання. Цей вентиль змінює квантовий стан кубіта з індексом *i*.

Квантовий вентиль CNOT (Controlled NOT) виконує операцію перевернення на кубіті з індексом $i + 1$, якщо кубіт з індексом *i* знаходиться в стані 1. Така операція забезпечує квантовий зв'язок між сусідніми кубітами. Якщо кубіт з індексом *i* перебуває в стані 1, кубіт з індексом $i + 1$ змінить свою амплітуду. Це є важливою частиною процесу перетворення, оскільки дозволяє передавати інформацію між кубітами.

Після виконання операції CNOT на кубіті з індексом $i + 1$, вентиль `qml.RY(2 * np.arctan(approximation_filter[1]), wires=i + 1)` виконує ще одне

обертання на цьому кубіті. Для цього кубіта обчислюється новий кут обертання, але вже з другим коефіцієнтом з масиву фільтра: `approximation_filter[1]`. Подібно до першого обертання, це змінює квантовий стан кубіта з індексом $i + 1$, залежно від апроксимації сигналу.

Другий цикл працює за тією ж схемою, тільки з непарними індексами. А тепер підсумуємо.

Кожен парний/непарний кубіт обробляється через обертання, яке змінює його стан відповідно до фільтра для апроксимації. Потім між кожною парою кубітів (парні/непарні індекси та їхні наступні сусіди) виконується операція CNOT, щоб "передати" або коректувати квантову інформацію. Після цього, другий кубіт в парі (з індексом $i + 1$) також обертається згідно з другим коефіцієнтом фільтра для апроксимації.

Тепер приступаємо до другого етапу перетворення – деталізації.

```
for i in range(0, wavelet_qubits - 2, 2):
    qml.CNOT(wires=[i + 1, i + 2])
    qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
    qml.CNOT(wires=[i, i + 1])


for i in range(1, wavelet_qubits - 2, 2):
    qml.CNOT(wires=[i + 1, i + 2])
    qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
    qml.CNOT(wires=[i, i + 1])


return qml.state()
```

Як і в блоці апроксимації, починаємо з елементів з парними індексами.

Операція CNOT змінює стан кубіта з індексом $i + 2$ тільки в тому випадку, якщо кубіт з індексом $i + 1$ знаходиться в стані 1. У контексті деталізації це створює взаємодію між сусідніми кубітами, що дозволяє коректувати їх стани в залежності один від одного, відповідно до фільтра деталізації.

Вентиль RY виконується на кубіті з індексом *i + 2* і обертає його на кут `2*arctan(detail_filter[0])`. Це обертання кубіта в просторі його станів навколо осі Y, і воно коригує амплітуду кубіта з індексом *i + 2* згідно з коефіцієнтом фільтра для деталізації (в даному випадку перший коефіцієнт з масиву `detail_filter`). Оскільки говоримо про деталізацію, цей крок дозволяє зберегти високочастотні компоненти сигналу, які відповідають "дрібним деталям" сигналу.

Операція `qml.CNOT(wires=[i, i + 1])` контролює кубіт *i + 1* за допомогою кубіта *i*. Якщо кубіт *i* перебуває в стані 1, то кубіт *i + 1* буде обернений. Цей вентиль є необхідним для коригування стану кубітів після виконання операцій обертання, гарантуючи правильну передачу інформації між кубітами, щоб деталі сигналу (високочастотні компоненти) були коректно зафіксовані.

Другий цикл працює за тією ж схемою, тільки з непарними індексами. Наприкінці функції повертається квантовий стан системи за допомогою `qml.state()`.

Обертання по осі Y пов'язано з особливістю реалізації вейвлет-перетворення. Вейвлетні перетворення, як правило, передбачають коригування амплітуд і фаз сигналу, що добре досягається через вентиль RY.

Створюємо вхідний сигнал.

```
t = np.linspace(0, 2 * np.pi, signal_length)
input_signal = np.sin(t)
```

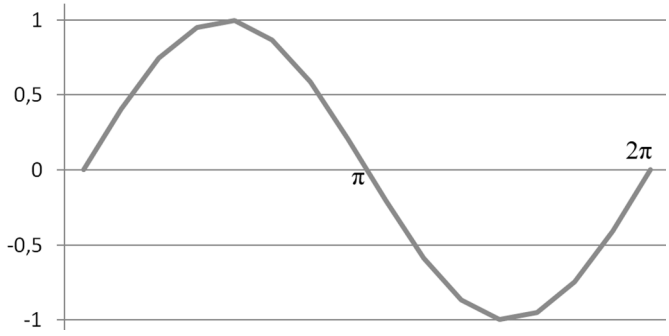


Рисунок 3.1 – Вхідний сигнал sin(t)

Функція `np.linspace(start, stop, num)` з бібліотеки numpy, яка створює масив чисел, рівномірно розподілених між значеннями start і stop. `start=0` є початоком інтервалу, `stop=2 * np.pi` є кінецем інтервалу. В той же час, це кут, що відповідає одному повному колу в радіанах. `num=signal_length` визначає кількість точок, на яких буде створено сигнал.

Таким чином, ця команда створює масив *t*, що містить рівномірно розподілені значення від 0 до 2π, причому загальна кількість значень визначається параметром `signal_length`. Якщо, наприклад, `signal_length` = 16, то масив буде містити 16 значень від 0 до 2π.

Масив `input_signal` містить значення синуса для кожного з елементів масиву *t*. Якщо значення в *t* рівномірно розподілені точки від 0 до 2π, то `input_signal` буде представляти одну повну хвилю синусоїдального сигналу, де значення будуть коливатися між -1 і 1.

Перед тим, як пропускати сигнал через перетворення, його потрібно нормалізувати. Це процес, коли масштабуємо вектор так, щоб його норма, тобто довжина, стала рівною 1. Це дозволяє працювати з сигналом у масштабі в квантових обчисленнях, так як необхідно підтримувати коректність обчислень на всіх етапах.

```
input_signal_normalized = input_signal / np.linalg.norm(input_signal)
```

Позначення `np.linalg.norm(input_signal)` обчислює норму вектора вхідного сигнала. В цьому випадку `input_signal` – це масив значень синусоїди:

$$\|input\_signal\| = \sqrt{\sum_{i=0}^{n-1} x_i^2}, \tag{3.1}$$

де x_i - це елементи сигналу, а n - кількість елементів. Дане значення представляє довжину вектора в просторі.

Після обчислення норми сигналу, цей вираз ділить кожен елемент сигналу на норму. Тобто, масштабуємо вектор так, щоб його норма стала рівною 1. Новий сигнал `input_signal_normalized` буде мати одиничну норму.

```
# Додавання нулів до сигналу, щоб довжина була ступенем двійки
padded_signal_length = 16
input_signal_padded = np.concatenate((input_signal_normalized, np.zeros(padded_signal_length - len(input_signal_normalized))))
```

Хочемо, щоб довжина сигналу становила 16, тому встановлюємо таке значення змінній. Оскільки вихідний сигнал може мати довжину менше ніж 16 елементів, потрібно додати достатньо нулів, щоб досягти бажаної довжини. Ось як це працює: `len(input_signal_normalized)` – поточна довжина сигналу після нормалізації, а `padded_signal_length - len(input_signal_normalized` – це різниця між бажаною довжиною (16) та фактичною довжиною сигналу. Це дає кількість нулів, які потрібно додати. Функцію `np.zeros` використовуємо, щоб створити масив з нулів, які треба додати. Функція `np.concatenate` об'єднує два масиви: нормалізований сигнал та масив нулів, який було створено тільки-но. В результаті отримаємо новий масив, довжина якого дорівнює 16.

Приступаємо до виконання перетворень. Почнемо з Фур'є. Для початку створюємо масив, який містить три елементи (так як у нас три кубіти), які дорівнюють нулю. Це основний квантовий стан і всі кубіти знаходяться у стані $|0\rangle$.

```
basis_state = [0, 0, 0]  # Основний стан для QFT
frequency_index=np.argmax(np.abs(np.fft.fft(input_signal_normalized)))
if frequency_index < 2**qft_qubits:
    basis_state[frequency_index] = 1


qft_result = circuit_qft(basis_state)
```

Після чого виконуємо аналіз сигналу за допомогою `frequency_index = np.argmax(np.abs(np.fft.fft(input_signal_normalized)))`.

Функція `np.fft.fft` виконує дискретне перетворення Фур'є для класичного сигналу, перетворюючи його з часової області в частотну. Результатом цього перетворення є комплексний масив, де кожен елемент відповідає амплітуді на певній частоті. Оскільки результат ДПФ є

комплексним числом, застосовуємо `np.abs()`, щоб отримати амплітуду кожної частоти, тобто модуль комплексних чисел. Це дає нам реальні величини, які описують інтенсивність кожної частоти у сигналу. Функція `np.argmax()` знаходить індекс максимального значення в масиві. В цьому випадку, шукаємо індекс, який відповідає найбільш вираженій частоті, тобто частоті з найбільшою амплітудою у сигналу. Таким чином, `frequency_index` міститиме індекс цієї найбільш вираженої частоти в спектрі сигналу.

На етапі, де виникає умова (третій рядок фрагменту коду), перевіряємо можливість представити знайдену частоту в межах доступних квантових бітів.

Змінна `qft_qubits` визначає кількість кубітів, які використовуємо для квантового перетворення Фур'є. Якщо маємо 3 кубіти, як у цьому випадку, то $2^3=8$, що означає, що ми можемо представити частоти в діапазоні від 0 до 7. Тобто, можемо закодувати максимум 8 різних частот.

Перевірка `if frequency_index < 2**qft_qubits:` для того, щоб переконатися, що індекс частоти, яку знайшли, не перевищує максимально допустимий індекс, який можна закодувати за допомогою доступних кубітів.

При виконанні умови, оновлюється `basis_state`, встановлюючи елемент у позиції `frequency_index` на 1. Це означає, що вибрали частоту з максимальним значенням амплітуди для подальшого квантового перетворення.

Нарешті відбудеться квантове перетворення Фур'є (`qft_result = circuit_qft(basis_state)`). Функція `circuit_qft(basis_state)` викликає квантове перетворення на основі стану `basis_state`. Вона була визначена раніше як квантовий вузол (qnode), що реалізує операції квантового перетворення Фур’є на заданому стані кубітів. У результаті функція повертає новий квантовий стан, який зберігається в змінній `qft_result`.

Тепер переходимо до виконання перетворення вейвлета Добеші D4.

```
quantum_state_d4 = d4_wavelet_transform(input_signal_padded)
```

Викликається функція, яку раніше прописували для сигналу `input_signal_padded`. Це той самий сигнал, який був нормалізований і доповнений нулями, щоб його довжина була ступенем двійки.

В нижче наведеному фрагменті коду створюємо функцію для аналізу квантового стану для того, щоб отримати коефіцієнти, які описують стан кубітів.

```
def extract_coefficients(quantum_state):
    amplitudes = np.abs(quantum_state)
```

Вхідний параметр функції `quantum_state` є квантовим станом (результат виконання квантового перетворення Фур'є). Він зазвичай є вектором комплексних чисел. Кожен елемент цього вектора є амплітудою для відповідного квантового стану. Наприклад, якщо стан має два кубіти, квантовий стан може бути представлений як вектор з 4 елементів (для 4 можливих комбінацій кубітів: $|00\rangle$,$|01\rangle$,$|10\rangle$,$|11\rangle$).

Функція `np.abs(quantum_state)` перетворює вектор комплексних амплітуд на вектор ймовірностей (модулів амплітуд для кожного стану), де кожен елемент вектора представляє модуль амплітуди для відповідного квантового стану в суперпозиції. Наприклад, якщо квантовий стан виглядає так:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle, \tag{3.2}$$

то виклик `np.abs(quantum_state)` дасть:

$$[\ |\alpha|, |\beta|, |\gamma|, |\delta|\ ]. \tag{3.3}$$

Ось ці модулі амплітуд і є коефіцієнтами, які аналізуємо для подальшого розбиття квантового стану на окремі компоненти: апроксимацію і деталізацію. Функція `extract_coefficients` дозволяє розділити ці коефіцієнти.

Далі розподіляємо амплітуди квантового стану на два набори коефіцієнтів: апроксимаційні та деталізаційні. Таким чином, виділяємо різні частини інформації, яку отримали після перетворення.

```
mid = len(amplitudes) // 2
    approximation_coeffs = amplitudes[:mid]
    detail_coeffs = amplitudes[mid:]


    return approximation_coeffs, detail_coeffs
```

За допомогою першого рядка визначаємо точку, де масив амплітуд буде розділений на дві частини: одна для апроксимаційних коефіцієнтів, а інша для деталізаційних. `len(amplitudes)` – це довжина масиву амплітуд, тобто кількість елементів у квантовому стані (наприклад, для 4 кубітів буде 16 елементів). У другому рядку відбираємо першу половину амплітуд з масиву `amplitudes`, що представляють собою апроксимаційні коефіцієнти. Тобто виділяємо низькочастотні компоненти сигналу (основну інформацію про сигнал). У третьому рядку відбираємо другу половину амплітуд з масиву `amplitudes`, що представляє собою деталізаційні коефіцієнти. Відповідають за високочастотні компоненти сигналу, які містять малі деталі, шуми чи швидкі зміни в сигналі. В кінці повертаємо два набори коефіцієнтів.

В наступному рядку викликаємо функцію `extract_coefficients`, щоб отримати коефіцієнти з квантового стану, який було обчислено після виконання перетворення D4.

```
approximation_coeffs, detail_coeffs = extract_coefficients(quantum_state_d4)
```

Наприкінці виводимо на екран результати обчислень.

```
print("QFT Result:", qft_result)
print("Quantum State (D4 Wavelet):", quantum_state_d4)
print("Approximation Coefficients:", approximation_coeffs)
print("Detail Coefficients:", detail_coeffs)
```

```
QFT Result: [ 0.35355339+0.j          0.        +0.35355339j -0.35355339+0.j
  0.        -0.35355339j  0.35355339+0.j          0.        +0.35355339j
 -0.35355339+0.j          0.        -0.35355339j]
Quantum State (D4 Wavelet): [-0.18151887+8.17480008e-18j -0.2575212 +1.19275975e-17j
 -0.3929905 +1.11635249e-17j  0.15036484-5.67434333e-18j
  0.44763187-1.67789778e-17j  0.08147727-2.55938875e-18j
  0.10865971-4.28798240e-18j  0.62190698-2.73113115e-17j
 -0.01573924+5.07352991e-18j -0.25104173+6.18044184e-17j
 -0.01336068+1.04213696e-17j  0.02632096-3.95632418e-18j
  0.15882805-2.37694630e-17j  0.01458512-5.65417615e-18j
 -0.08876239+2.07523966e-17j -0.14365144+3.29104707e-17j]
Approximation Coefficients: [0.18151887 0.2575212  0.3929905  0.15036484 0.44763187 0.08147727
 0.10865971 0.62190698]
Detail Coefficients: [0.01573924 0.25104173 0.01336068 0.02632096 0.15882805 0.01458512
 0.08876239 0.14365144]
```

Рисунок 3.2 – Результат виконання коду

Розглянемо результат детальніше. Почнемо з результату виконання квантового перетворення Фур'є.

Результат є вектором комплексних чисел, що представляє квантовий стан після виконання перетворення. Вектор має 8 елементів, що відповідає 2^3 (3 кубіти). Це свідчить про те, що перетворення виконувалось на трьох кубітах. Кожен елемент вектора є амплітудою відповідного стану. Наприклад, 0.35355339+0.j – амплітуда стану $|000\rangle$, 0.+0.35355339j – амплітуда стану $|001\rangle$. Амплітуда визначає ймовірність знаходження системи в певному стані після вимірювання. Ймовірність P для стану $|i\rangle$ обчислюється як квадрат модуля амплітуди

$$P(i) = |a_i|^2. \tag{3.4}$$

Наприклад, P(000) = $|0.35355339|^2$=0.125. Тобто, якщо виміряти систему, є 12,5% ймовірність отримати стан $|000\rangle$.

Комплексні числа в результаті можуть мати реальні та уявні частини, але для ймовірностей важливий лише модуль. Значення 0.35355339+0.j та 0.+0.35355339j можуть вказувати на стани, але з різними фазовими зсувами. Також видно, що значення повторюються в певній симетрії. Це може свідчити про певні закономірності у вхідному сигналі.

Тепер переходимо до результатів перетворення вейвлета Добеші D4. Після виконання перетворення вейвлету отримали вектор комплексних

чисел, який описує квантовий стан. Як і у випадку з перетворенням Фур'є, кожен елемент вектора є амплітудою відповідного квантового стану, причому це комплексні числа. Реальна частина (наприклад, -0.18151887) вказує на величину амплітуди, а уявна частина (наприклад, +8.17480008e-18j) – на фазу цього стану. Як і у випадку з перетворенням Фур'є, кожен елемент вектора є комплексним числом, яке визначає ймовірність знаходження системи в певному стані після вимірювання.

Якщо порівнювати коефіцієнти апроксимації та деталізації, можна побачити, що загальні величини для апроксимації (перша частина вектора) значно більші, ніж для деталізації. Сигнал з такою структурою, ймовірно, має плавні коливання або невеликі зміни (малий рівень високочастотних коливань), що і видно з низьких коефіцієнтів деталізації.

Величини уявної частини дуже малі (наприклад, 8.17480008e-18j, 1.19275975e-17j), що вказує на те, що фази в основному є незначними або майже рівними нулю. У багатьох випадках квантові системи мають значні фазові зміщення між різними станами. А в цьому – це може свідчити про те, що система перебуває у відносно простому стані, де фази між компонентами сигналу майже відсутні.

Звертаючи увагу на значення коефіцієнтів, можна помітити певну симетрію в їх розподілі. Наприклад, найбільші коефіцієнти апроксимації знаходяться в середині списку (наприклад, 0.44763187, 0.3929905), що може свідчити про певну закономірність або структуру вхідного сигналу, яку перетворення вейвлету вдало зберегло. Симетрія у коефіцієнтах може бути результатом регулярних властивостей сигналу, таких як періодичність чи наявність певних шаблонів у даних, що квантове перетворення вейвлету виявило.

Деталізація відповідає за вищі частоти, але можна помітити, що в даному випадку деякі коефіцієнти деталізації, як наприклад 0.25104173, мають порівняно великі значення. Це може вказувати на те, що у сигналу є

певні суттєві зміни на середніх частотах, що є важливим аспектом для розуміння, як квантове перетворення зберігає ці властивості.

Навіть незначні коефіцієнти деталізації, такі як 0.01573924, можуть містити важливу інформацію для відновлення сигналу або для виявлення шумів чи швидких змін. Це також може свідчити про те, що квантове перетворення здатне виділити навіть дуже тонкі зміни у даних, що може бути корисним для задач, де важливо виділити малі коливання в сигналі.

Є одна важлива деталь, яку варто проговорити. З класичної точки зору, перетворення вейвлета зазвичай зменшує розмірність сигналу вдвічі при кожному рівні перетворення. Однак, вихідні коефіцієнти квантового перетворення можуть не зменшуватися через природу квантових обчислень. Квантові стани можуть представляти суперпозицію (стан може одночасно мати багато значень), тому кількість значень у виході може відображати більш комплексну структуру сигналу.

3.3 Порівняння запропонованих перетворень

Перейдемо до порівняння двох квантових методів перетворення сигналів: квантове перетворення Фур'є та квантове вейвлет-перетворення Добеші (D4). Обидва використовувались для аналізу сигналу, але підходи до їхнього виконання й аналізу суттєво відрізняються. Тому порівняння дозволить краще зрозуміти їхні переваги, обмеження та області застосування.

Почнемо з принципу роботи. Квантове перетворення Фур'є є квантовим аналогом класичного перетворення Фур'є, яке використовується для аналізу частотного складу сигналу. Принцип роботи полягає в тому, що сигнал, заданий у часовій області, перетворюється в частотну область за допомогою квантового оператора, що виконує операції над квантовими кубітами. Основні етапи виконання квантового перетворення Фур'є:

– активізація кубітів у певному початковому стані (наприклад, у стані, що представляє сигнал);

– виконання квантових операцій, зокрема перетворення Фур'є;

– результат перетворення – новий квантовий стан, у якому можна виділити частотні компоненти сигналу.

Принципова перевага квантового перетворення полягає в тому, що воно дозволяє здійснити перетворення за час, який залежить лінійно від кількості кубітів, на відміну від класичного підходу, де складність обчислення зростає лінійно з розміром сигналу.

Вейвлет-перетворення Добеші (D4) є одним з видів вейвлет-перетворень, яке дозволяє не лише виявити частоти сигналу, але й аналізувати зміни на різних часових масштабах. Це особливо корисно для сигналів з локальними змінами або з різною частотною структурою. Вейвлети працюють в тимчасово-частотній області, дозволяючи отримувати інформацію про локальні частоти сигналу, що неможливо досягти за допомогою лише квантового перетворення Фур’є.

Вейвлет-перетворення Добеші здійснюється через кілька етапів.

Перший етап – апроксимування, де основна частина сигналу аналізується за допомогою фільтру апроксимації.

Другий етап – деталізація, де виявляються більш дрібні коливання сигналу через фільтр деталізації.

Цей процес можна інтерпретувати як поетапне виявлення сигналів на різних рівнях масштабів, зберігаючи як глобальну, так і локальну інформацію.

Переходимо до особливості реалізації. У класичному випадку перетворення Фур'є складається з низки математичних операцій, таких як множення на експоненціальну функцію і сума таких множень для кожної точки сигналу. Квантове перетворення Фур'є реалізується через набір квантових гейтів, що виконують операції на кубітах.

Hadamard-гейт – базовий квантовий гейт, який часто використовується для створення рівномірного суперпозиційного стану.

Rotation-гейти – гейти, які виконують фазовий зсув над кубітами. Вони застосовуються для отримання потрібних частотних компонентів у процесі перетворення.

В реалізації квантового перетворення Фур'є у коді немає явних викликів Hadamard-гейтів або Rotation-гейтів, але ці операції автоматично виконуються, коли використовуєм `qml.QFT(wires=range(qft_qubits))`. Це вбудована функція в PennyLane, яка автоматично застосовує ці гейти до кубітів у зазначеній лінії коду. Механізм складається з серії Hadamard-операцій і фазових зсувів, що на рівні коду представлені цією функцією.

Квантове вейвлет-перетворення Добеші реалізується через фільтрацію сигналу, яка відокремлює низькочастотні (апроксимуючі) та високочастотні (деталізуючі) компоненти сигналу.

У коді це реалізовано через серію квантових операцій, таких як RY-гейти (використовуються для створення потрібних фазових зсувів) та CNOT-гейти (що виконують операції заплутування між кубітами). На кожному етапі виконуються маніпуляції з кубітами, щоб виділити частини сигналу, відповідні до апроксимації та деталізації. Для апроксимації використовуються RY-гейти, щоб обробляти парні та непарні індекси сигналу (через відповідні фазові зсуви). Для деталізації застосовуються CNOT-гейти для заплутування кубітів та подальшої обробки сигналу на більш дрібних масштабах.

Кількість кубітів в обох перетвореннях визначає, скільки елементів сигналу можна обробити одночасно, а сама реалізація алгоритмів на квантових комп'ютерах дозволяє зменшити складність обчислень в порівнянні з класичними методами.

Тепер щодо області застосування. Квантове перетворення Фур'є є ідеальним для аналізу сигналів у частотній області, що робить його надзвичайно корисним для задач, де важливо виявляти основні частоти в

складних сигналах. Аналіз звукових хвиль (наприклад, для музики або мови) за допомогою КПФ дозволяє швидко визначити основні частоти, що використовуються в музиці, або навіть для виявлення аномалій у звукових даних. У телекомунікаціях часто необхідно визначати основні частоти для передачі даних через різні канали. КПФ може бути використане для швидкої оптимізації пропускної здатності каналів. У медицині можна застосовувати для обробки сигналів з медичних приладів, таких як ЕКГ чи ЕЕГ, для виділення частотних компонентів, що можуть вказувати на патології або аномалії.

D4 вейвлет-перетворення є потужним інструментом для розкладу сигналу на багаторівневі компоненти, що особливо корисно для аналізу складних сигналів з локальними змінами. Вейвлети, зокрема D4, використовуються для стиснення та покращення зображень [36], що дозволяє виділяти ключові деталі, зменшуючи шуми. Це може бути корисно для обробки медичних зображень, де важливо зберегти деталізацію при зменшенні обсягу даних. Вейвлет-перетворення дозволяє розглядати зміни температури, тиску або інших погодних показників на різних часових масштабах, що важливо для прогнозування та моделювання кліматичних явищ. Аналіз фінансових даних, таких як зміни цін на акції чи валютні курси, може виграти від вейвлет-аналізу. За допомогою D4 можна виділити як довгострокові тренди, так і короткострокові коливання на фінансових ринках.

Квантові методи також можуть бути використані для антипідробки сигналів або для захисту від атак [37] на інформаційні системи, де перетворення Фур'є та вейвлети дозволяють швидко виявляти аномалії в сигналах, що можуть свідчити про втручання. Оскільки квантові методи можуть забезпечити високу стійкість до зламування, їх можна використовувати для шифрування сигналів [38] та передачі даних з використанням квантових алгоритмів.

Викладемо все вище сказане у таблицю 3.1.

Таблиця 3.1 – Порівняння квантового перетворення Фур'є з квантовим вейвлет-перетворенням Добеші

Характеристика	Квантове перетворення Фур’є	Квантове вейвлет-перетворення Добеші
Призначення	Перетворення сигналу з часової області в частотну.	Розкладання сигналу на апроксимаційні та деталізуючі компоненти.
Тип аналізу	Частотний аналіз (перехід у частотну область).	Багатошарове розкладання (аналог часу-частоти).
Основна операція	Квантове перетворення Фур'є.	Вейвлетне перетворення (локалізоване в часі та частоті).
Тип сигналу	Підходить для перетворення сигналів з чітко вираженими частотами.	Підходить для обробки сигналів з різною частотною складністю та масштабами.
Алгоритмічні операції	Використовує Hadamard-гейти та фазові гейти для створення суперпозицій.	Використовує RY-гейти для фазових зсувів та CNOT-гейти для заплутування.
Простота реалізації	Вбудована функція в PennyLane (qml.QFT), автоматичне застосування гейтів.	Реалізація через побудову специфічних квантових операцій (RY та CNOT).
Основні ресурси	Кількість кубітів визначає розмір сигналу, що обробляється.	Кількість кубітів визначає глибину розкладу сигналу.
Тип застосування	Аналіз спектра частот, швидке виявлення основних частотних компонентів.	Розкладання сигналу на різні рівні деталізації для подальшої обробки.
Використання в реальних задачах	Застосовується в обробці звукових сигналів, телекомунікаціях, розпізнаванні образів у медицині.	Використовується в стисненні зображень, прогнозуванні погоди, фінансових ринках.
Підхід до обробки даних	Задача перетворення сигналу на спектр, що дозволяє визначати частоти.	Задача фільтрації сигналу на різних рівнях деталізації для виділення важливих компонентів.
Ефективність у великому масштабі	Використовується для великих наборів даних, де потрібно швидко отримати частотні компоненти.	Ідеально підходить для багатокрокових аналізів з різними масштабами (час/частота).

ВИСНОВКИ

В ході дипломної роботи було проаналізовано, після чого реалізовано квантове вейвлет-перетворення Добеші та використано готовий інструмент бібліотеки pennylane квантового перетворення Фур'є з подальшим порівняльним аналізом цих двох методів обробки сигналу.

Для початку було розглянуто перетворення Фур'є, зокрема дискретне. Після чого було розглянуто швидке перетворення Фур'є, що є алгоритмом, який визначає дискретне перетворення Фур'є об'єкта швидше, ніж обчислює його. Для більш детального розуміння, було розглянуто алгоритм Кулі-Тьюкі, який є найвідомішим алгоритмом ШПФ.

Після було розпочато дослідження вейвлет-перетворення. Вейвлет – хвилеподібне коливання з амплітудою, яка починається з нуля, збільшується або зменшується, а потім повертається до нуля один або кілька разів. Слово вейвлет еквівалент французького слова ondelette, що означає «мала хвиля». Зупинились на вейвлеті Добеші для подальшого розглядання та реалізації. Вейвлети Добеші є сімейством ортогональних вейвлетів, які визначають дискретне вейвлет-перетворення та мають максимальну кількість моментів зникнення для конкретно вибраної опори.

В третьому розділі виконано реалізацію. Для квантового перетворення Фур'є було використано готовий інструмент бібліотеки penylane, а вейвлет Добеші реалізовано самостійно через побудову специфічних квантових операцій (RY для фазових зсувів та CNOT для заплутування). Після отримано результат і був проведений порівняльний аналіз даних методів обробки сигналу. Результати викладені в таблиці 3.1.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Michael T. Heideman, Don H. Johnson, and C. Sidney Burrus Gauss and the History of the Fast Fourier Transform. Archive for History of Exact Sciences. 1985. vol. 34, no. 3, p. 265-277.

2. Frank Yates The design and analysis of factorial experiments. Technical Communication No. 35 of the Commonwealth Bureau of Soils. 1937. 142 (3585): 90–92.

3. G.C. Danielson, C. Lanczos Some improvements in practical Fourier analysis and their application to x-ray scattering from liquids. Journal of the Franklin Institute. 1942. 233 (4): 365–380.

4. James W. Cooley The Re-Discovery of the Fast Fourier Transform Algorithm. Thomas J. Watson Research Center. 1987. p. 33-45.

5. Fourier Series URL: https://www.thefouriertransform.com/series/fourier.php (дата звернення 03.10.2024)

6. The Fourier Transform URL: https://www.thefouriertransform.com/transform/fourier.php (дата звернення 03.10.2024)

7. Discrete Fourier Transform - Frequencies - StatLect. URL: https://www.statlect.com/matrix-algebra/discrete-Fourier-transform-frequencies (дата звернення 03.10.2024)

8. Discrete Fourier Transform - Frequencies StatLect. URL: https://www.statlect.com/matrix-algebra/discrete-Fourier-transform-frequencies (дата звернення 03.10.2024)

9. ftshift - MathWorks. URL: https://www.mathworks.com/help/matlab/ref/fftshift.html (дата звернення 03.10.2024)

10. Fast Fourier Transform Explained – builtin. URL: https://builtin.com/articles/fast-fourier-transform (дата звернення 03.10.2024)

11. Jack Dongarra, Francis Sullivan Guest Editors Introduction to the top 10 algorithms. Computing in Science & Engineering. 2000. 2 (1). p. 22–23.

12. W. Morven Gentleman, G. Sande Fast Fourier Transforms: for fun and profit. AFIPS Fall Joint Computing Conference. 1966. p. 563-578.

13. Schatzman, James C. Accuracy of the discrete Fourier transform and the fast Fourier transform. SIAM Journal on Scientific Computing. 1996. p. 1150–1166.

14. Welch, Peter D. A fixed-point fast Fourier transform error analysis. IEEE Transactions on Audio and Electroacoustics. 1969. p. 151–157.

15. Ergün, Funda Testing multivariate linear functions. Proceedings of the twenty-seventh annual ACM symposium on Theory of computing. 1995. p. 407–416.

16. What Makes a Fourier Transform Fast? URL: https://www.algorithm-archive.org/contents/cooley_tukey/cooley_tukey.html (дата звернення 05.10.2024)

17. Duhamel P., M. Vetterli Fast Fourier transforms: a tutorial review and a state of the art. Signal Processing. 1990. Vol. 19. p. 259–299.

18. Steven G. Johnson, Matteo Frigo Implementing FFTs in Practice. Computer Science, Mathematics. 2009. p. 287.

19. I. Daubechies Where Do Wavelets Come From? Proceedings of the IEEE. 1996. Vol. 84, No. 4. p. 510-513.

20. Y. Meyer Wavelets, Applications and Algorithms. SIAM. 1993.

21. Wavelet URL: https://en.wikipedia.org/wiki/Wavelet (дата звернення 10.10.2024)

22. What Are Wavelet Transforms? - MatWorks. URL: https://www.mathworks.com/discovery/wavelet-transforms.html (дата звернення 11.10.2024)

23. Yves Meyer Wavelets and Operators. Cambridge: Cambridge University Press, 2009. 223 p.

24. Charles Sidney Burrus Introduction to Wavelets and Wavelet Transform : A Primer La Recherche. Houston, 1998. 291 p.

25. The Wavelet Tutorial URL: https://web.archive.org/web/20040210231301/http://users.rowan.edu/~polikar/WAVELETS/WTtutorial.html (дата звернення 11.10.2024)

26. A. N. Skodras Discrete Wavelet Transform: An Introduction. Technical Report. 2003.

27. David Barina Real-time wavelet transform for infinite image strips. Journal of Real-Time Image Processing. 2020. 18(3). p. 585-591.

28. James S. Walker A Primer on Wavelets and Their Scientific Applications. Eau Claire: Chapman & Hall, 2008. 318 p.

29. Elfouly, F. H., Mahmoud, M. I., Dessouky, M. I. M., & Deyab, S. Comparison between Haar and Daubechies Wavelet Transformions on FPGA Technology. International Journal of Computing. 2008. 6(3). p. 23-29.

30. Haar wavelet URL: https://en.wikipedia.org/wiki/Haar_wavelet (дата звернення 15.10.2024)

31. I. Daubechies Ten Lectures on Wavelets. SIAM. 1992. p. 194.

32. Software library for programming quantum computers URL: https://xanadu.ai/products/pennylane (дата звернення 30.10.2024)

33. Налаштування середовища виконання в Google Colab: поради та лайфхаки URL: https://foxminded.ua/google-colab/ (дата звернення 30.10.2024)

34. The QML Reference URL: https://doc.qt.io/qt-6/qmlreference.html (дата звернення 30.10.2024)

35. qml.QFT URL: https://docs.pennylane.ai/en/stable/code/api/pennylane.QFT.html (дата звернення 05.11.2024)

36. Murtaza Saadique Basha, M.Ramakrishnan Color Image Contrast Enhancement using Daubechies D4 Wavelet and Luminance Analysis. International Journal of Computer Applications. 2014. Volume 86 – No 6.

37. Чемерис К. М., Дейнега Л. Ю. Застосування методу вейвлет-аналізу для виявлення атак в мережах. Наука і техніка Повітряних Сил Збройних Сил України. 2022. № 1(46). С. 99-107.

38. Астраханцев А. А. Аналіз ефективності застосування вейвлет-перетворення в стеганографічних системах передавання даних / А. А. Астраханцев, О. О. Вовк // Вісник Національного університету «Львівська політехніка». Серія: Інформаційні системи та мережі: збірник наукових праць. – 2015. – № 832. – С. 9–17.

39. В. В. Лукічов Методи та засоби стеганографічного захисту інформації на основі вейвлет-перетворень: монографія / В. В. Лукічов, В. А. Лужецький, А. С. Васюра. – Вінниця : ВНТУ, 2014. – 160 с.

40. World’s smallest quantum computer unveiled, solves problems with just 1 photon URL: https://interestingengineering.com/innovation/worlds-smallest-quantum-computer-unveiled-solves-problems-with-just-1-photon (дата звернення 18.10.2024)

ДОДАТОК А

КОД РЕАЛІЗАЦІЇ КВАНТОВОГО ПЕРЕТВОРЕННЯ ФУР'Є ТА КВАНТОВОГО ПЕРЕТВОРЕННЯ ВЕЙВЛЕТА ДОБЕШІ D4

```
!pip install pennylane --upgrade
!pip install numpy

import numpy as np
import pennylane as qml

# Параметри для вейвлет-перетворення D4
h0 = (1 + np.sqrt(3)) / (4 * np.sqrt(2))
h1 = (3 + np.sqrt(3)) / (4 * np.sqrt(2))
h2 = (3 - np.sqrt(3)) / (4 * np.sqrt(2))
h3 = (1 - np.sqrt(3)) / (4 * np.sqrt(2))

g0 = (1 - np.sqrt(3)) / (4 * np.sqrt(2))
g1 = (-3 - np.sqrt(3)) / (4 * np.sqrt(2))
g2 = (3 - np.sqrt(3)) / (4 * np.sqrt(2))
g3 = (-1 - np.sqrt(3)) / (4 * np.sqrt(2))

approximation_filter = np.array([h0, h1, h2, h3])
detail_filter = np.array([g0, g1, g2, g3])

# Кількість кубітів
wavelet_qubits = 4
qft_qubits = 3
signal_length = 16  # довжина сигналу (2^n)

dev_qft = qml.device('default.qubit', wires=qft_qubits)
dev_d4 = qml.device("default.qubit", wires=wavelet_qubits)

# Реалізація QFT
@qml.qnode(dev_qft)
def circuit_qft(basis_state):
    qml.BasisState(basis_state, wires=range(qft_qubits))
    qml.QFT(wires=range(qft_qubits))
    return qml.state()

# Перетворення D4
@qml.qnode(dev_d4)
```

```
    def d4_wavelet_transform(input_state):
        qml.MottonenStatePreparation(input_state,
wires=range(wavelet_qubits))

        # Перший етап: апроксимація
        for i in range(0, wavelet_qubits - 1, 2):  # Обробка парних
індексів
            qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
            qml.CNOT(wires=[i, i + 1])
            qml.RY(2 * np.arctan(approximation_filter[1]), wires=i +
1)

        for i in range(1, wavelet_qubits - 1, 2):  # Обробка непарних
індексів
            qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
            qml.CNOT(wires=[i, i + 1])
            qml.RY(2 * np.arctan(approximation_filter[1]), wires=i +
1)

        # Другий етап: деталізація
        for i in range(0, wavelet_qubits - 2, 2):  # Обробка парних
індексів
            qml.CNOT(wires=[i + 1, i + 2])
            qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
            qml.CNOT(wires=[i, i + 1])

        for i in range(1, wavelet_qubits - 2, 2):  # Обробка непарних
індексів
            qml.CNOT(wires=[i + 1, i + 2])
            qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
            qml.CNOT(wires=[i, i + 1])

        return qml.state()

    # Генерація сигналу
    t = np.linspace(0, 2 * np.pi, signal_length)
    input_signal = np.sin(t)

    # Нормалізація сигналу
    input_signal_normalized = input_signal /
np.linalg.norm(input_signal)

    # Додавання нулів до сигналу, щоб довжина була ступенем двійки
    padded_signal_length = 16  # приклад для 16 елементів (2^4)
```

```
    input_signal_padded = np.concatenate((input_signal_normalized,
np.zeros(padded_signal_length - len(input_signal_normalized))))

    # Виконання QFT
    basis_state = [0, 0, 0]  # Основний стан для QFT
    frequency_index =
np.argmax(np.abs(np.fft.fft(input_signal_normalized)))
    if frequency_index < 2**qft_qubits:
        basis_state[frequency_index] = 1

    qft_result = circuit_qft(basis_state)

    # Виконання перетворення D4
    quantum_state_d4 = d4_wavelet_transform(input_signal_padded)

    # Аналіз квантового стану для отримання коефіцієнтів
    def extract_coefficients(quantum_state):
        amplitudes = np.abs(quantum_state)      # Підрахунок амплітуд
для отримання коефіцієнтів

        # Визначення коефіцієнтів апроксимації та деталізації
        mid = len(amplitudes) // 2
        approximation_coeffs = amplitudes[:mid]
        detail_coeffs = amplitudes[mid:]

        return approximation_coeffs, detail_coeffs

    # Отримання коефіцієнтів
    approximation_coeffs,               detail_coeffs               =
extract_coefficients(quantum_state_d4)

    # Виведення результатів
    print("QFT Result:", qft_result)
    print("Quantum State (D4 Wavelet):", quantum_state_d4)
    print("Approximation Coefficients:", approximation_coeffs)
    print("Detail Coefficients:", detail_coeffs)
```

ДОДАТОК Б

ПРЕЗЕНТАЦІЯ

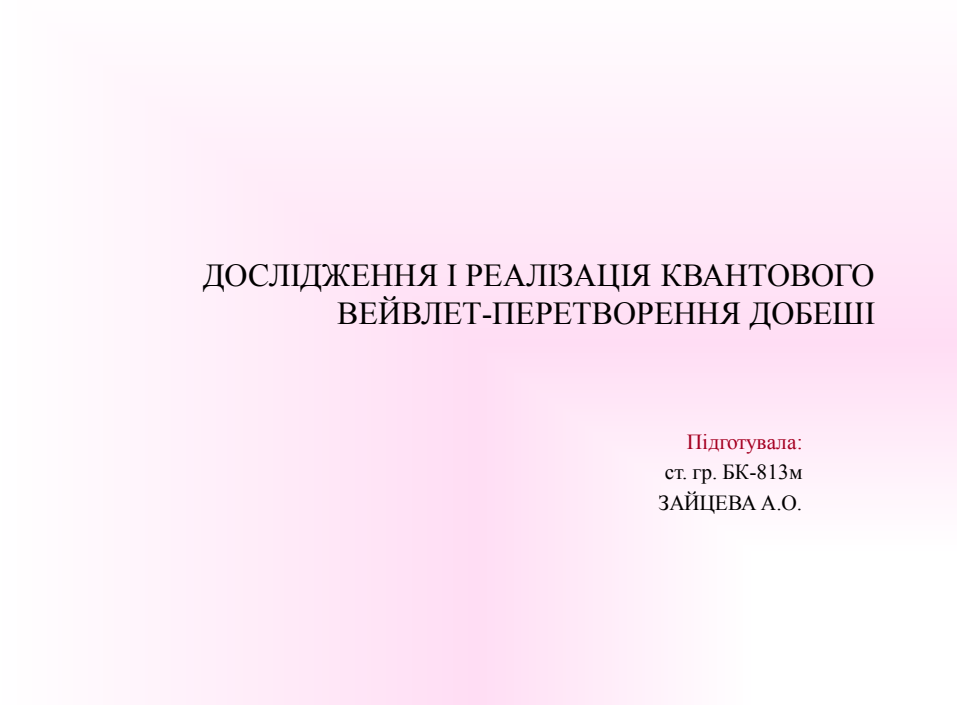


Рисунок Б.1 – Титульний слайд

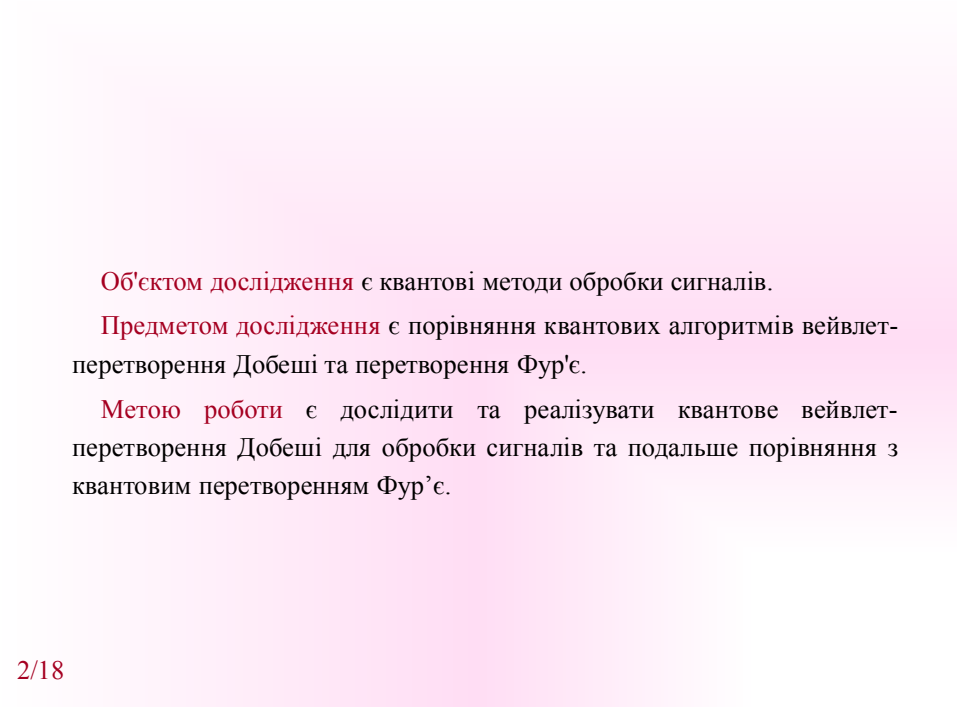


Рисунок Б.2 – Слайд 2

Завданнями роботи є:

1) розглянути перетворення Фур’є;
2) розглянути вейвлет Добеші;
3) реалізувати квантове перетворення Добеші та застосувати готовий інструмент квантового перетворення Фур’є;
4) зробити порівняльний аналіз перетворень.

3/18

Рисунок Б.3 – Слайд 3

ВСТУП

Квантові обчислення відкривають нові можливості для обробки інформації, зокрема через квантове перетворення Фур'є та вейвлет-перетворення. Обидва методи дозволяють ефективно аналізувати сигнали з високою точністю та швидкістю. Квантове перетворення Фур'є зменшує складність обчислень у порівнянні з класичними алгоритмами, а квантове вейвлет-перетворення дає змогу локалізувати інформацію як у частотній, так і в часовій області. У роботі розглянемо реалізацію та порівняємо дані методи.

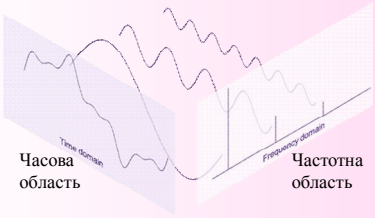
4/18

Рисунок Б.4 – Вступ

Перетворення Фур’є

Дискретне перетворення Фур'є дає змогу виявити частотний склад сигналу та перетворити його з часової області в частотну.

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-j\frac{2\pi nk}{N}}$$



Часова область

Частотна область

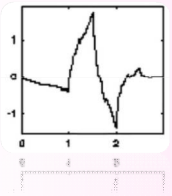
1. Послідовності дискретних відліків сигналу x[n].
2. Обчислення комплексних амплітуд.
3. Результат представляється послідовністю комплексних чисел X[k].

Алгоритм, який визначає дискретне перетворення Фур’є об’єкта швидше, ніж обчислює його – швидке перетворення Фур’є.

5/18

Рисунок Б.5 – Перетворення Фур’є

Вейвлет Добеші



Вейвлет – математична функція, яку використовують для поділу заданої функції або сигналу безперервного часу на компоненти масштабу.

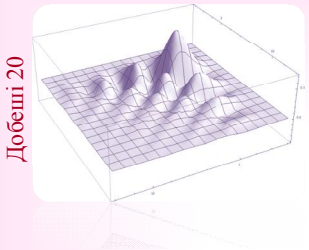
Вейвлети Добеші є сімейством ортогональних вейвлетів, які визначають дискретне вейвлет-перетворення та мають максимальну кількість моментів зникнення для конкретно вибраної опори.

Дискретне вейвлет-перетворення сигналу x обчислюється, пропускаючи його через низку фільтрів.

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}.$$

$$g_k = (-1)^k h_{N-1-k},$$

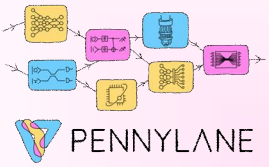
$$g_0 = h_3, g_1 = -h_2, g_2 = h_1, g_3 = -h_0.$$



6/18

Рисунок Б.6 – Вейвлет Добеші

PennyLane & Google Colab



бібліотека програмного забезпечення з відкритим кодом.

Розроблена для квантового машинного навчання та квантових обчислень.

Дозволяє користувачам створювати, тренувати та запускати квантові схеми на різних квантових пристроях і симуляторах.

хмарне середовище для роботи з Jupyter Notebook.

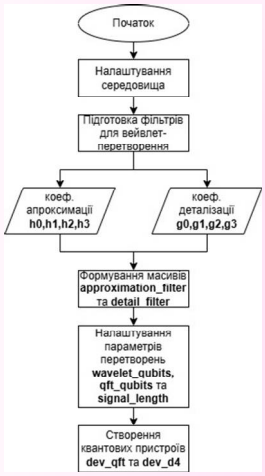
Присутній безкоштовний доступ до графічних процесорів, що дозволяє прискорити виконання складних обчислень і моделей машинного навчання.

Підтримує популярні бібліотеки Python, має зручний інтерфейс і забезпечує можливість спільної роботи над проєктами.

7/18

Рисунок Б.7 – Pennylane & Google Colab

Кодова реалізація. Етап підготовки



Встановлюємо та підключаємо бібліотеки **pennylane** та **numpy**.

Підготуємо фільтри для вейвлет-перетворення. Спочатку пропишемо значення коефіцієнтів апроксимації та деталізації, після чого сформуємо два масиви відповідно: **approximation_filter** = np.array([h0, h1, h2, h3]) та **detail_filter** = np.array([g0, g1, g2, g3]).

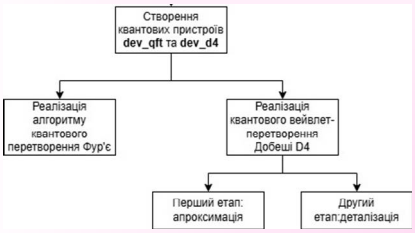
Налаштуємо початкові параметри для подальшої роботи: встановимо значення **кількості кубітів** для квантового перетворення Фур'є та вейвлет-перетворення D4, а також **довжину сигналу**.

Наостанок створимо квантові пристрої **dev_qft** та **dev_d4**.

8/18

Рисунок Б.8 – Кодова реалізація. Етап підготовки

Етап реалізації перетворень



Декоратор **@qml.qnode(dev_qft)** позначає функцію як квантову схему. Функція **def circuit_qft** описує схему, де вхідний параметр **basis_state** задає початковий стан кубітів. Операція **qml.BasisState** ініціалізує кубіти в цей стан. **qml.QFT** виконує квантове перетворення Фур'є. Після чого **return qml.state()** повертає квантовий стан кубітів.

Декоратор **@qml.qnode(dev_d4)** визначає функцію як квантову схему. Функція **def d4_wavelet_transform** описує вейвлет-перетворення D4, яке обробляє вхідний класичний сигнал **input_state**. Операція **qml.MottonenStatePreparation** готує цей сигнал для квантової системи, перетворюючи його на квантовий стан, який зберігається в **wavelet_qubits**.

```
@qml.qnode(dev_d4)
def d4_wavelet_transform(input_state):
    qml.MottonenStatePreparation
  (input_state, wires=range(wavelet_qubits))
```

9/18

Рисунок Б.9 – Етап реалізації перетворень

```
# Перший етап: апроксимація
for i in range(0, wavelet_qubits - 1, 2):
    qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
    qml.CNOT(wires=[i, i + 1])
    qml.RY(2 * np.arctan(approximation_filter[1]), wires=i + 1)

for i in range(1, wavelet_qubits - 1, 2):
    qml.RY(2 * np.arctan(approximation_filter[0]), wires=i)
    qml.CNOT(wires=[i, i + 1])
    qml.RY(2 * np.arctan(approximation_filter[1]), wires=i + 1)
```

Кожен парний/непарний кубіт обробляється через **обертання**, яке змінює його стан відповідно до фільтра для апроксимації. Потім між кожною парою кубітів (парні/непарні індекси та їхні наступні сусіди) виконується операція **CNOT**, щоб "передати" або коректувати квантову інформацію. Після цього, другий кубіт в парі (з індексом i + 1) також **обертається** згідно з другим коефіцієнтом фільтра для апроксимації.

Для кожної пари кубітів виконується операція **CNOT** між кубітами $i + 1$ та $i + 2$, що дозволяє передавати чи коригувати квантову інформацію. Потім кубіт $i + 2$ обертається за допомогою операції **RY**, де кут обертання залежить від коефіцієнта фільтра деталізації. Після цього між кубітами i та $i + 1$ знову застосовується операція **CNOT** для корекції чи передачі інформації.

```
# Другий етап: деталізація
for i in range(0, wavelet_qubits - 2, 2):
    qml.CNOT(wires=[i + 1, i + 2])
    qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
    qml.CNOT(wires=[i, i + 1])

for i in range(1, wavelet_qubits - 2, 2):
    qml.CNOT(wires=[i + 1, i + 2])
    qml.RY(2 * np.arctan(detail_filter[0]), wires=i + 2)
    qml.CNOT(wires=[i, i + 1])

return qml.state()
```

10/18

Рисунок Б.10 – Слайд 10

Етап генерації сигналу

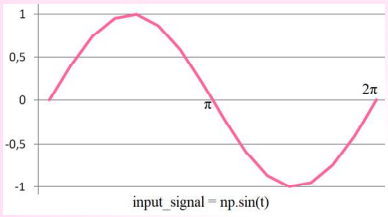
Створюємо вхідний сигнал. Масив *t*, що містить рівномірно розподілені значення від 0 до 2π, причому загальна кількість значень визначається параметром **signal_length**.

Перед тим, як пропускати сигнал через перетворення, його потрібно нормалізувати. Це процес, коли масштабуємо вектор так, щоб його норма, тобто довжина, стала рівною 1.

Оскільки вихідний сигнал може мати довжину менше ніж 16 елементів, потрібно додати достатньо нулів, щоб досягти бажаної довжини.

```
(padded_signal_length - len(input_signal_normalized))))
```

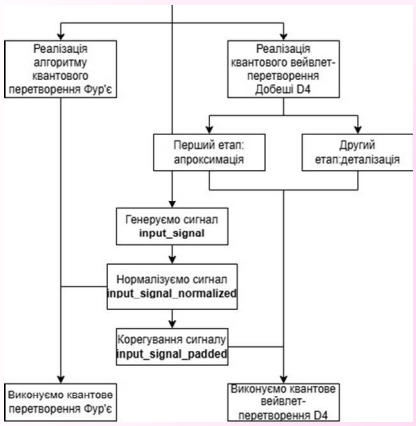
різниця між бажаною довжиною та фактичною довжиною сигналу. Це дає кількість нулів, які потрібно додати. Функцію **np.zeros** створює масив з нулів, які треба додати. Функція **np.concatenate** об'єднує два масиви: нормалізований сигнал та масив нулів.



11/18

Рисунок Б.11 – Етап генерації сигналу

Етап виконання квантових перетворень



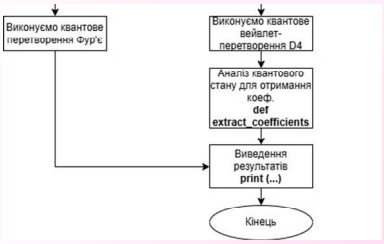
Виконання вейвлет-перетворення. Подаємо сигнал у квантовий вузол **d4_wavelet_transform**. Після чого отримуємо квантовий стан.

Виконання перетворення Фур'є. Створюємо масив - основний квантовий стан. Функція **np.fft.fft** перетворює сигнал з часової області в частотну. За допомогою **np.abs()** отримуємо амплітуди, а **np.argmax()** знаходить індекс найбільш вираженої частоти. Потім виконується квантове перетворення Фур'є через **circuit_qft**.

12/18

Рисунок Б.12 – Етап виконання квантових перетворень

Завершальний етап



Створюємо функцію **def extract_coefficients** для аналізу квантового стану, щоб отримати коефіцієнти, які описують стан кубітів.

Далі розподіляємо амплітуди квантового стану на два набори коефіцієнтів: **апроксимаційні** та **деталізаційні**.

```
mid = len(amplitudes) // 2
    approximation_coeffs = amplitudes[:mid]
    detail_coeffs = amplitudes[mid:]

    return approximation_coeffs, detail_coeffs
```

В наступному рядку викликаємо функцію, щоб отримати коефіцієнти з квантового стану.

```
approximation_coeffs, detail_coeffs=
extract_coefficients(quantum_state_d4)
```

І виводимо результати на екран.

13/18

Рисунок Б.13 – Завершальний етап

Результати

```
QFT Result: [ 0.35355339+0.j          0.        +0.35355339j -0.35355339+0.j
  0.        -0.35355339j  0.35355339+0.j          0.        +0.35355339j
 -0.35355339+0.j          0.        -0.35355339j]
Quantum State (D4 Wavelet): [-0.18151887+8.17480008e-18j -0.2575212 +1.19275975e-17j
 -0.3929905 +1.11635249e-17j  0.15036484-5.67434333e-18j
  0.44763187-1.67789778e-17j  0.08147727-2.55938875e-18j
  0.10865971-4.28798240e-18j  0.62190698-2.73113115e-17j
 -0.01573924+5.07352991e-18j -0.25104173+6.18044184e-17j
 -0.01336068+1.04213696e-17j  0.02632096-3.95632418e-18j
  0.15882805-2.37694630e-17j  0.01458512-5.65417615e-18j
 -0.08876239+2.07523966e-17j -0.14365144+3.29104707e-17j]
Approximation Coefficients: [0.18151887 0.2575212  0.3929905  0.15036484 0.44763187 0.08147727
 0.10865971 0.62190698]
Detail Coefficients: [0.01573924 0.25104173 0.01336068 0.02632096 0.15882805 0.01458512
 0.08876239 0.14365144]
```

14/18

Рисунок Б.14 – Результати

Порівняльний аналіз перетворень

Характеристика	Квантове перетворення Фур'є	Квантове вейвлет-перетворення Добеші
Призначення	Перетворення сигналу з часової області в частотну.	Розкладання сигналу на апроксимаційні та деталізуючі компоненти.
Тип аналізу	Частотний аналіз (перехід у частотну область).	Багатошарове розкладання (аналог часу-частоти).
Основна операція	Квантове перетворення Фур'є.	Вейвлетне перетворення Добеші.
Тип сигналу	Підходить для перетворення сигналів з чітко вираженими частотами.	Підходить для обробки сигналів з різною частотною складністю та масштабами.
Алгоритмічні операції	Використовує Hadamard-гейти та фазові гейти для створення суперпозицій.	Використовує RY-гейти для фазових зсувів та CNOT-гейти для заплутування.
Простота реалізації	Вбудована функція в PennyLane (qml.QFT), автоматичне застосування гейтів.	Реалізація через побудову специфічних квантових операцій (RY та CNOT).

15/18

Рисунок Б.15 – Порівняльний аналіз перетворень

Характеристика	Квантове перетворення Фур'є	Квантове вейвлет-перетворення Добеші
Основні ресурси	Кількість кубітів визначає розмір сигналу, що обробляється.	Кількість кубітів визначає глибину розкладу сигналу.
Тип застосування	Аналіз спектра частот, швидке виявлення основних частотних компонентів.	Розкладання сигналу на різні рівні деталізації для подальшої обробки.
Використання в реальних задачах	Застосовується в обробці звукових сигналів, телекомунікаціях, розпізнаванні образів у медицині.	Використовується в стисненні зображень, прогнозуванні погоди, фінансових ринках.
Використання квантових гейтів	Hadamard-гейти, фазові гейти, CNOT.	RY-гейти для фазових зсувів, CNOT для заплутування.
Підхід до обробки даних	Задача перетворення сигналу на спектр, що дозволяє визначати частоти.	Задача фільтрації сигналу на різних рівнях деталізації для виділення важливих компонентів.
Ефективність у великому масштабі	Використовується для великих наборів даних, де потрібно швидко отримати частотні компоненти.	Ідеально підходить для багатокрокових аналізів з різними масштабами (час/частота).

16/18

Рисунок Б.16 – Слайд 16

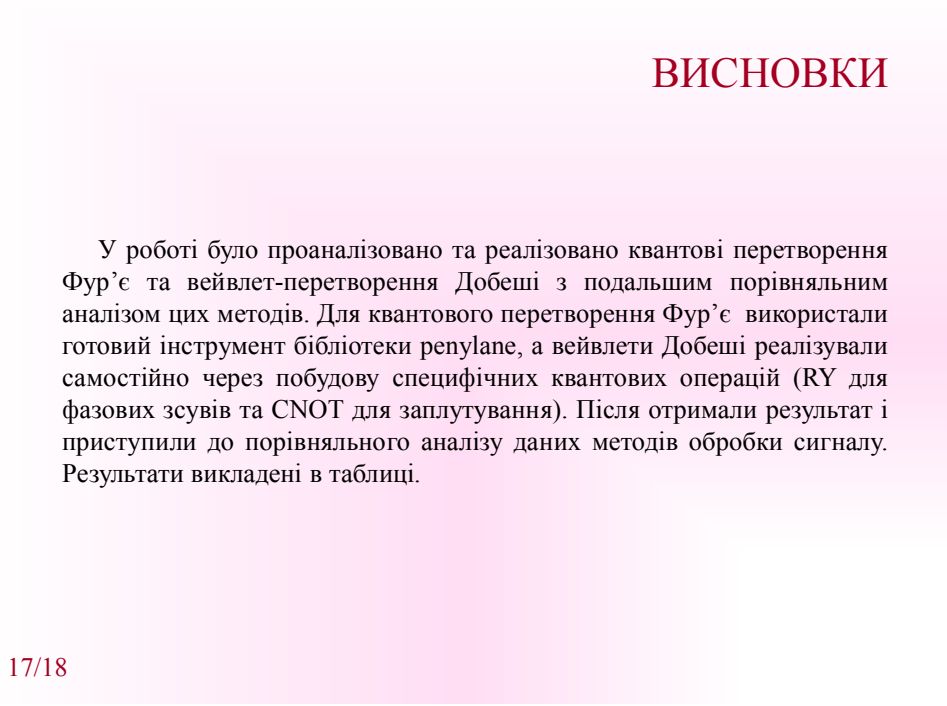


Рисунок Б.17 – Висновки

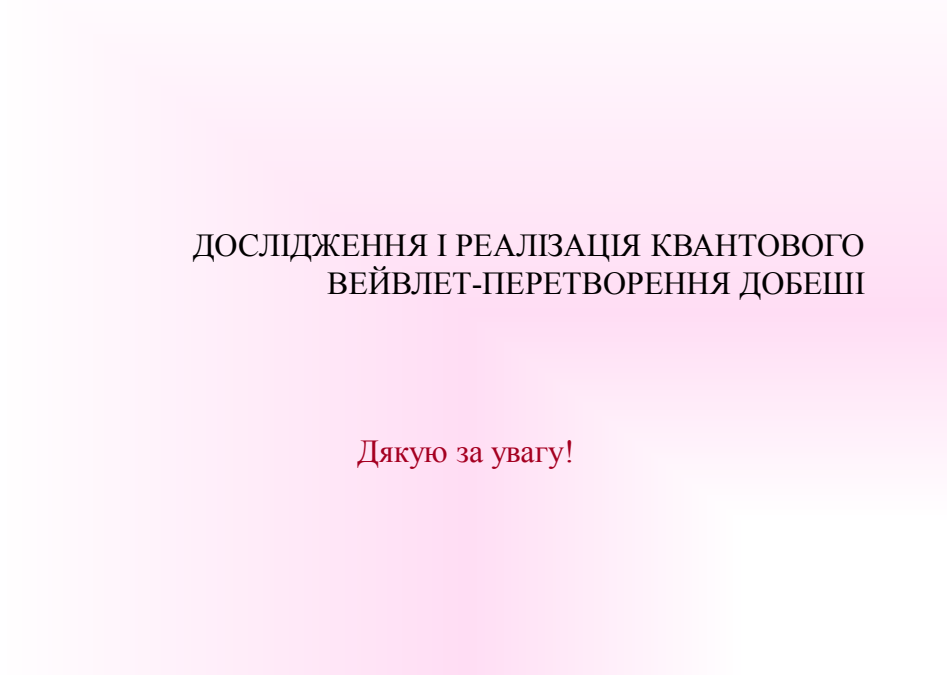


Рисунок Б.18 – Фінальний слайд