|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА №6**

**«Исследование качества генератора случайных чисел»**

**ДИСЦИПЛИНА: «Моделирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-72Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Никитенко У.В. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2023

**Цель:** изучить и практически освоить оценки качества генераторов случайных чисел (ГСЧ) в различных системах программирования по заданным теоретическим показателям, с помощью критериев согласия и с помощью нормированной автокорреляционной функции на предмет независимости случайных чисел.

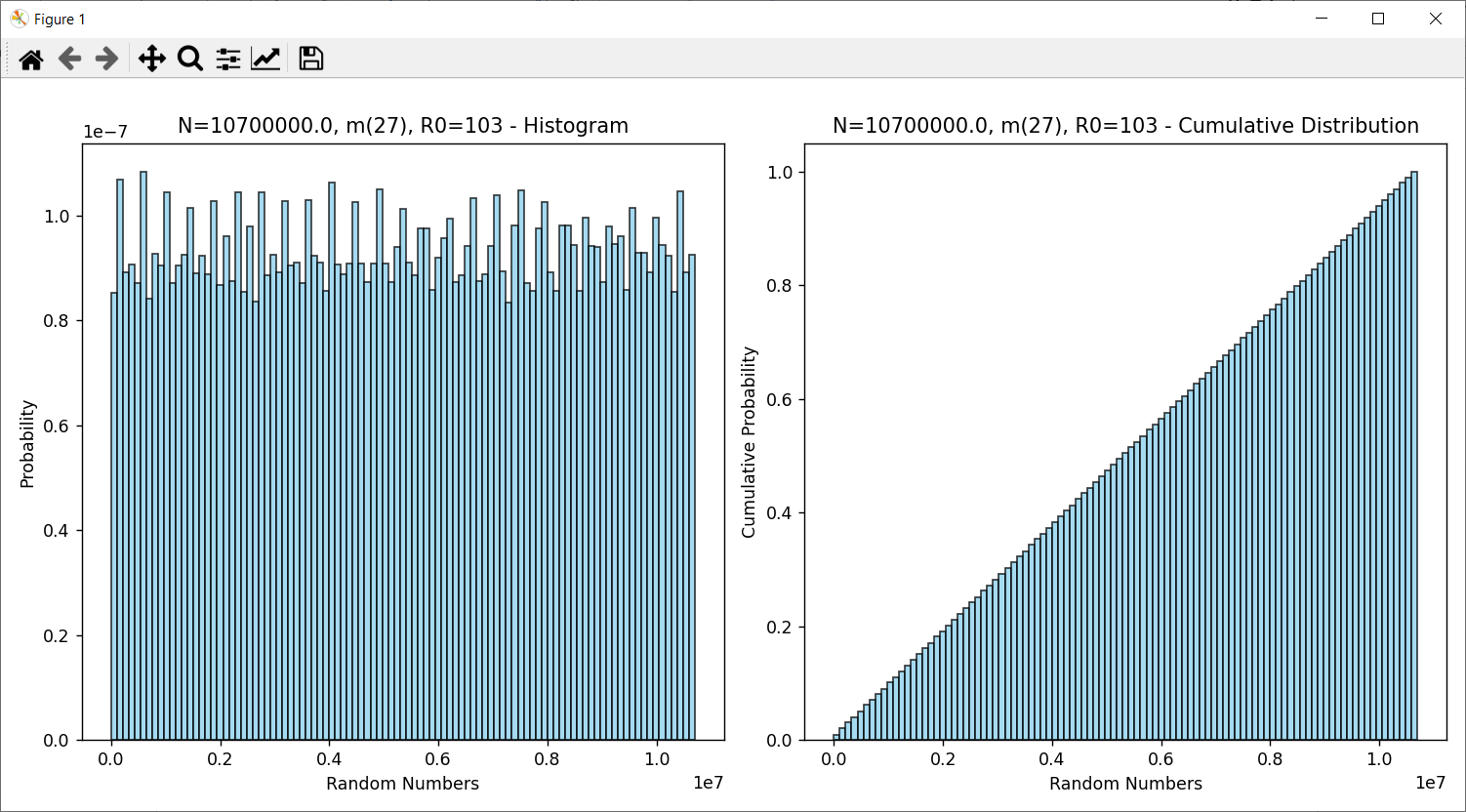
**Задачи:** проанализировать анализ качества ГСЧ статистическими методами: по критерию отклонения математического ожидания, дисперсии, среднего квадратического отклонения;

**Вариант 7**

**Задание 4.2(7-9):**

1. Полагая в формуле , написать в MATLAB(PYTHON) программу формирования случайных чисел, приняв следующие числа для расчета модуля:
2. В качестве первого назначаемого случайного числа принять следующие значения (m – массив простых чисел, сформированный с помощью выражения m = primes(N)):
3. Вычислить период формируемой случайной последовательности;
4. Произвести статистический анализ созданного ГСЧ по линейному конгруэнтному методу;
5. Построить гистограммы полученных распределений случайных чисел.
6. Построить функции плотности и распределения для сформированных выборок случайных чисел. Совместить диаграммы с теоретическими функциями

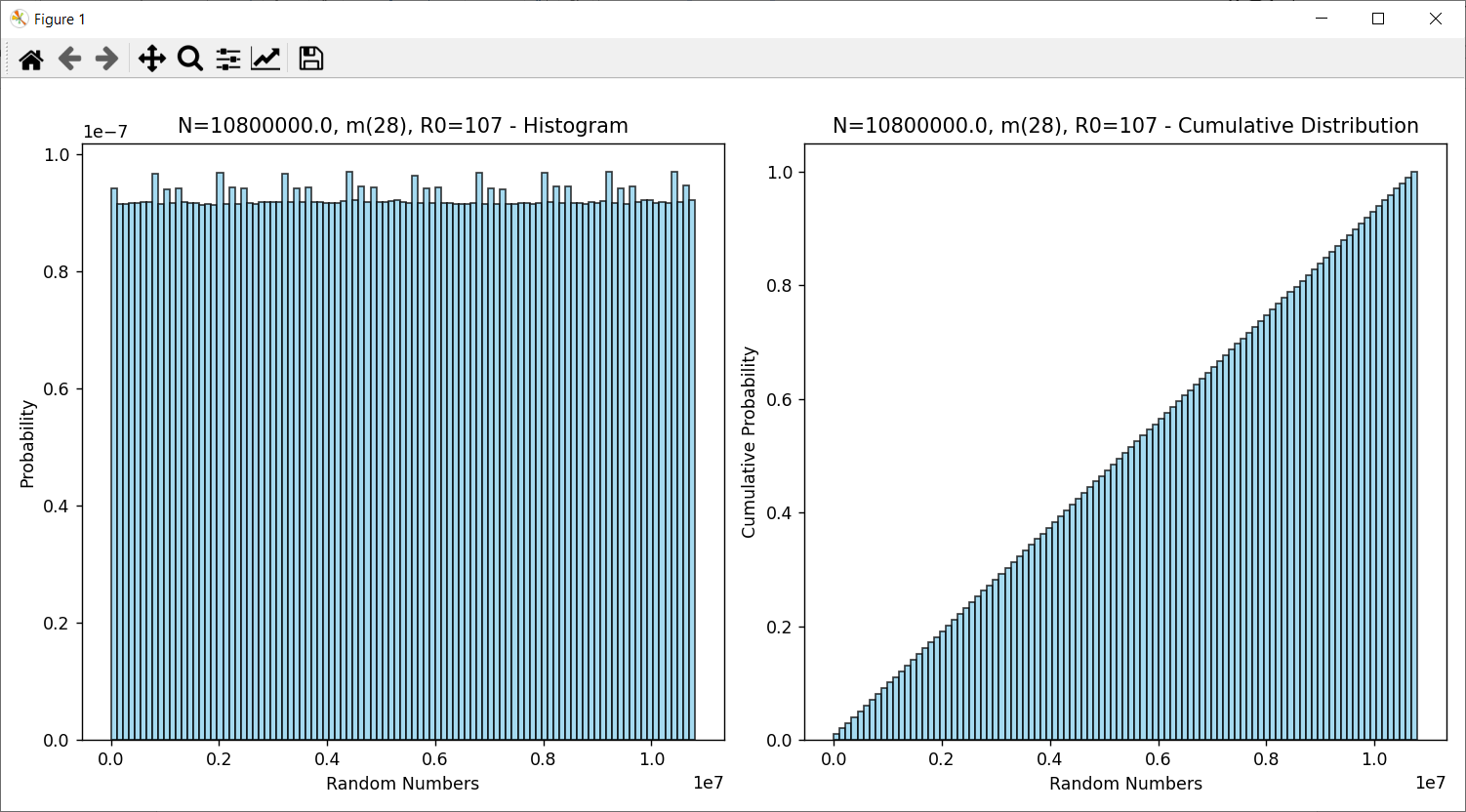
**Решение:**



**Рис. 1.1.** Результат для



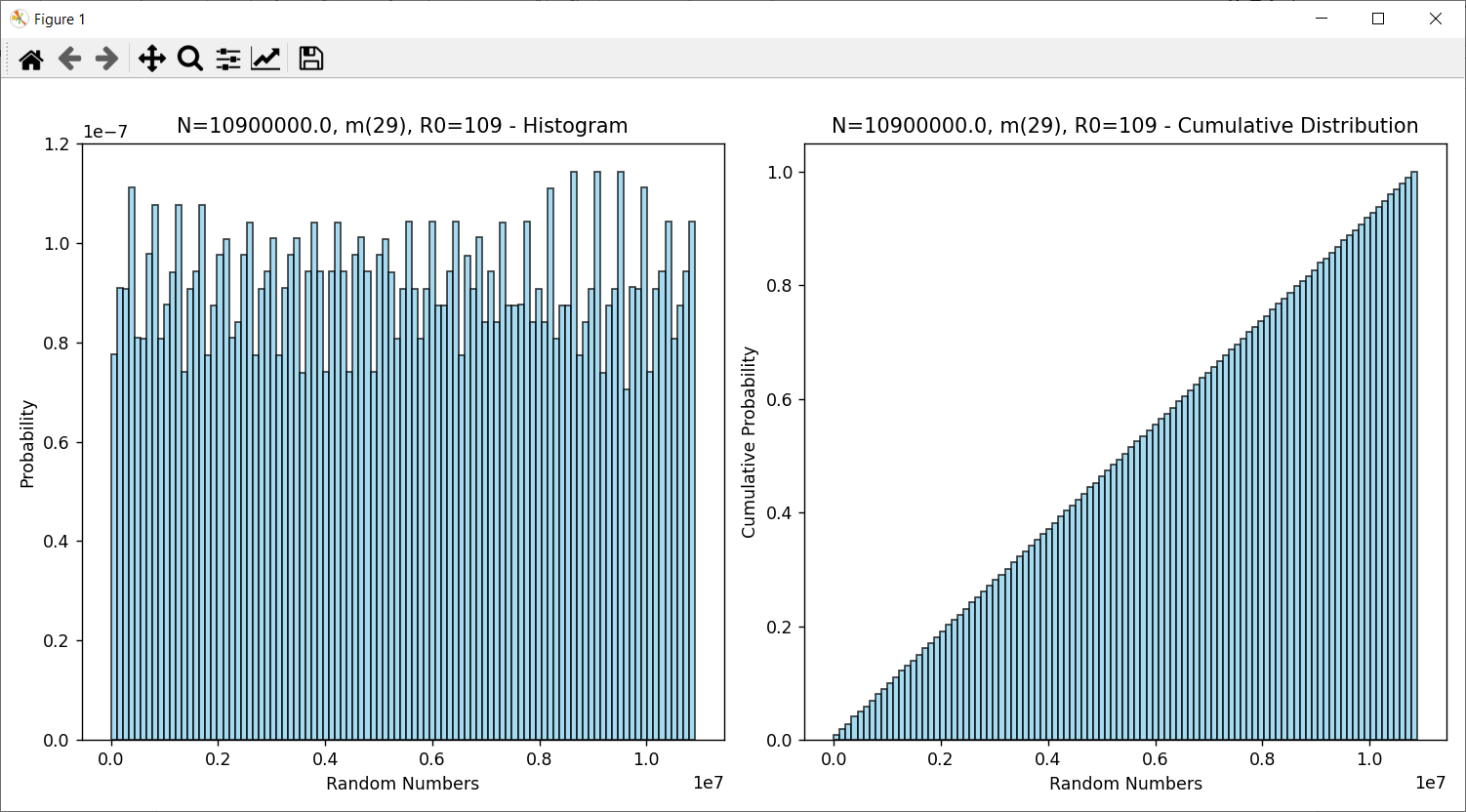
**Рис. 1.2.** Результат для



**Рис. 2.1.** Результат для



**Рис. 2.2.** Результат для



**Рис. 3.1.** Результат для

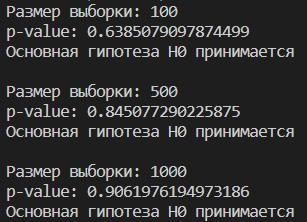


**Рис. 3.2.** Результат для

**Задание 5.2:**

По критерию Колмогорова-Смирнова протестировать выборки случайных чисел объема 100, 500, 1000, сформированных по линейному конгруэнтному методу

**Решение:**



**Рис. 4.** Результат для уровня значимости 0.05

**Задание 6.4:**

1. Произвести расчет нормированной корреляционной функции для интервального сдвига z в пределах от 0 до 50.
2. Построить график нормированной автокорреляционной функции, т. е.

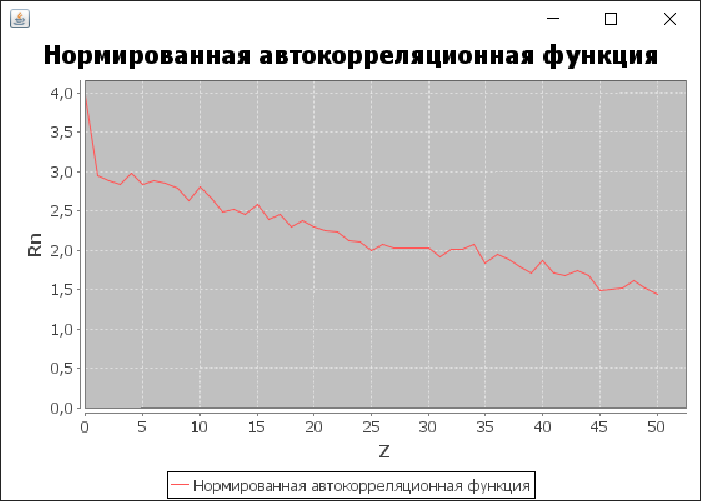
зависимость Rn от z.

1. Произвести расчет нормированной корреляционной функции для объема

выборки N = 740

1. Выполнить первые три пункта задания для анализа ГСЧ в EXCEL(CALC).
2. Выполнить первые три пункта задания для анализа ГСЧ в C#(JAVA).
3. Выполнить первые три пункта задания для анализа ГСЧ в PYTHON.
4. Выполнить первые три пункта задания для анализа ГСЧ в С.
5. Сделать заключение о системе программирования, в которой ГСЧ является наиболее качественным.

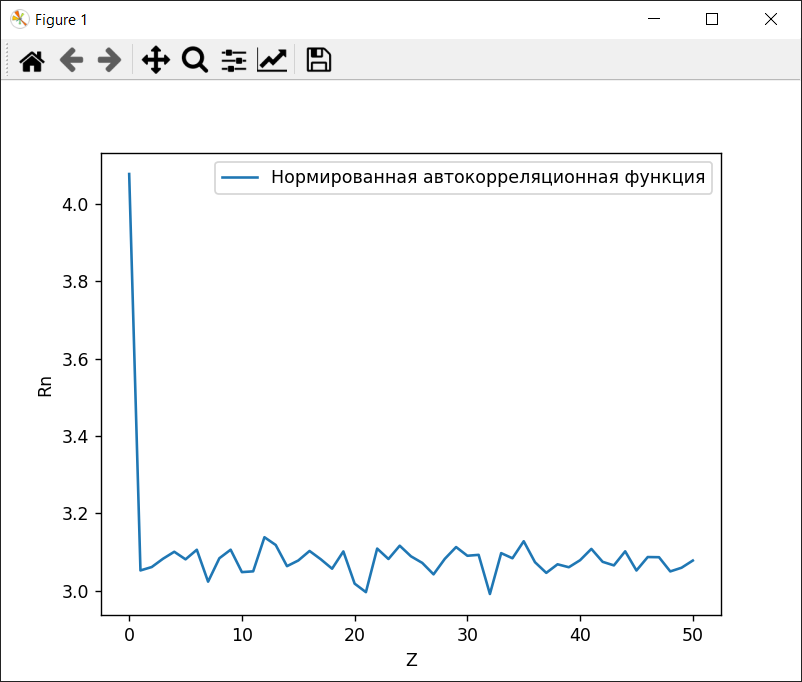
**Решение:**



**Рис. 5.1.** Результат на Java



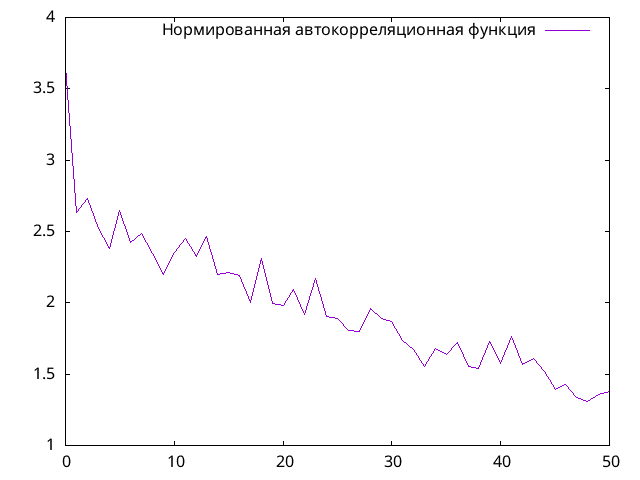
**Рис. 5.2.** Результат на Java



**Рис. 6.1.** Результат на Python



**Рис. 6.2.** Результат на Python



**Рис. 7.1.** Результат на С

C:\Users\Multiname\Downloads\Telegram Desktop\image_2023-12-19_14-55-34.png

**Рис. 7.2.** Результат на С

Python оказался системой программирования, в которой ГСЧ является наиболее качественным.

**Вывод:** в ходе выполнения лабораторной работы были изучены и практически освоены оценки качества генераторов случайных чисел (ГСЧ) в различных системах программирования по заданным теоретическим показателям, с помощью критериев согласия и с помощью нормированной автокорреляционной функции на предмет независимости случайных чисел.

**ПРИЛОЖЕНИЯ**

**Листинг:**

***Task\_4.py:***

import matplotlib.pyplot as plt

import numpy as np

def linear\_congruential\_generator(a, m, c, R0, n):

    random\_numbers = [R0]

    for \_ in range(n - 1):

        next\_number = (a \* random\_numbers[-1] + c) % m

        random\_numbers.append(next\_number)

    return random\_numbers

def period\_length(sequence):

    for i in range(1, len(sequence) // 2 + 1):

        if sequence[:i] == sequence[i:2\*i]:

            return i

    return -1

def histogram\_and\_distribution\_plot(data, bins, title):

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)

    plt.hist(data, bins=bins, density=True, alpha=0.75, color='skyblue', edgecolor='black')

    plt.title(title + ' - Histogram')

    plt.xlabel('Random Numbers')

    plt.ylabel('Probability')

    plt.subplot(1, 2, 2)

    plt.hist(data, bins=bins, density=True, cumulative=True, alpha=0.75, color='skyblue', edgecolor='black')

    plt.title(title + ' - Cumulative Distribution')

    plt.xlabel('Random Numbers')

    plt.ylabel('Cumulative Probability')

    plt.tight\_layout()

    plt.show()

N\_values = [10.7e6, 10.8e6, 10.9e6]

m\_values = [27, 28, 29]

R0\_values = [103, 107, 109]

for N, m, R0 in zip(N\_values, m\_values, R0\_values):

    a = 7\*\*5

    c = 0

    M = N

    random\_sequence = linear\_congruential\_generator(a, M, c, R0, n=int(1e5))

    period = period\_length(random\_sequence)

    print(f"Period for N={N}, m({m}), R0={R0}: {period}")

    bins = np.linspace(0, M, num=100)

    histogram\_and\_distribution\_plot(random\_sequence, bins, f"N={N}, m({m}), R0={R0}")

***Task\_5.py:***

import numpy as np

from scipy.stats import kstest

def generate\_random\_nums(n):

    m = 2\*\*32

    a = 1103515245

    c = 12345

    x = 0

    nums = []

    for \_ in range(n):

        x = (a \* x + c) % m

        nums.append(x)

    return np.array(nums) / m

sizes = [100, 500, 1000]

for size in sizes:

    sample = generate\_random\_nums(size)

    \_, p\_value = kstest(sample, 'uniform')

    print(f"Размер выборки: {size}")

    print(f"p-value: {p\_value}")

    if p\_value > 0.05:

        print("Основная гипотеза H0 принимается")

    else:

        print("Основная гипотеза H0 отвергается")

    print()

***Task\_6.py:***

import numpy as np

import matplotlib.pyplot as plt

data = np.random.rand(1000)

def normalized\_correlation\_function(data, z):

    N = len(data)

    Rn = np.correlate(data, np.roll(data, -z)) / (np.std(data) \* np.std(np.roll(data, -z)) \* N)

    return Rn[0]

z\_values = np.arange(0, 51, 1)

correlation\_values = [normalized\_correlation\_function(data, z) for z in z\_values]

plt.plot(z\_values, correlation\_values, label='Нормированная автокорреляционная функция')

plt.xlabel('Z')

plt.ylabel('Rn')

plt.legend()

plt.show()

N = 740

data = np.random.rand(N)

correlation\_N = normalized\_correlation\_function(data, z\_values)

print(f"Нормированная корреляционная функция при выборке N={N}: {correlation\_N}")

***CorrelationAnalysis.java:***

package org.example;

import org.jfree.chart.ChartFactory;

import org.jfree.chart.ChartPanel;

import org.jfree.chart.JFreeChart;

import org.jfree.chart.plot.PlotOrientation;

import org.jfree.data.xy.XYSeries;

import org.jfree.data.xy.XYSeriesCollection;

import javax.swing.\*;

public class CorrelationAnalysis extends JFrame {

    private static final int N = 100;

    private static double[] data = new double[N];

    private static double normalizedCorrelationFunction(int z) {

        double sum = 0;

        for (int i = 0; i < N - z; ++i) {

            sum += (data[i] \* data[i + z]);

        }

        double mean = 0;

        for (double datum : data) {

            mean += datum;

        }

        mean /= N;

        double variance = 0;

        for (double datum : data) {

            variance += (datum - mean) \* (datum - mean);

        }

        return sum / Math.sqrt(variance \* variance);

    }

    private void plotGraph(XYSeries series) {

        XYSeriesCollection dataset = new XYSeriesCollection(series);

        JFreeChart chart = ChartFactory.createXYLineChart(

                "Нормированная автокорреляционная функция",

                "Z",

                "Rn",

                dataset,

                PlotOrientation.VERTICAL,

                true,

                true,

                false

        );

        ChartPanel chartPanel = new ChartPanel(chart);

        chartPanel.setPreferredSize(new java.awt.Dimension(560, 370));

        setContentPane(chartPanel);

        pack();

        setLocationRelativeTo(null);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setVisible(true);

    }

    public static void main(String[] args) {

        CorrelationAnalysis correlationAnalysis = new CorrelationAnalysis();

        for (int i = 0; i < N; ++i) {

            data[i] = Math.random() \* 1000;

        }

        XYSeries series = new XYSeries("Нормированная автокорреляционная функция");

        for (int z = 0; z <= 50; ++z) {

            double result = correlationAnalysis.normalizedCorrelationFunction(z);

            series.add(z, result);

        }

        correlationAnalysis.plotGraph(series);

        int N\_new = 740;

        System.out.printf("Нормированная корреляционная функция при выборке N = %d: %f\n", N\_new, correlationAnalysis.normalizedCorrelationFunction(0));

    }

}

***main.c:***

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#define N 100

double data[N];

double normalized\_correlation\_function(int z) {

    double sum = 0;

    for (int i = 0; i < N - z; ++i) {

        sum += (data[i] \* data[i + z]);

    }

    double mean = 0;

    for (int i = 0; i < N; ++i) {

        mean += data[i];

    }

    mean /= N;

    double variance = 0;

    for (int i = 0; i < N; ++i) {

        variance += (data[i] - mean) \* (data[i] - mean);

    }

    return sum / sqrt(variance \* variance);

}

int main() {

    for (int i = 0; i < N; ++i) {

        data[i] = rand() % 1000;

    }

    FILE \*gnuplotPipe = popen("gnuplot -persistent", "w");

    fprintf(gnuplotPipe, "set term png\n");

    fprintf(gnuplotPipe, "set output 'correlation\_plot.png'\n");

    fprintf(gnuplotPipe, "plot '-' with lines title 'Нормированная автокорреляционная функция'\n");

    for (int z = 0; z <= 50; ++z) {

        double result = normalized\_correlation\_function(z);

        fprintf(gnuplotPipe, "%d %lf\n", z, result);

    }

    fprintf(gnuplotPipe, "e\n");

    fclose(gnuplotPipe);

    int N\_new = 740;

    printf("Нормированная корреляционная функция при выборке N = %d: %lf\n", N\_new, normalized\_correlation\_function(0));

    return 0;

}