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 №2.2

з дисципліни
«Інтелектуальні вбудовані системи»

на тему «Дослідження алгоритму швидкого перетворення Фур’є з проріджуванням відліків сигналів у часі»

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

Перевірив:
Регіда П. Г.

Київ 2021

Основні теоретичні відомості

Швидкі алгоритми ПФ отримали назву схеми Кулі-Тьюкі. Всі ці алгоритми використовують регулярність самої процедури ДПФ і те, що будь-який складний коефіцієнт W_N^{pk} можна розкласти на прості комплексні коефіцієнти.

$$W_N^{pk} = W_N^1 W_N^2 W_N^3$$

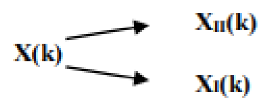
Для стану таких груп коефіцієнтів процедура ДПФ повинна стати багаторівневою, не порушуючи загальних функціональних зв'язків графа процедури ДПФ.

Існують формальні підходи для отримання регулярних графів ДПФ. Всі отримані алгоритми поділяються на 2 класи:

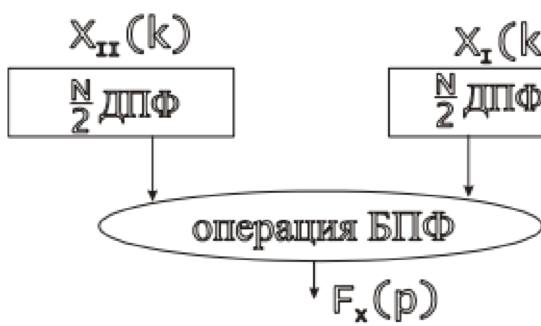
1) На основі реалізації принципу зріджені за часом $\mathbf{X_к}$

2)на основі реалізації принципу зріджені відліків шуканого спектру **F(p).**

Найпростіший принцип зріджені - поділу на парні/непарні пів-послідовності, які потім обробляють паралельно. А потім знаходять алгоритм, як отримати шуканий спектр.



Якщо нам вдасться ефективно розділити, а потім алгоритм отримання спектра, то ми можемо перейти від **N** ДПФ до **N/2** ДПФ.



Розглянемо формальний висновок алгоритму ШПФ, який реалізує в одноразовому застосуванні принцип проріджування по часу:

$$F_x(p) = \sum_{k=0}^{N-1} X(k) W_N^{pk} = \sum_{k=0}^{N-2} X_{II}(k) W_N^{pk} + \sum_{k=1}^{N-2} X_I(k) W_N^{pk}$$

$$X_{II}(k) \rightarrow X(2k^*);\; X_I(k) \rightarrow X(2k^*+1);\; k^* = 0; \frac{N}{2} - 1$$

$$F_x(p) = \sum_{k^*=0}^{\frac{N}{2}-1} X(2k^*) W_N^{pk} + \sum_{k^*=0}^{\frac{N}{2}-1} X(2k^*+1) W_N^{p(2k^*+1)}$$

$$W_N^{p2k^*} = e^{-j\frac{2\pi}{N}p2k^*} = e^{-j\frac{2\pi}{N/2}pk^*} = W_{\frac{N}{2}}^{pk^*}$$

У цій першій сумі з'явилися коефіцієнти в 2 рази менше.

У другій сумі з'явився множник, який не залежить від k * тобто він може бути винесений за знак суми.

$$W_N^{p(2k^*+1)} = W_N^{p2k^*} \cdot W_N^p = W_{\frac{N}{2}}^{pk^*} W_N^p$$

$$F_x(p) = \underbrace{\sum_{k^*=0}^{\frac{N}{2}-1} X(2k^*) W_{\frac{N}{2}}^{pk^*}}_{F_{II}(p^*)} + \underbrace{W_N^p \sum_{k^*=0}^{\frac{N}{2}-1} X(2k^*+1) W_{\frac{N}{2}}^{pk^*}}_{F_I(p^*)}$$

Ми бачимо, що всі вирази можна розділити на 2 частини, які обчислюються паралельно.

$F_I(p^*)$ - проміжний спектр, побудований на парних відліку. У цьому алгоритмі передбачається, щоб отримати спектр **F(p)** треба виконати 2 незалежних **N/2** ШПФ.

1) $$F_{II}(p^*) = \sum_{k^*=0}^{\frac{N}{2}-1} X(2k^*) W_{\frac{N}{2}}^{pk^*} \qquad p^* = 0, \frac{N}{2}-1$$

2) $$F_I(p^*) = \sum_{k^*=0}^{\frac{N}{2}-1} X(2k^*+1) W_{\frac{N}{2}}^{pk^*}$$

А на наступному кроці буде реалізована швидка збірка, тобто ШПФ з зрідженим за часом за формулою:

$$F_\tau(p^*) = F_{II}(p^*) + W_N^{p^*} F_I(p^*)$$

Але в цьому виразі різні **p** для зв'язку їх

Якщо p <N/2, то p = p * 1-а половина спектру

Якщо p **≥** N/2, то p = p * + N/2 2-а половина спектру

В алгоритмі БПФ вже використовуються 2 рівня

$$F_x(p^*) = F_{II}(p^*) + W_N^{p^*} F_I(p^*)$$

$$F_x\left(p^* + \frac{N}{2}\right) = F_{II}(p^*) + W_N^{p^*+\frac{N}{2}} F_I(p^*)$$

Алгоритм ШПФ з зріджeним по часу:

$$F_x(p^*) = F_{II}(p^*) + W_N^{p^*} F_I(p^*) \qquad p^* = 0, \frac{N}{2} - 1$$

$$F_x\left(p^* + \frac{N}{2}\right) = F_{II}(p^*) - W_N^{p^*} F_I(p^*) \qquad W_N^{p^*}$$

$\frac{N}{2}$ помножений на комплексний коефіцієнт.

Загальна схема самого ДПФ змінилася замість однорівневого перетворення.

Умови завдання для варіанту бригади

Варіант: 14.

Число гармонік в сигналі: 6.

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

Кількість дискретних відліків: 1024

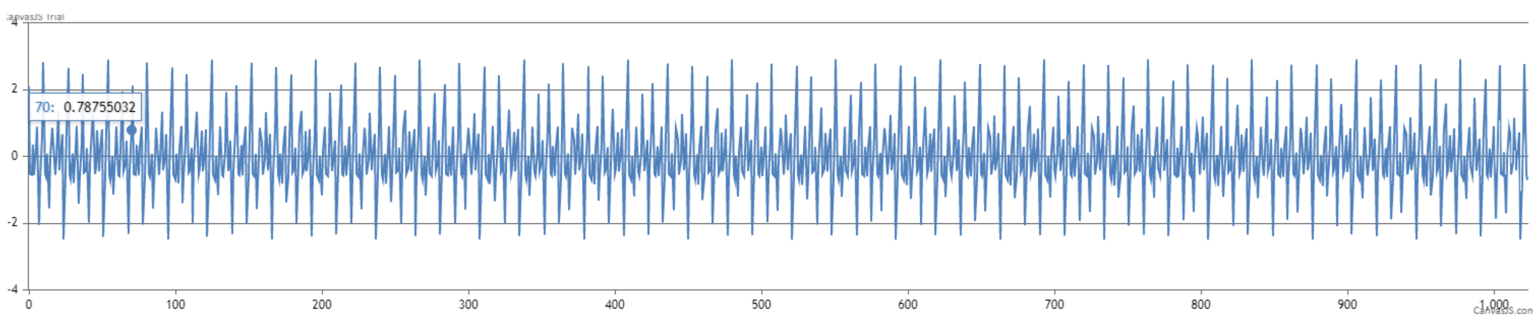
Лістинг програми із заданими умовами завдання

```
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
}
```

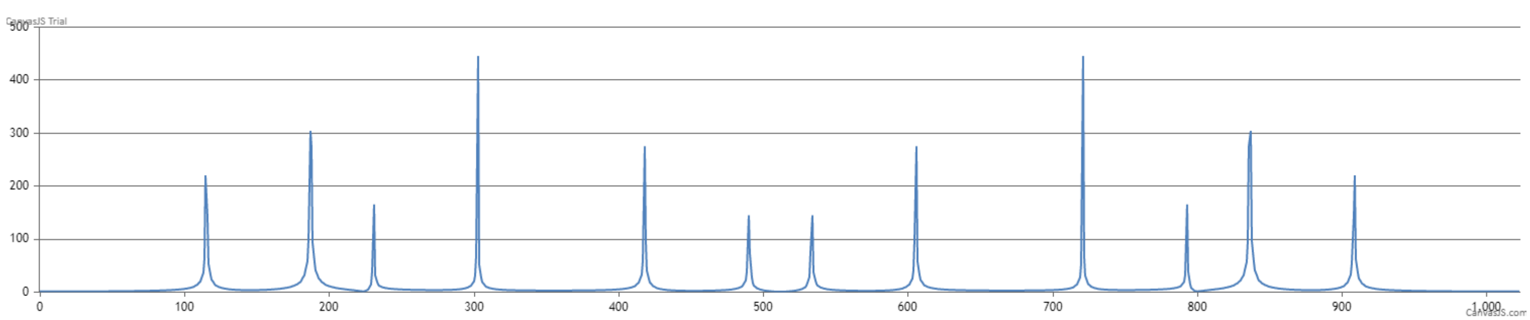
```
const complexNumber = () => {
    return {real:0, im: 0}
}

const fastFourier = (signal) => {
    const result = [];
    result.length = N;
    for(let p = 0; p < N/2; p++) {
        let even = complexNumber()
        let odd = complexNumber()
        for(let k = 0; k < N/2; k++) {
            even.real += signal[2*k].y * Math.cos(2 * Math.PI * p * 2*k / N);
            even.im -= signal[2*k].y * Math.sin(2 * Math.PI * p * 2*k / N);
            odd.real += signal[2*k + 1].y * Math.cos(2 * Math.PI * p * (2*k + 1) / N);
            odd.im -= signal[2*k + 1].y * Math.sin(2 * Math.PI * p * (2*k + 1) / N);
        }
        result[p] = {y: Math.sqrt((even.real + odd.real) ** 2 + (even.im + odd.im) **
2)};
        result[N/2 + p] = {y: Math.sqrt((even.real - odd.real) ** 2 + (even.im - odd.i
m) ** 2)};
    }
    return result
}
```

Результати виконання програми

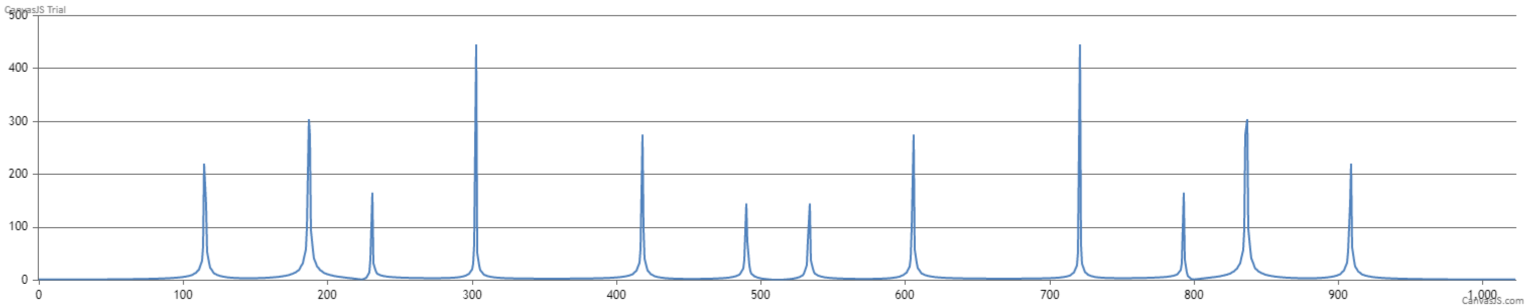
Сигнал:



Дискретне перетворення Фур’є:



Швидке перетворення Фур’є



Висновок

Під час виконання лабораторної роботи №2.2 ознайомився з принципами реалізації прискореного спектрального аналізу випадкових сигналів на основі алгоритму швидкого перетворення Фур’є. Написав програму за варіантом. Наведені скріншоти підтверджують правильність її роботи.