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»

Виконала:
студентка
групи ІП-83
Гомілко Діана Володимирівна

Перевірив:
Регіда Павло Геннадійович

Київ 2021

Основні теоретичні відомості, необхідні для виконання лабораторної роботи

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

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

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.

При переході до процедури БПФ з зрідженим за часом від класичного ДПФ вже з'являється кілька рівнів перетворення даних і на кожному рівні ми в деякій послідовності поділяємо знову на парні і непарні відліки. У БПФ з зрідженим за часом розмірність замінних перетворень ДПФ знизу вгору.

Загальна кількість рівнів заміщення в БПФ:

$$m = \log^2(N)\ ,\ N\text{-ціла ступінь 2.}$$

Тут мова йде про найпростішому БПФ з основою 2, тобто в базові операції є 2-х точкове БПФ. На кожному рівні БПФ з зрідженим за часом треба здійснити додаткову перестановку відліків на парні і непарні, що призводить до загального недоліку алгоритму - необхідність спеціальної перестановки відліків досліджуваного сигналу перед перетворенням.

Для реалізації ШПФ з зрідженим за часом треба здійснювати двійкову инверсную перестановку відліків або просто обчислити адреси.

Базова операція БПФ може виконуватися в один такт по паралельній схемі з 4-ма множниками; в 2 такти по послідовно-паралельною схемою з 2 множниками; в 4 такти по послідовній схемі з одним помножувачем і декількома регістрами.

Умови завдання для варіанту

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: 04 (номер заліковки — 8304):

Варіант	Число гармонік в сигналі, n	Гранична частота, ω	Кількість дискретних відліків, N
4	12	2400	1024

Лістинг програми із заданими умовами завдання

fft.py

```
import math
import sys
sys.path.append('../')
from lab2_1.dft import fourierCoefficient

def fastFourierTransform(signal):
N = len(signal)
if N == 1: return signal
length = int(N / 2)
spectre = [None] * N
evens = fastFourierTransform(signal[::2])
odds = fastFourierTransform(signal[1::2])
for p in range(length):
w = fourierCoefficient(p, N)
product = odds[p] * w
spectre[p] = evens[p] + product
spectre[p + length] = evens[p] - product
return spectre
```

complexity.py

```
from fft import fastFourierTransform
import time
import sys
sys.path.append('../')
from lab1.signalGenerator import createSignal
from lab2_1.dft import discreteFourierTransform

def getComplexity(stepsCount, harmonics, maxFrequency, type):
elapsed = []
size = []
for i in range(stepsCount):
count = int(2 ** (i + 1))
size.append(count)
signal = createSignal(harmonics, maxFrequency, count)
start = time.perf_counter()
if (type == "DFT"):
discreteFourierTransform(signal, "list")
```

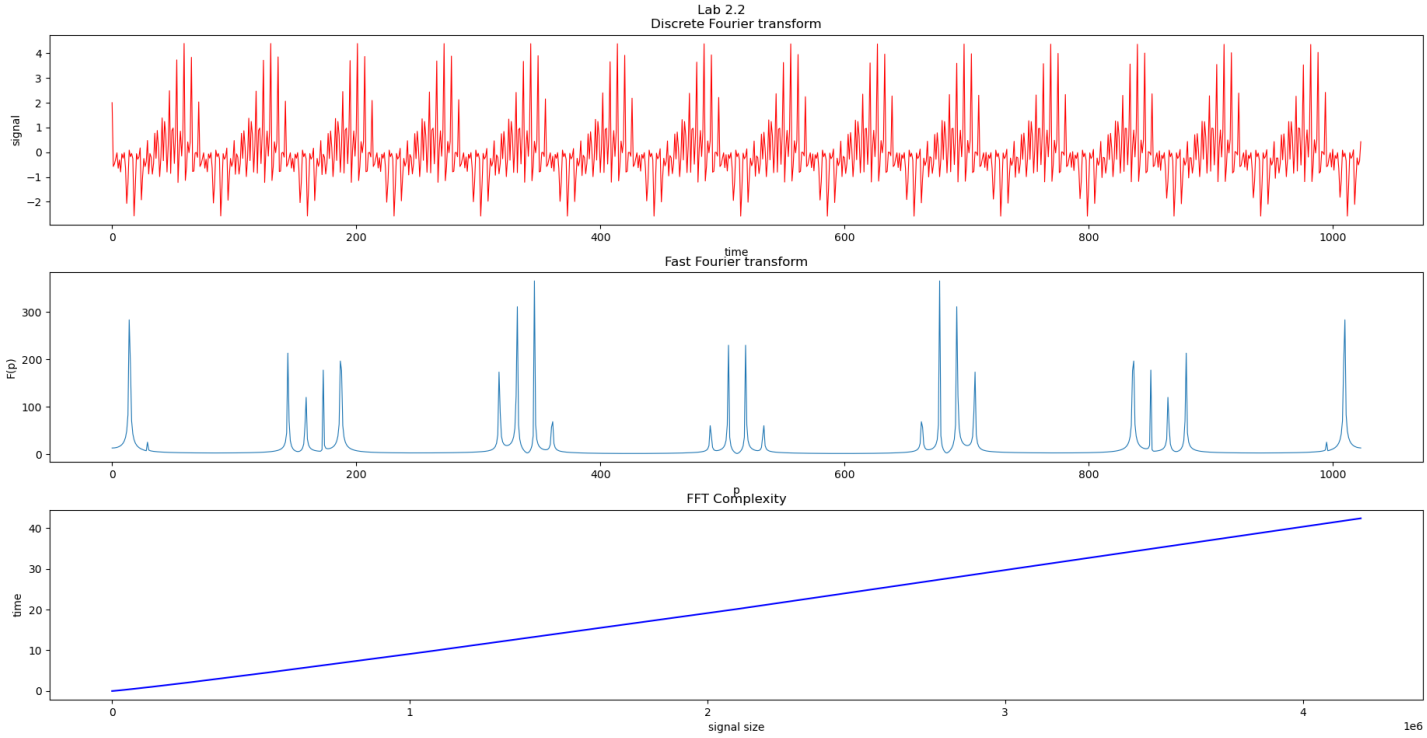
```
else:
fastFourierTransform(signal)
stop = time.perf_counter()
elapsed.append(stop - start)
return size, elapsed
```

lab2-2.py

```
import matplotlib.pyplot as plt
from fft import fastFourierTransform
from complexity import getComplexity
import sys
sys.path.append('../')
from lab1.signalGenerator import createSignal
from lab2_1.dft import discreteFourierTransform

HARMONICS = 12
MAX_FREQUENCY = 2400
DISCRETE_CALLS = 1024
COMPLEXITY_COUNT_LOOPS = 22

toRealNum = lambda x: list(map(lambda a: abs(a), x))

signal = createSignal(
HARMONICS,
MAX_FREQUENCY,
DISCRETE_CALLS
)
spectre = toRealNum(fastFourierTransform(signal))

size, elapsedFFT = getComplexity(
COMPLEXITY_COUNT_LOOPS,
HARMONICS,
MAX_FREQUENCY,
"FFT"
)

fig, axs = plt.subplots(3, 1)
plt.subplots_adjust(left=0.05, top=0.94, bottom=0.05, right=0.97, hspace=0.25)
fig.suptitle('Lab 2.2')

axs[0].plot(signal, color='r', linewidth=0.8)
axs[0].set_title('Discrete Fourier transform')
axs[0].set(xlabel='time', ylabel='signal')

axs[1].plot(spectre, linewidth=0.8)
axs[1].set_title('Fast Fourier transform')
axs[1].set(xlabel='p', ylabel='F(p)')

axs[2].plot(size, elapsedFFT, color='b')
axs[2].set_title('FFT Complexity')
axs[2].set(xlabel='signal size', ylabel='time')
fig.savefig('graphs/lab2-2.png')
plt.show()

plt.show()
```

Результати виконання кожної програми

Графіки згенерованого випадкового сигналу, спектру, отриманого в результаті швидкого перетворення Фур’є, та складності процедури ШПФ:



Висновки щодо виконання лабораторної роботи

Під час виконання даної лабораторної роботи ми ознайомилися 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. Було створено програму, яка, користуючись процедурою ШПФ, створює спектр для випадкового сигналу. Результатом виконання стали графіки сигналу та спектру, а також графік складності ШПФ при збільшенні розміру сигналу.