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 №4

з дисципліни
«Інтелектуальні вбудовані системи»

на тему
«ДОСЛІДЖЕННЯ АЛГОРИТМУ ШВИДКОГО ПЕРЕТВОРЕННЯ ФУР'Є З ПРОРІДЖУВАННЯМ ВІДЛІКІВ СИГНАЛІВ У ЧАСІ»

Виконала:

студентка групи ІП-84
Скрипник Єлена Сергіївна
номер залікової книжки: 8422

Перевірив:

викладач
Регіда Павло Геннадійович

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.

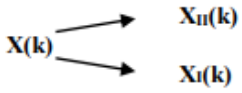
Існують формальні підходи для отримання регулярних графів ДПФ. Всі отримані алгоритми поділяються на 2 класи:

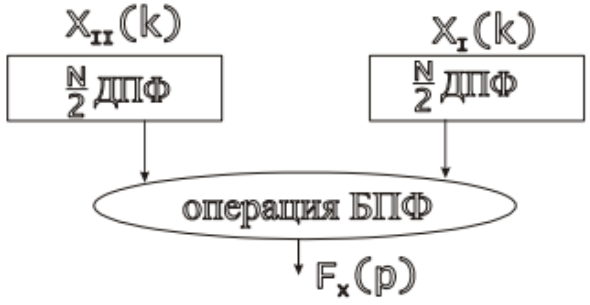
1) На основі реалізації принципу зріджені за часом $\mathbf{X_к}$

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:

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

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

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

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

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

Завдання:

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

Варіант:

22 – число гармонік в сигналі = 10; гранична частота = 1200; кількість дискретних відліків = 64.

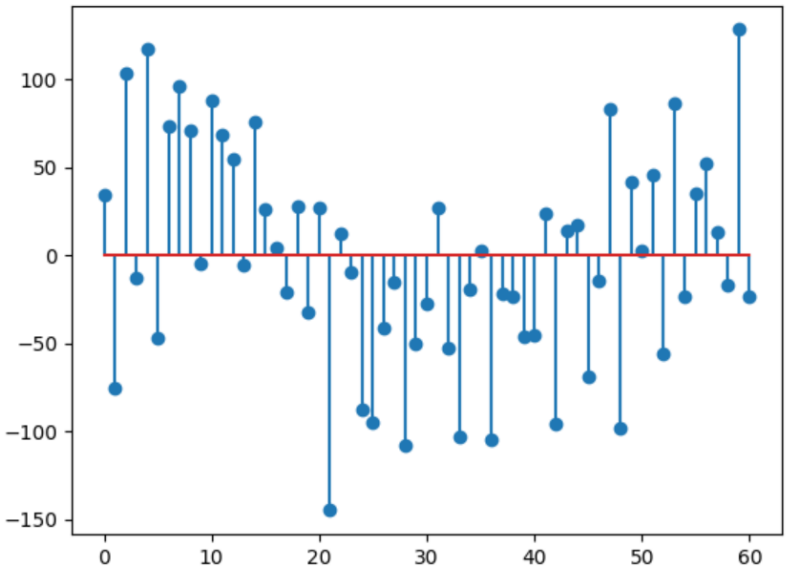
Лістинг програми:

```
import matplotlib.pyplot as plt
import random
import math
import time
from multiprocessing import Process, Pipe, Manager
from threading import Thread
```

```
n = 10
omegaMax = 1200
N = 64

k = 128
tau = 64
W = []

def Plot(g):
    A = []
    fi = []
    for i in range(n):
        A.append(random.random())
        fii = random.random() * omegaMax
        fi.append(fii)
    for i in range(k):
        res = 0
        for j in range(n):
            res += A[j] * math.sin((omegaMax / (j + 1)) * i + fi[j])
        g.append(res)
        yy = i

def Fourier(g):
    Fp = []
    Re = []
    Im = []
    for i in range(len(g)):
        Re.append(math.sin(i * 2 * math.pi / 4))
        Im.append(math.cos(i * 2 * math.pi / 4))
        W.append(math.sqrt((Re[i] * Re[i]) + (Im[i] * Im[i])))
    for k in range(len(g) - 1):
        Wpk = 0
        for p in range(len(g) - 1):
            Wpk = Wpk + W[(p * k) % len(g)]
        Fp.append(g[k] * Wpk)
    return Fp

# Accepts DFT
def FFT(g1, g2):
    fastg = []
    for p in range(int(N / 2) - 1):
        fastg.append(g1[p] + g2[p] * W[p])
    for p in range(int(N / 2), N - 2):
        fastg.append(g1[p - int(N/2)] - g2[p - int(N/2)] * W[p])
    return fastg

def ParallelCompute(g, Fp):
```

```
    Fp.extend(Fourier(g))

if __name__ == "__main__":
    x = []
    x1 = []
    x2 = []
    Fp1 = []
    Fp2 = []
    Plot(x)
    for i in range(N):
        if(i % 2 == 1):
            x2.append(x[i])
        else:
            x1.append(x[i])
    p1 = Thread(target=ParallelCompute, args=(x1, Fp1))
    p2 = Thread(target=ParallelCompute, args=(x2, Fp2))
    p1.start()
    p2.start()
    p1.join()
    p2.join()
    FastFp = FFT(Fp1, Fp2)
    plt.stem(FastFp)
    plt.show()
```

Результат виконання:



Висновки:

На даній лабораторній роботі було здійснено ознайомлення з принципами реалізації прискореного спектрального аналізу випадкових сигналів на основі алгоритму швидкого перетворення Фур'є, вивчено та досліджено особливості даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок. Також було проведено ознайомлення із особливостями організації швидкого перетворення Фур‘є з проріджуванням відліків сигналів.