Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет електроніки та комп’ютерних технологій

Звіт
про виконання лабораторної роботи №4

Обчислення
ШВИДКОГО ПЕРЕТВОРЕННЯ ФУР’Є (ШПФ, FFT)

Виконала
студентка групи ФеС-21
Шавало А.А.

Перевірив
Вдовиченко В. М.

Львів 2024 р.

Мета : Дослідити та реалізувати алгоритм Швидкого Перетворення Фур’є (ШПФ) для оптимізації обчислень у спектральному аналізі сигналів, а також порівняти ефективність ШПФ із традиційним дискретним перетворенням Фур'є (ДПФ) у різних програмних та апаратних середовищах.

Теоретичні відомості :

Основні теоретичні відомості:

1. **Дискретне перетворення Фур'є (ДПФ):**
   - ДПФ перетворює дискретний сигнал у часовій області в частотну область, представляючи сигнал як суму синусоїд і косинусоїд різних частот.
   - Формула ДПФ:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j \cdot \frac{2\pi}{N} \cdot n \cdot k}, \quad k = 0, 1, \ldots, N-1$$

   Де:
   - $x(n)$ — вхідний сигнал у часовій області;
   - $X(k)$ — результат у частотній області;
   - N — кількість точок (розмір сигналу);
   - $e^{-j \cdot \frac{2\pi}{N} \cdot n \cdot k}$ — комплексні експоненти.
2. **Складність ДПФ:**
   - Виконання ДПФ напряму має складність $O(N^2)$, оскільки потрібно виконати N обчислень для кожного значення $X(k)$, і так для всіх N значень k. Це обчислювально дорого для великих сигналів.
3. **Алгоритм ШПФ (FFT):**
   - ШПФ є вдосконаленим методом обчислення ДПФ, який дозволяє скоротити кількість операцій з $O(N^2)$ до $O(N \log N)$.
   - Алгоритм базується на принципі **"розділяй і володарюй"**: сигнал ділиться на частини (зазвичай на парні і непарні індекси), і потім для кожної частини рекурсивно виконується ДПФ. Це дозволяє суттєво зменшити кількість необхідних обчислень.
   - Найпоширенішою версією ШПФ є **алгоритм Кулі-Тьюкі (Cooley-Tukey)**.
4. **Алгоритм Кулі-Тьюкі:**
   - Він працює для сигналів, розмір яких є ступенем двійки ($N = 2^m$).
   - Суть алгоритму полягає в тому, що ДПФ розбивається на два менші ДПФ: один для парних індексів сигналу, інший для непарних. Це робиться рекурсивно, поки не залишаться ДПФ розміром 2, які можна обчислити дуже швидко.
   - Наприклад, якщо сигнал має 8 елементів, ШПФ розбиває його на дві групи по 4 елементи, потім кожну групу ще на 2 групи по 2 елементи, і так далі.

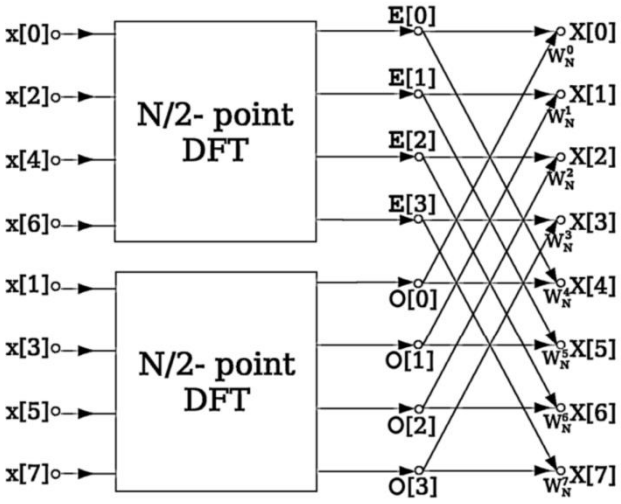

Рисунок 2.1 – структура алгоритму ШПФ.

Завдання : Скористайтеся програмою прямого обчислення ШПФ і знайдіть коефіцієнти такої дискретної в часі послідовності:

x(n)={5, 7, 3, 3, 0, 0, 15, 32, 13, 7}

Лістинг програми :

```
import numpy as np


def fft(x):
    N = len(x)
    if N <= 1:
        return x
    even = fft(x[0::2])
    odd = fft(x[1::2])
    T = [np.exp(-2j * np.pi * k / N) * odd[k] for k in range(N
// 2)]
    return [even[k] + T[k] for k in range(N // 2)] + [even[k]
- T[k] for k in range(N // 2)]


def ifft(x):
    N = len(x)
    x_conj = np.conjugate(x)
    result = fft(x_conj)
    return np.conjugate(result) / N
```

```
def main():
    data = list(map(int, input("Ввести послідовність:
").split()))

    while len(data) & (len(data) - 1) != 0:
        data.append(0)

    data = np.array(data, dtype=complex)

    fft_result = fft(data)
    print("Результат прямого ШПФ:")
    for r in fft_result:
        print(f"{r.real:.3f} + {r.imag:.3f}j")

    ifft_result = ifft(fft_result)
    print("¥nРезультат оберненого ШПФ:")
    for r in ifft_result:
        print(f"{r.real:.3f} + {r.imag:.3f}j")


if __name__ == "__main__":
    main()
#5 7 3 3 0 0 15 32 13 7
```

Результат виконання :

```
Ввести послідовність: 5 7 3 3 0 0 15 32 13 7
Результат прямого ШПФ:
85.000 + 0.000j
-44.901 + -27.745j
48.406 + 22.607j
-14.532 + -41.144j
-0.000 + 21.000j
15.503 + -15.688j
-12.406 + -1.393j
11.931 + -2.290j
-13.000 + 0.000j
11.931 + 2.290j
-12.406 + 1.393j
15.503 + 15.688j
0.000 + -21.000j
-14.532 + 41.144j
48.406 + -22.607j
-44.901 + 27.745j
```