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 параметрів алгоритму швидкого перетворення Фур’є»

Виконав:
студент групи ІП-84
Гудь В.В.
№ залікової книжки: ІП-8405

Перевірив:
викладач
Регіда П.Г.

Київ 2021

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.

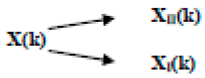
Існують формальні підходи для отримання регулярних графів ДПФ. Всі отримані алгоритми поділяються на 2 класи:

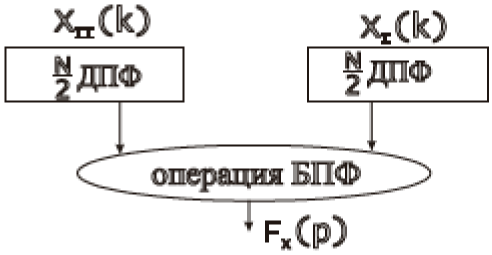
1) На основі реалізації принципу зріджені за часом $\mathbf{X_к}$

2)на основі реалізації принципу зріджені відліків шуканого спектру $\mathbf{F(p)}$.

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

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

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

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

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

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.

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

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

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.

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

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

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

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

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 зрідженим 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.

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 зріджженим 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.

$$X(2 \cdot 2k^*) \to X(4k^{**}),\ X(2 \cdot (2k^*+1)) \to X(4k^*+2)$$

$$X_I(2k^*+1) \to X(2 \cdot 2k^{**}+1) = X(4k^{**}+1);$$

$$\to X(2 \cdot (2k^*+1)+1) = X(4k^{**}+3);$$

Завдання на лабораторну роботу

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.

Умови завдання

Варіант 5:

n = 14, ω гр = 2000, N = 256

Вихідний код

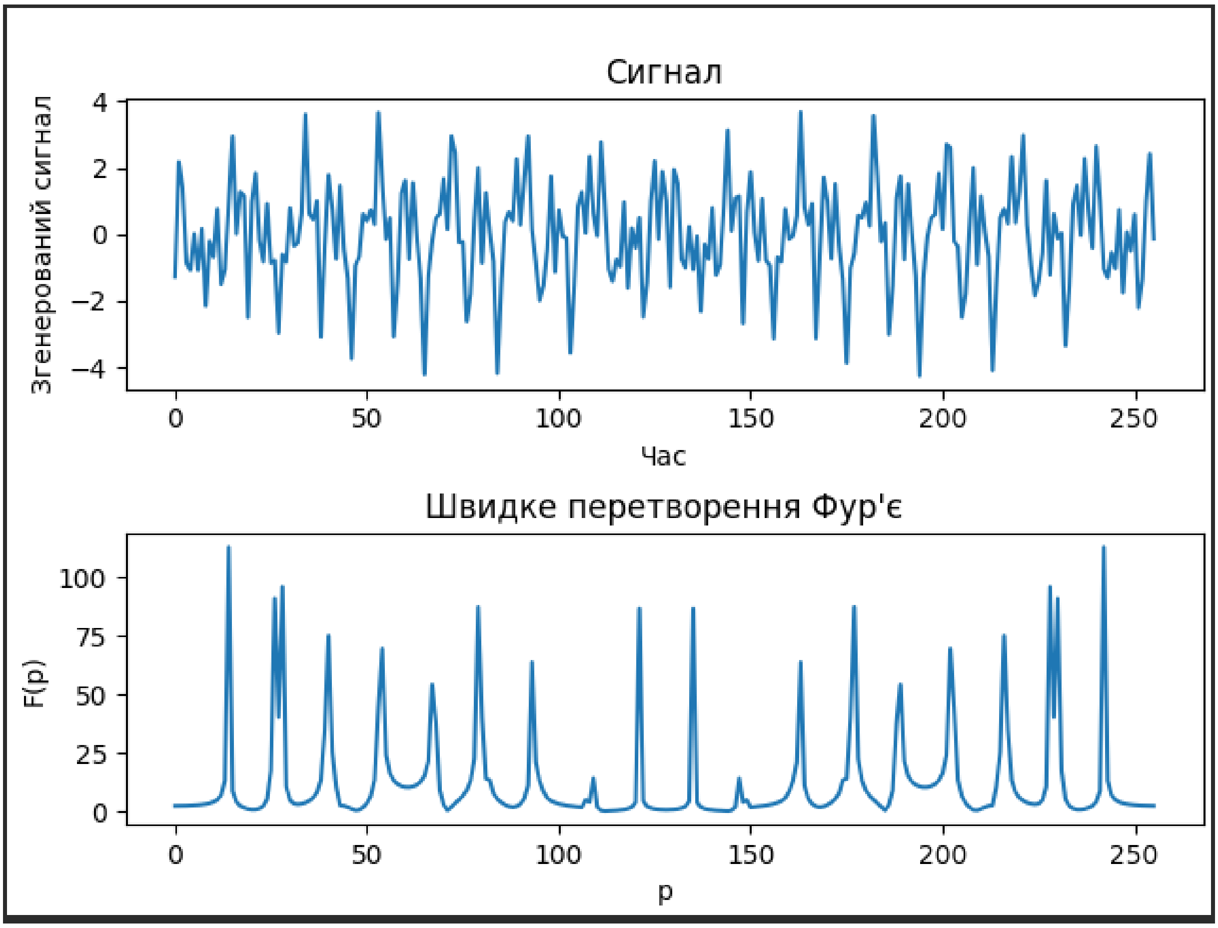
```
import numpy as np
import matplotlib.pyplot as plotter
n = 14
w = 2000
N = 264


def generate_signal(amplitude, phase, frequency, time):
```

```
    return amplitude * np.sin(frequency * time + phase)


def signal_generator(harmonics, frequency_max, samples_amount):
    signal_collection = np.zeros(samples_amount)
    for f in range(1, harmonics + 1):
        phase = np.random.uniform((-np.pi / 2), (np.pi / 2))
        amplitude = np.random.uniform(0, 1)
        frequency = (f * frequency_max) / harmonics
        for time in range(samples_amount):
            signal_collection[time] += generate_signal(amplitude, phase, frequency, time)
    return signal_collection

import numpy as np


def expectation(signal):
    sig_sum = 0
    for i in range(len(signal)):
        sig_sum += signal[i]
    expect = sig_sum / len(signal)
    return expect


def correlate(x, y, t):
    x_expectation = expectation(x)
    y_expectation = expectation(y)
    x_length = len(x)
    sig_sum = 0
    for time in range(x_length - t):
        sig_sum += (x[time] - x_expectation) * (y[time + t] - y_expectation)
    corr = sig_sum / (x_length - 1)
    return corr


def correlation(x, y, step):
    correlations = np.zeros(step)
    for t in range(step):
        correlations[t] = correlate(x, y, t)
    return correlations


def autocorrelation(x, step):
    return correlation(x, x, step)


def w(p, k, N):
    internal = 2 * np.pi * p * k / N
```

```
    return complex(np.cos(internal), np.sin(internal))


def discrete_fourier_transform(signal):
    N = len(signal)
    spectre = np.zeros(N)
    for p in range(N):
        sum = 0
        for k in range(N):
            xk = signal[k]
            wpkN = w(p, k, N)
            sum += xk * wpkN
        spectre[p] = abs(sum)
    return spectre

def fft(signal):
    N = len(signal)
    length = int(N / 2)
    spectre = np.zeros(N, dtype=complex)
    e_indices = signal[::2]
    o_indices = signal[1::2]
    for p in range(N):
        odds = 0
        evens = 0
        for k in range(length):
            wpk = w(p, k, length)
            odds += o_indices[k] * wpk
            evens += e_indices[k] * wpk
        wpk_odd = w(1, p, N)
        spectre[p] = evens + wpk_odd * odds
    return spectre



signal = signal_generator(n, w, N)

spectre = abs(fft(signal))

figure, axis = plotter.subplots(2, 1)

plotter.subplots_adjust(left=0.1, top=0.9, bottom=0.1, right=0.99, hspace=0.5)

axis[0].plot(range(N), signal)
axis[0].set_title("Сигнал")
axis[0].set(xlabel='Час', ylabel='Згенерований сигнал')

axis[1].plot(range(N), spectre)
axis[1].set_title("Швидке перетворення Фур'є")
```

axis[1].set(xlabel='p', ylabel='F(p)')
plotter.show()

Результати роботи програми



Висновки
Під час даної лабораторної роботи ми ознайомились з принципами реалізації прискореного спектрального аналізу випадкових сигналів на основі алгоритму швидкого перетворення Фур'є, вивчили та дослідили особливості даного алгоритму з використанням засобів моделювання і сучасних програмних оболонок.

.