Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №2.1

з дисципліни
«Інтелектуальні вбудовані системи»

на тему «Дослідження параметрів алгоритму дискретного перетворення Фур’є»

Виконав:
студент 3 курсу
ФІОТ, групи ІП-84
Кришталь Дмитро Вікторович
Номер залікової - 8414

Перевірив:
Регіда П. Г.

Київ 2021

Основні теоретичні відомості

В основі спектрального аналізу використовується реалізація так званого дискретного перетворювача Фур'є (ДПФ) з неформальним (не формульним) поданням сигналів, тобто досліджувані сигнали представляються послідовністю відліків $x(k)$

$$F_x(p) = \sum_{k=0}^{N-1} x(k) \cdot e^{-jk\Delta t p \Delta\omega}$$

$$\omega \to \omega_p \to p\Delta\omega \to p \qquad \Delta\omega = \frac{2\pi}{T}$$

На всьому інтервалі подання сигналів **T**, **2π** - один період низьких частот. Щоб підвищити точність треба збільшити інтервал **T.**

$$t \to t_k \to k\Delta t \to k; \qquad \Delta t = \frac{T}{N} = \frac{1}{k_{зап.}} \cdot f'гр.$$

ДПФ - проста обчислювальна процедура типу звірки (тобто Σ-е парних множень), яка за складністю також має оцінку **N^2 + N.** Для реалізації ДПФ необхідно реалізувати поворотні коефіцієнти ДПФ:

$$W_N^{pk} = e^{-jk\Delta t\Delta\omega p}$$

Ці поворотні коефіцієнти записуються в ПЗУ, тобто є константами.

$$W_N^{pk} = e^{-jk\frac{T}{N}p\frac{2\pi}{T}} = e^{-j\frac{2\pi}{N}pk}$$

W_N^{pk} не залежать від **T**, а лише від розмірності перетворення **N.** Ці коефіцієнти подаються не в експоненційній формі, а в тригонометричній.

$$W_N^{pk} = \cos\left(\frac{2\pi}{N}pk\right) - j\sin\left(\frac{2\pi}{N}pk\right)$$

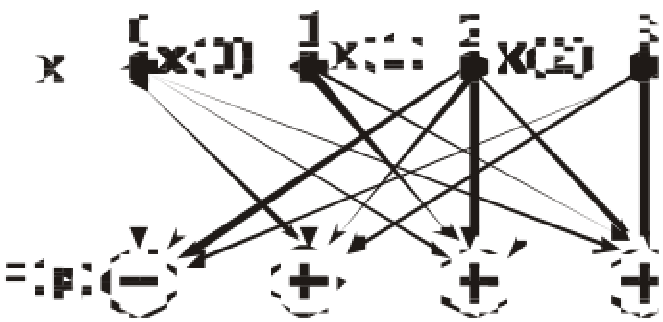
Ці коефіцієнти повторюються (тому і **p** до **N-1**, і **k** до **N-1**, а **(N-1) • (N-1)**) з періодом **N(2π)**.. Т.ч. в ПЗУ треба зберігати N коефіцієнтів дійсних і уявних частин. Якщо винести знак коефіцієнта можна зберігати **N/2** коефіцієнтів.

2π/N- деякий мінімальний кут, на який повертаються ці коефіцієнти. У ПЗУ окремо зберігаються дійсні та уявні частини компілюють коефіцієнтів. Більш загальна форма ДПФ представляється як:

$$F_x(p) = \sum_{k=0}^{N-1} x(k) \cdot W_N^{pk}$$

ДПФ дуже зручно представити у вигляді відповідного графа.

Приклад: граф 4-х точкового ДПФ. ($k = \overline{0,3}$; $p = \overline{0,3}$)



Коефіцієнти зручно представити у вигляді таблиці:

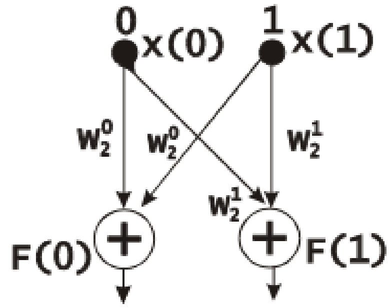
p / k	0	1	2	3
0	W_4^0	W_4^0	W_4^0	W_4^0
1	W_4^0	W_4^1	W_4^2	W_4^3
2	W_4^0	W_4^2	W_4^0	W_4^2
3	W_4^0	W_4^3	W_4^2	W_4^1

Різних тут всього 4 коефіцієнта:

$$W_4^0 = \cos\left(\frac{2\pi}{4} \cdot 0\right) - j\sin\left(\frac{2\pi}{4} \cdot 0\right) = 1 \qquad (W_4^1 = -j;\ W_4^2 = -1;\ W_4^3 = +j)$$

Можна в пам'яті зберігати тільки 2, а решта брати з "-", якщо $\frac{N}{2} - 1 < pk$. 4 ДПФ це вироджені перетворення, по модулю ці коефіцієнти = 1 і всі 4 ДПФ можуть реалізуватися на 24-х суматора. Це буде далі використовуватися в реалізації ШПФ з основою 4.

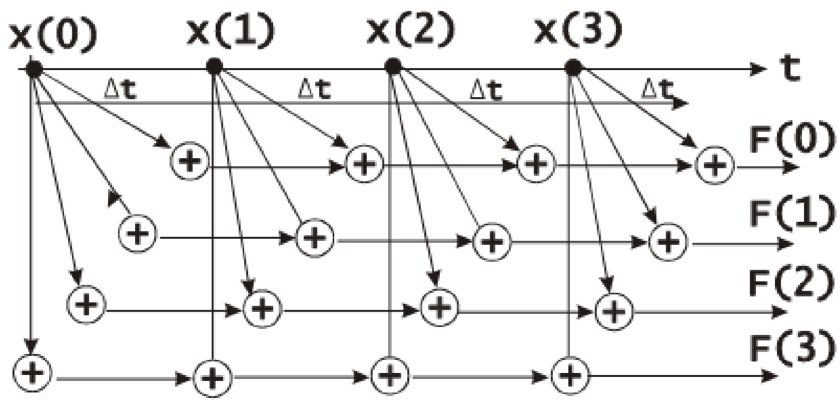
2ДПФ реалізується ще простіше:



($W_2^0 = +1$; $W_2^1 = -1$)

Спеціальна схема реалізації ДПФ з активним використанням пауз між відліками

При реалізації ДПФ можна організувати обробку в темпі надходження даних. Реалізація схеми в БПФ з активним використанням пауз на 4-х точках виглядає так:



Ця схема сильно залежить от **Δt** и **N**.

Умови завдання для варіанту бригади

Варіант: 14.

Число гармонік в сигналі: 6.

Гранична частота, $\omega_{гр}$: 2100.

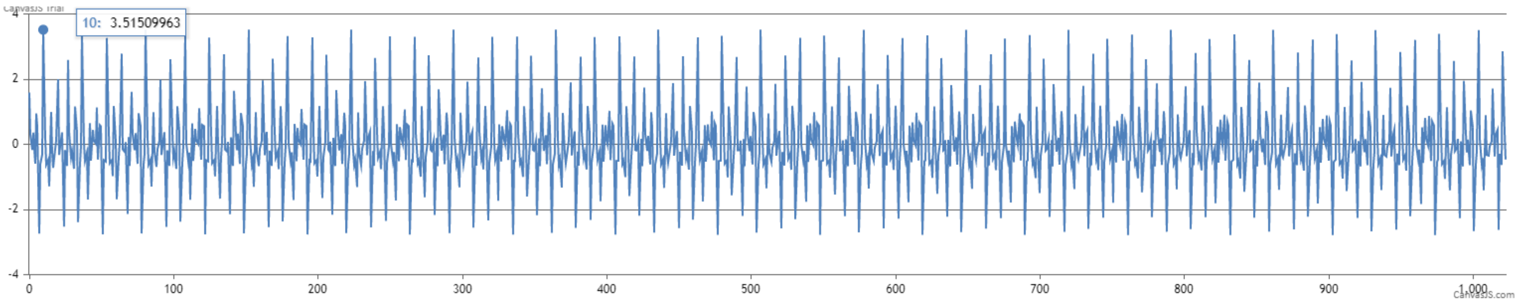
Кількість дискретних відліків: 1024

Лістинг програми із заданими умовами завдання

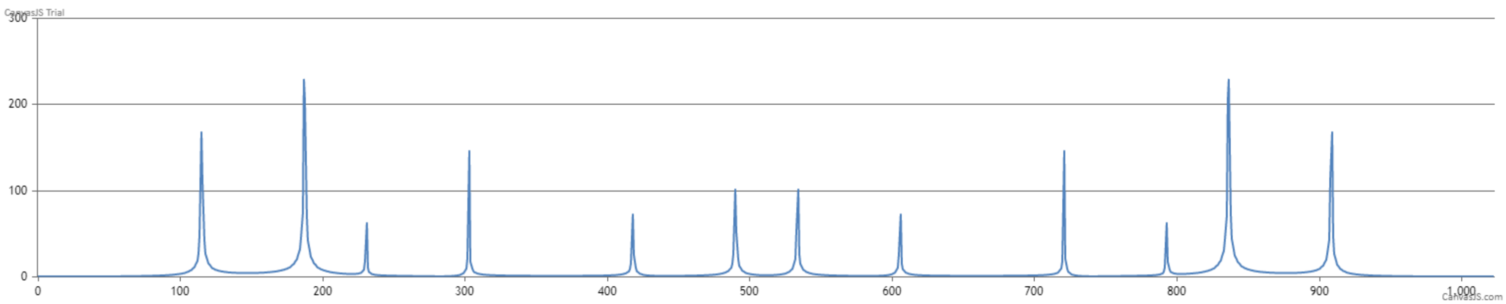
```
const n = 6
const w = 2100
const N = 1024

const getSignal = () => {
    const x = []
    while(x.length != N) {
        x.push({y: 0})
    }

    for(let i = 0; i < n; i++) {
        const omega = w/n * (i + 1);
        const A = Math.random();
        const Fi = Math.random();

        for(let t = 0; t < N; t++) {
            x[t].y += A * Math.sin(omega * t + Fi)
        }
    }

    return x
}

const complexNumber = () => {
    return {real:0, im: 0}
}

const discreteFourier = (signals) => {
    const result = []
    for (let p = 0; p < N; p++) {
        let num = complexNumber()
        for(let k = 0; k < N; k++) {
            num.real += signals[k].y * Math.cos(2 * Math.PI * p * k / N)
            num.im -= signals[k].y * Math.sin(2 * Math.PI * p * k / N)
        }
        result.push({y: Math.sqrt(Math.pow(num.im, 2) + Math.pow(num.real, 2))})
    }
    return result
}
```

Результати виконання програми

Сигнал:



Дискретне перетворення Фур’є:



Висновок

Під час виконання лабораторної роботи №2.1 ознайомився з принципами реалізації спектрального аналізу випадковиї сигналів на основі алгоритму дискретного перетворення Фур’є. Написав програму. Наведені скріншоти підтверджують правильність її роботи.