НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №2.1
“Дослідження параметрів алгоритму дискретного перетворення Фур'є”

Виконав:
студент групи ІП-84
ІП-8408
Засько Євгеній

Київ 2021

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).

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

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

На всьому інтервалі подання сигналів T, 2 - 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 -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^{pk}_N 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(2).. Т.ч. в ПЗУ треба зберігати N коефіцієнтів дійсних і уявних частин. Якщо винести знак коефіцієнта можна зберігати N/2 коефіцієнтів. 2/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 = 0,3; p = 0,3).

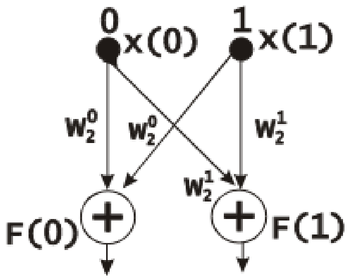
Коефіцієнти зручно представити у вигляді таблиці:

k \ p	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:

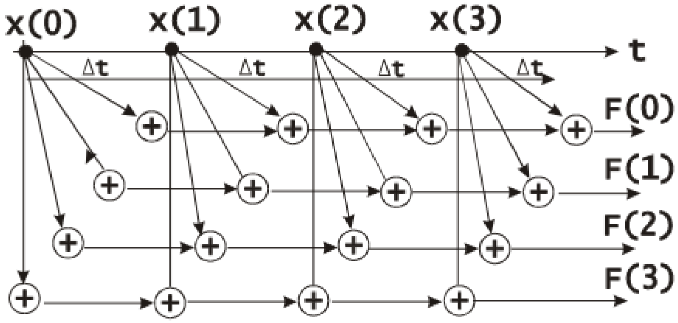
$$W_4^0 = \cos\left(\frac{2\pi}{4}\cdot 0\right) - j\sin\left(\frac{2\pi}{4}\cdot 0\right) = 1 \qquad (W_4^1 = -j;\ W_4^2 = -1;\ W_4^3 = +j)$$

Можна в пам'яті зберігати тільки 2, а решта брати з "-", якщо 1 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. 2 ДПФ 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

Для згенерованого випадкового сигналу з Лабораторної роботи N 1 відповідно до заданого варіантом (Додаток 1) побудувати його спектр, використовуючи процедуру дискретного перетворення Фур'є. Розробити відповідну програму і вивести отримані значення і графіки відповідних параметрів.

Лістинг програми із заданими умовами завдання

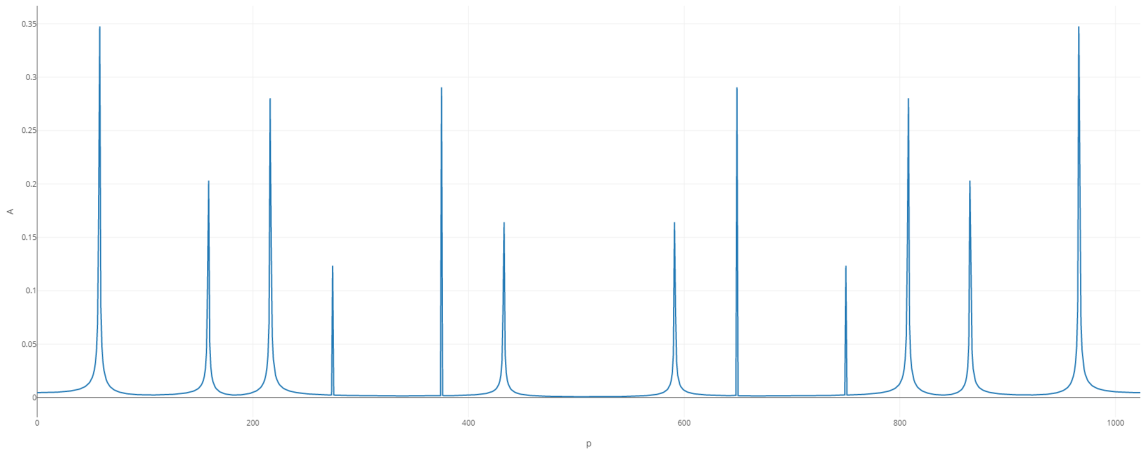
```
fun main() {
   /*
      Grade book number - 8408
      Variant - 8
    */
   val harmonic = 6
   val frequency = 1500
   val discreteCountDown = 1024

   val signalGenerator = SignalGenerator(harmonic, frequency, discreteCountDown)
   val plotsDrawer = PlotsDrawer()

   val result = mutableMapOf<Double, Double>()
   val signal = signalGenerator.generate()
   for (p in 0 until discreteCountDown) {
      var f = Complex(0.0, 0.0)
      for (k in 0 until discreteCountDown) {
         f += result.getOrDefault(p.toDouble(), 0.0) + signal.getOrDefault(k.toDouble(), 0.0) *
```

```
                    (cos(2 * PI * p * k / discreteCountDown) - sin(2 * PI * p * k /
discreteCountDown) * (1.0).j)
        }
        result[p.toDouble()] = f.abs() / discreteCountDown
    }

    val normalized = result.filter { it.key < discreteCountDown / 2 }

    plotsDrawer.createPlot(result, "p", "A")
    plotsDrawer.createPlot(normalized, "p", "A")
}

import kotlin.math.sqrt

data class Complex(val re: Double, val im: Double)

val Double.j: Complex
    get() = Complex(0.0, this)

operator fun Double.times(c: Complex): Complex {
    return Complex(this * c.re, this * c.im)
}

operator fun Double.minus(c: Complex): Complex {
    return Complex(this - c.re, - c.im)
}

operator fun Double.plus(c: Complex): Complex {
    return Complex(this + c.re, c.im)
}

operator fun Complex.plus(c: Complex): Complex {
```

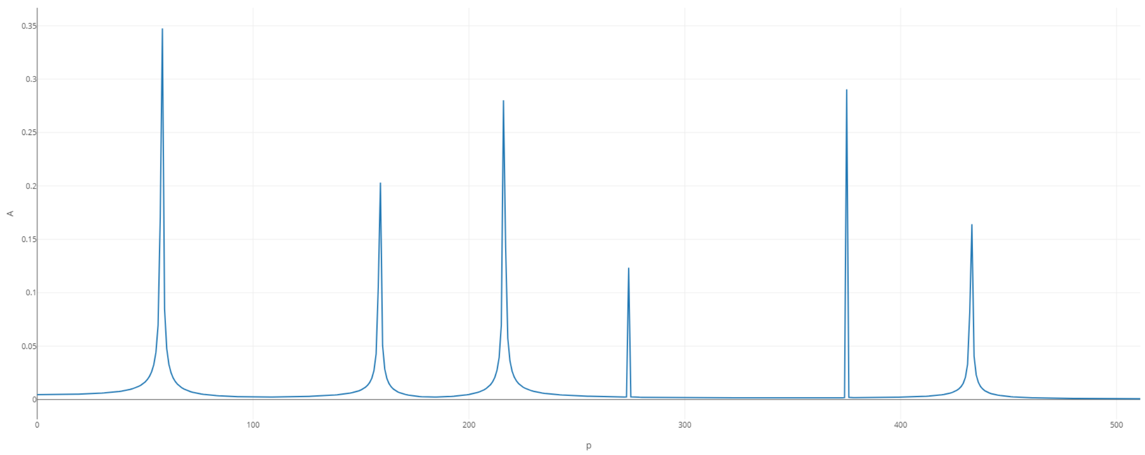
```
    return Complex(re + c.re, im + c.im)
}

fun Complex.abs() = sqrt(re * re + im * im)
```

Результати виконання кожної програми

Default



Normalized



Висновки щодо виконання лабораторної роботи

В ході виконання лабораторної роботи ознайомився з принципом роботи перетворення Фур’є. Було створено програму, яка виконує перетворення Фур’є та будує два графіки амплітудно-частотної характеристики сигналу(нормалізований та звичайний)