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 № 2.1 - 2.2

з дисципліни

“Інтелектуальні вбудовані системи”

на тему

“ДОСЛІДЖЕННЯ ПАРАМЕТРІВ АЛГОРИТМУ ДИСКРЕТНОГО ПЕРЕТВОРЕННЯ ФУР’Є",
“ДОСЛІДЖЕННЯ АЛГОРИТМУ ШВИДКОГО ПЕРЕТВОРЕННЯ ФУР'Є З ПРОРІДЖУВАННЯМ ВІДЛІКІВ СИГНАЛІВ У ЧАСІ”

Виконала:
Студент групи ІП-84
Василашкоа А.О.
№ ЗК: ІП-8402

Перевірив:
викладач Регіда П.Г.

Київ 2021

Основні 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 $x(k)$

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

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

На всьому інтервалі подання сигналів **T**, **2π** - один період низьких частот. Щоб підвищити точність треба збільшити інтервал **T.**

$$t \to t_k \to k\Delta t \to k; \quad \Delta t = \frac{T}{N} = \frac{1}{k_{зш.}} \cdot f'гр.$$

ДПФ - проста обчислювальна процедура типу звірки (тобто Σ-е парних множень), яка за складністю також має оцінку $\mathbf{N^2 + N}$. 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_N^{pk} 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(2π)**.. Т.ч. в ПЗУ треба зберігати N коефіцієнтів дійсних і уявних частин. Якщо винести знак коефіцієнта можна зберігати **N/2** коефіцієнтів.

2π/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:

$$F_x(p) = \sum_{k=0}^{N-1} x(k) \cdot W_N^{pk}$$

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.

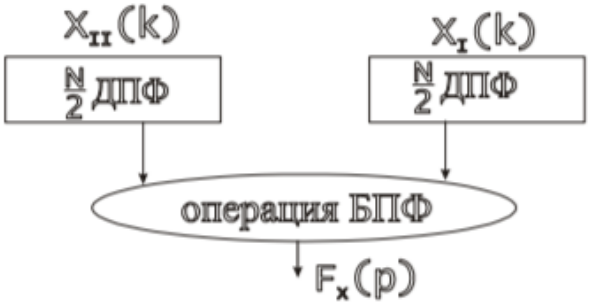
Існують формальні підходи для отримання регулярних графів ДПФ. Всі отримані алгоритми поділяються на 2 класи:

1) На основі реалізації принципу зріджені за часом $X_к$

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.



Умови завдання для варіанту

№ ЗК: 8402, тому число гармонік в сигналі (n) = 10, гранична частота ($w_{гр}$) = 900, кількість дискретних відліків (N) = 256. Для згенерованого випадкового сигналу побудувати його спектр, використовуючи процедуру дискретного перетворення Фур'є та швидкого перетворення Фур'є з проріджуванням відліків сигналу за часом. Розробити відповідну програму і вивести отримані значення і графіки відповідних параметрів.

Лістинг програми із заданими умовами завдання

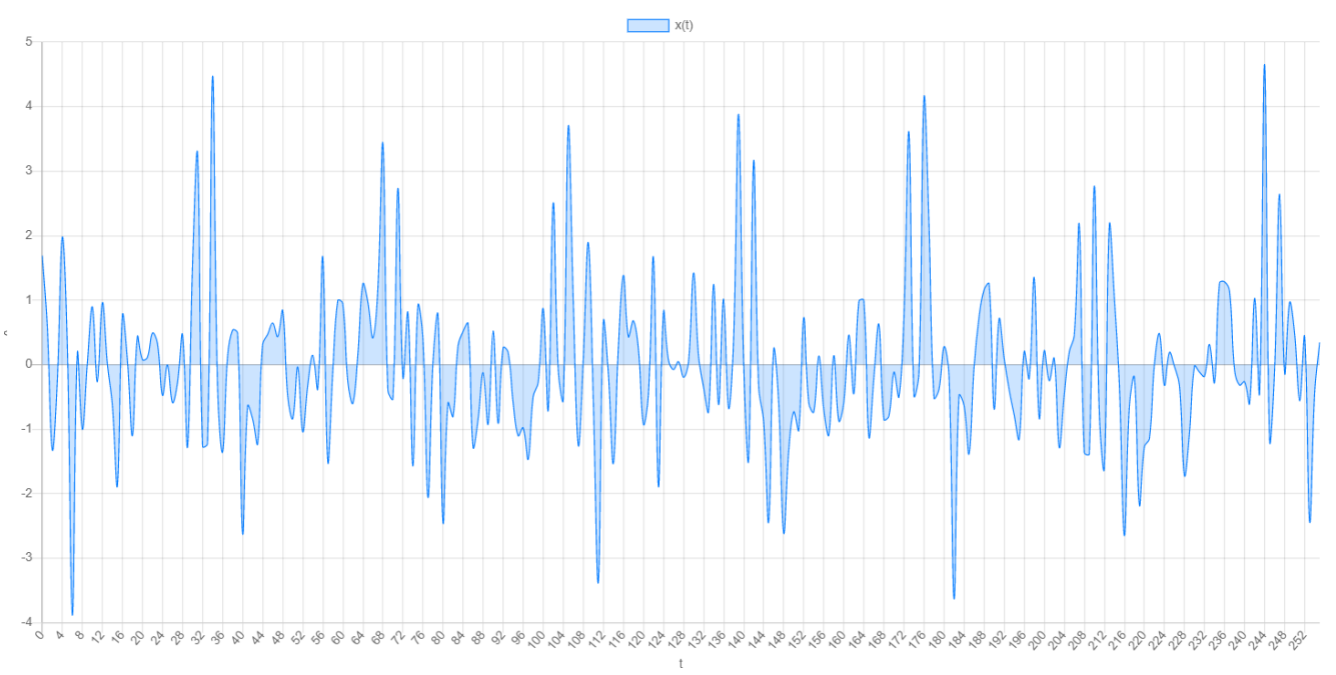
```
const generateRandomSignals = (n, wl, N) => {
  let signals = new Array(N);
  for (let i = 0; i < N; ++i) signals[i] = 0;

  let Wp = 0;

  for (let i = 1; i <= n; i++) {
    Wp += wl / n;

    for (let t = 0; t < N; t++) {
      let fp = Math.random();
      let Ap = Math.random();

      signals[t] += Ap * Math.sin(Wp * t + fp);
    }
  }

  return signals;
};

const sum = (signals) => signals.reduce((p, c) => p + c, 0);
const average = (signals) => sum(signals) / signals.length;
const dispercy = (signals) => {
  let mx = average(signals);
  return sum(signals.map((xt) => Math.pow(xt - mx, 2))) / (signals.length - 1);
};

function complex(real, imag) {
```

```
  return {
    real: real,
    imag: imag,
  };
}

function dft(samples) {
  var len = samples.length;
  var arr = Array(len);
  var invlen = 1 / len;
  for (var i = 0; i < len; i++) {
    arr[i] = complex(0, 0);
    for (var n = 0; n < len; n++) {
      var theta = -Math.PI * 2 * i * n * invlen;
      var costheta = Math.cos(theta);
      var sintheta = Math.sin(theta);
      if (samples[n].real == undefined) samples[n] = complex(samples[n], 0);
      arr[i].real += samples[n].real * costheta - samples[n].imag * sintheta;
      arr[i].imag += samples[n].real * sintheta + samples[n].imag * costheta;
    }
  }
  return arr;
}

function fft(samples) {
  var len = samples.length;
  var arr = Array(len);

  if (len == 1) {
    if (samples[0].real == undefined) return [complex(samples[0], 0)];
    else return [complex(samples[0].real, samples[0].imag)];
  }

  let even = samples.filter((sample, index) => index % 2 == 0);
  let odd = samples.filter((sample, index) => index % 2 == 1);

  var arrEven = fft(even);
  var arrOdd = fft(odd);

  var invlen = 1 / len;
```

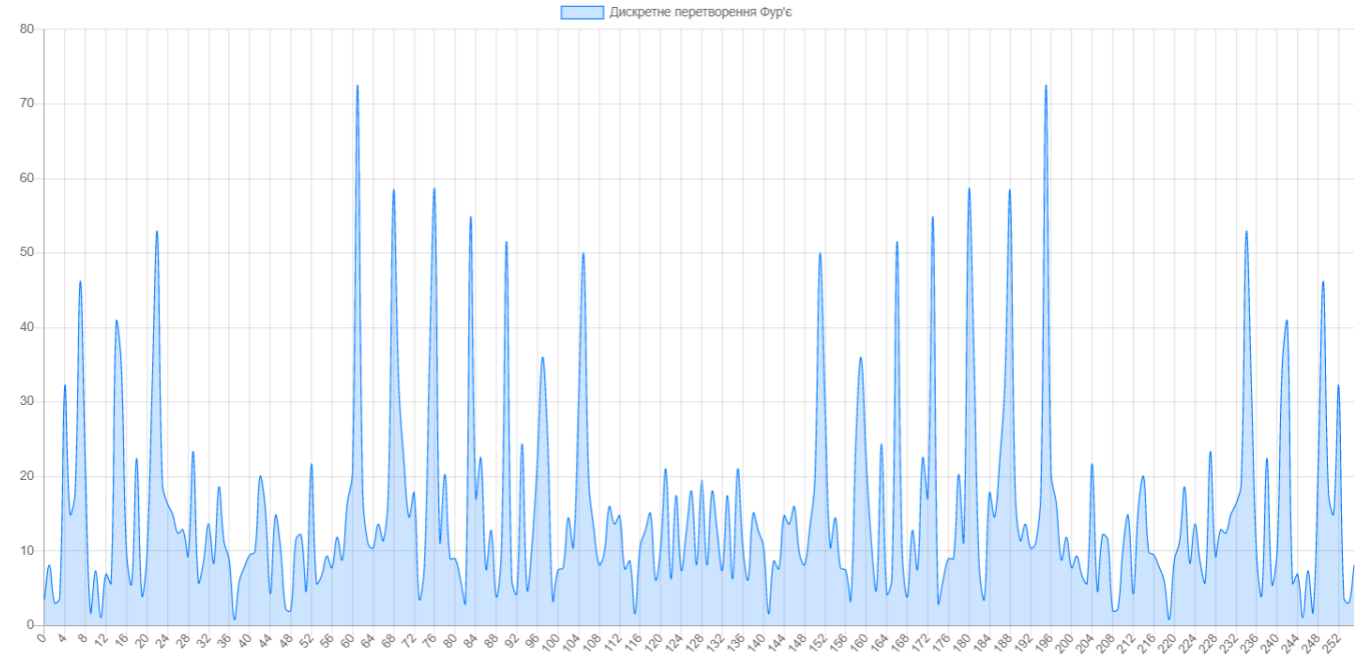
```
  for (var k = 0; k < len / 2; k++) {
    let temp = complex(0, 0);

    var theta = -Math.PI * 2 * k * invlen;
    var costheta = Math.cos(theta);
    var sintheta = Math.sin(theta);

    temp.real = arrOdd[k].real * costheta - arrOdd[k].imag * sintheta;
    temp.imag = arrOdd[k].real * sintheta + arrOdd[k].imag * costheta;

    arr[k] = complex(arrEven[k].real + temp.real, arrEven[k].imag + temp.imag);

    arr[k + len / 2] = complex(
      arrEven[k].real - temp.real,
      arrEven[k].imag - temp.imag
    );
  }

  return arr;
}
```

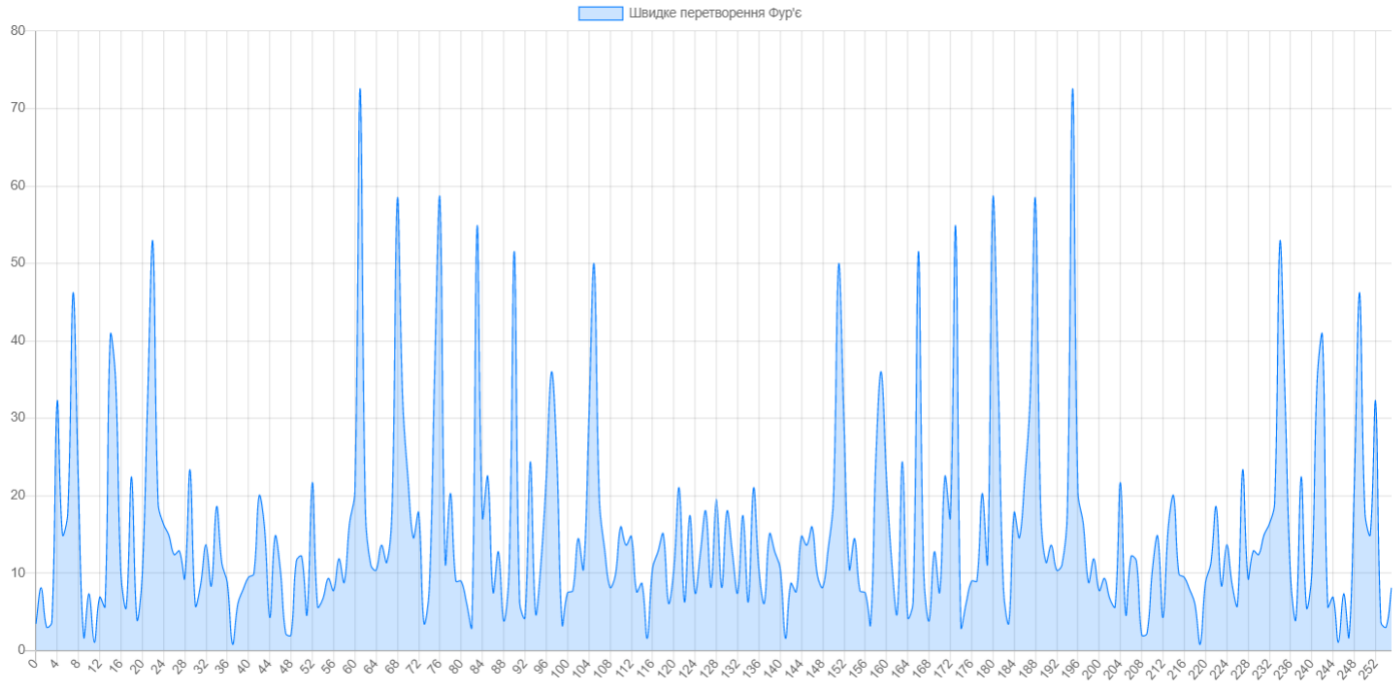
Результати виконання програми

Сигнали



Дискретне перетворення Фур’є та швидке перетворення Фур’є





Висновки

Під час даної лабораторної роботи ми навчились виділяти частотний спектр з сигналу за допомогою дискретного перетворення Фур’є та як можна пришвидшити дискретне перетворення Фур’є.