МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №1

з дисципліни «Системи реального часу»

на тему *«Дослідження і розробка моделей випадкових сигналів. Аналіз їх характеристик»*

Виконав:

студент гр. ІП-83

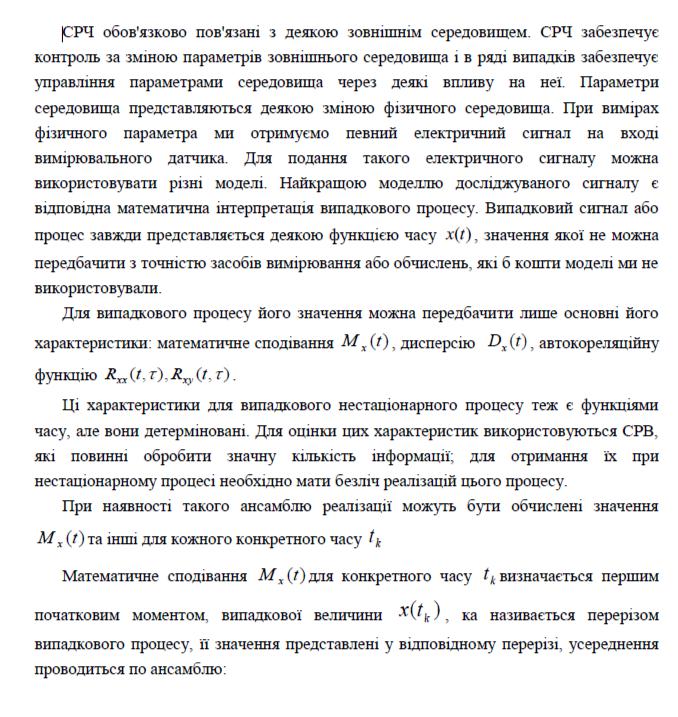
Канаєв Є.Д.

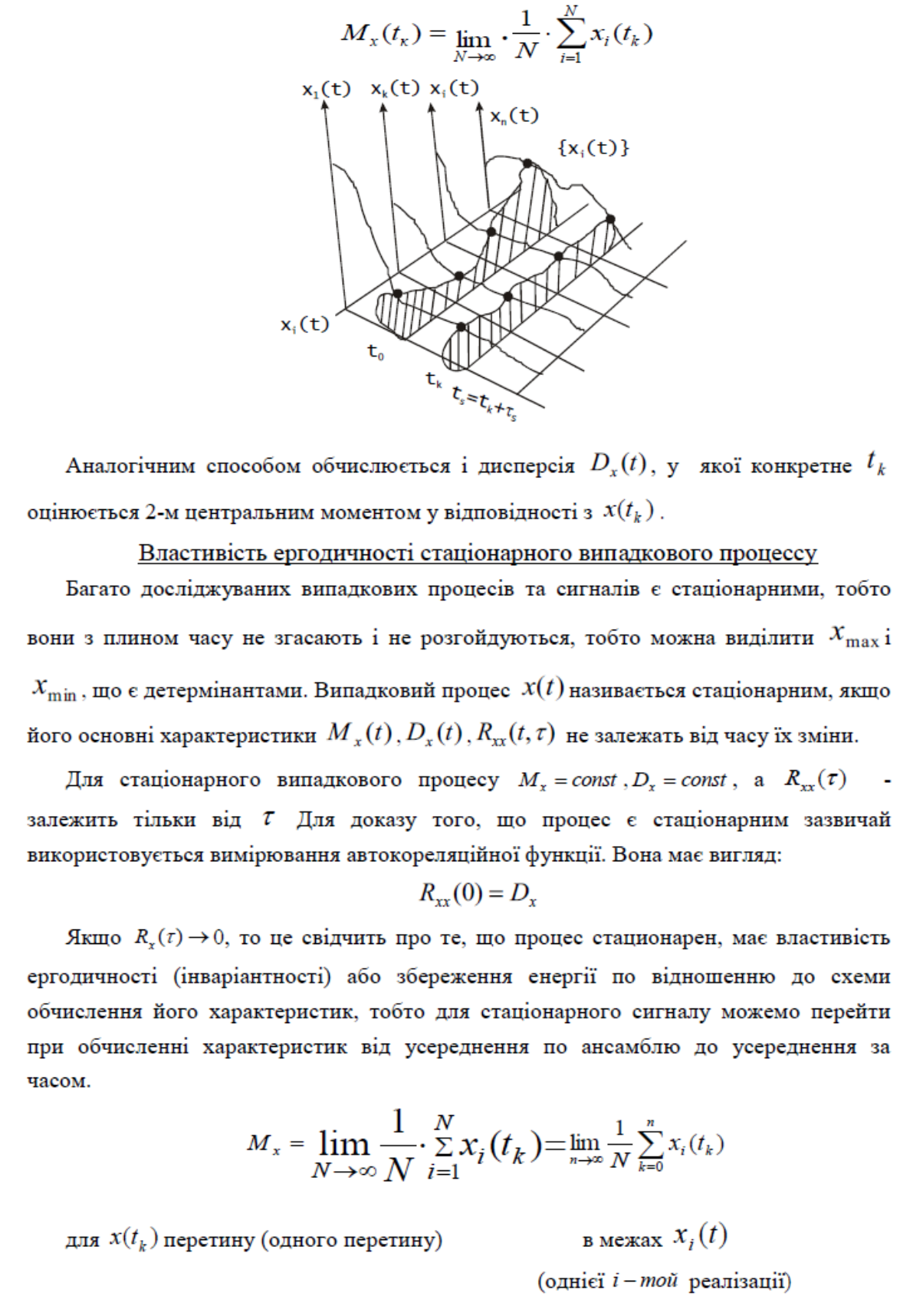
Перевірив:

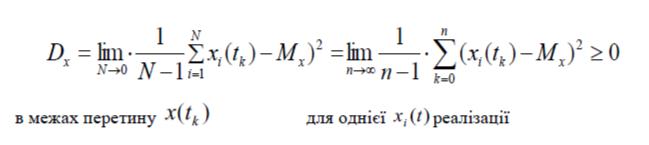
Регіда П.Г.

Київ 2021

**Основні теоретичні відомості:**

****





**Завдання:**

Заліковка 8310

Варіант 10

Число гармонік в сигналі n = 14

Гранична частота, ωгр = 1700

Кількість дискретних відліків, N = 64 **Лістинг програми**:

1. Програма для генерації сигналу, розрахунку мат очікування та дисперсії

#include "signal.h"

#include <algorithm>

#include <bits/c++config.h>

#include <random>

std::vector<double>

generate\_signal(ulong harm, ulong freq, ulong inter, double dt) { std::random\_device r;

std::default\_random\_engine rng(r());

std::uniform\_real\_distribution<double> dist;

std::vector<double> signal(inter, 0.0);

const auto dw = static\_cast<double>(freq) / static\_cast<double>(harm); auto w = dw;

for (ulong h = 0; h < harm; h++) {

const auto a = dist(rng);

const auto phi = dist(rng);

for (ulong t = 0; t < inter; t++) {

const auto x = a \* std::sin(w \* static\_cast<double>(t) \* dt + phi); signal[t] += x;

}

w += dw;

}

return signal;

}

static double average(std::span<const double> a) { return std::accumulate(a.begin(), a.end(), 0.0) / a.size();

}

double correlation(std::span<const double> a, std::span<const double> b) { auto ma = average(a);

auto mb = average(b);

double sum = 0.0;

for (std::size\_t i = 0; i < a.size(); i++) {

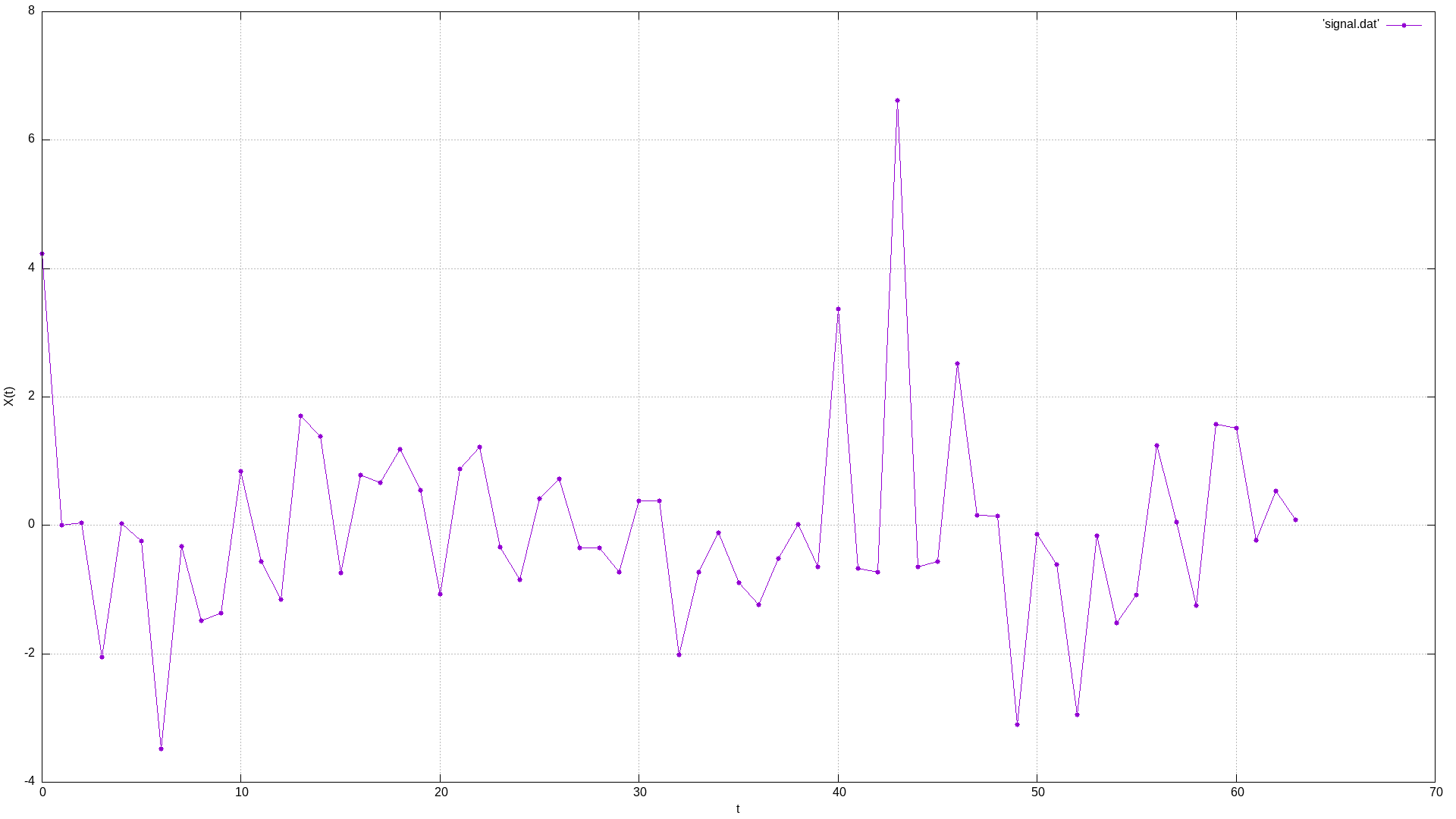
sum += (a[i] - ma) \* (b[i] - mb);

}

return sum / a.size();

}

**Отриманий графік**:



1. Програма для вимірювання складності обчислень в залежності від часу (кількості дискретних відліків)

*#include "signal.h"*

*#include <iostream>*

*constexpr ulong harm\_low = 2;*

*constexpr ulong harm\_hi = 2 << 18;*

*constexpr ulong harm\_step = 2;*

*int main() {*

*std::cout << "# n\tO(n)\n";*

*for (auto harm = harm\_low; harm < harm\_hi; harm \*= harm\_step) { auto dur = measure([harm]() {*

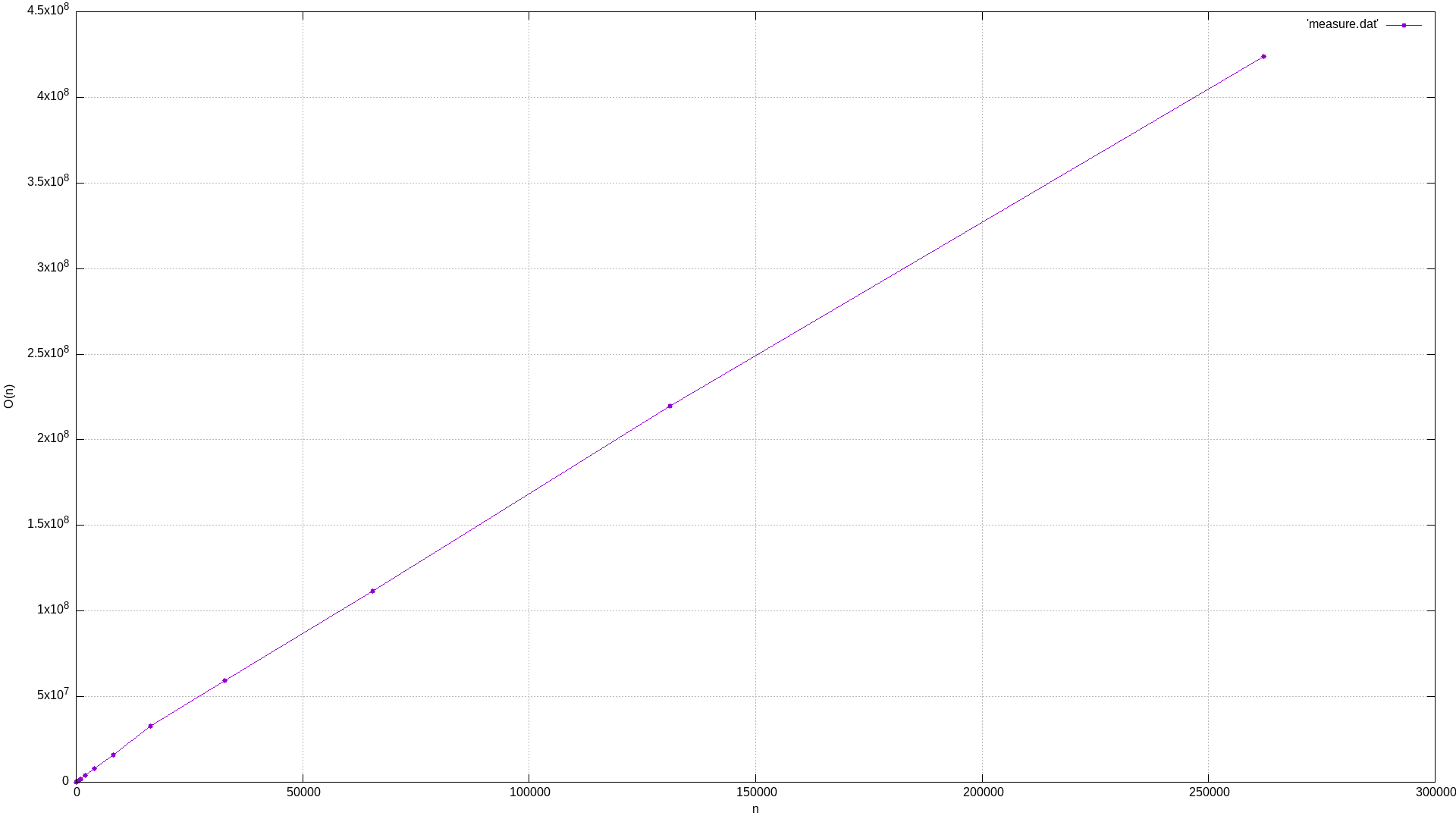
*generate\_signal(harm, params::freq, params::inter, params::dt); });*

*std::cout << harm << '\t' << dur << '\n';*

*}*

*}*

**Результати вимірювань**:



Графік демонструє складність, наближену до теоретичної O(N).