МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра математического моделирования и анализа данных

МИЛЬГУНОВ Евгений Вадимович

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ СТАТИСТИЧЕСКОГО ТЕСТИРОВАНИЯ ВЫХОДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ГЕНЕРАТОРОВ СЛУЧАЙНЫХ ЧИСЕЛ

Дипломная работа

Научный руководитель:
ст. преподаватель,
М.А. Казловский

Допущена к защите
«__»_____2025 г.
Зав. кафедрой математического
моделирования и анализа данных
доктор экономических наук,
профессор В.И. Малюгин

Минск, 2025

ОГЛАВЛЕНИЕ

Термины, обозначения, сокращения ... 4
ВВЕДЕНИЕ ... 8
1. ОБЩАЯ ХАРАКТЕРИСТИКА ПРОБЛЕМЫ АНАЛИЗА ГЕНЕРАТОРОВ СЛУЧАЙНЫХ ЧИСЕЛ ... 9
1.1 Введение в генерацию случайных чисел ... 9
1.2 Требования СТБ 34.101.27 ... 9
1.3 Цели методики МИ.10127.10.03 ... 11
2. МЕТОДИКА СТАТИСТИЧЕСКОГО ТЕСТИРОВАНИЯ ВЫХОДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ГЕНЕРАТОРОВ СЛУЧАЙНЫХ ЧИСЕЛ ... 12
2.1 Тест многомерной дискретной равномерности по непересекающимся отрезкам ... 12
2.2 Тест многомерной дискретной равномерности по пересекающимся отрезкам ... 13
2.3 Тест «пустых ящиков» ... 14
2.4 Тест серий ... 15
2.5 Тест длинных серий ... 16
2.6 Тест аппроксимации энтропии ... 18
2.7 Тест скалярного произведения ... 19
2.8 Спектральный тест ... 21
2.9 Батарея Бернулли ... 22
2.10 Батарея хи-квадрат ... 23
2.11 Тестирование одной последовательности ... 24
2.12 Тестирование нескольких последовательностей ... 25
2.13 Неточности документа методики ... 27
3. ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ТЕСТИРОВАНИЯ ПОСЛЕДОВАТЕЛЬНОСТЕЙ СЛУЧАЙНЫХ ВЕЛИЧИН ... 29
3.1 Основные компоненты программного комплекса ... 29
3.2 Демонстрация работы программного комплекса ... 29
3.3 Результаты тестирования некоторых генераторов случайных чисел 33
3.4 Результаты тестирования выходной последовательности генератора случайных чисел с искусственно добавленными ошибками ... 35
ЗАКЛЮЧЕНИЕ ... 40
Список используемой литературы ... 41
ПРИЛОЖЕНИЕ А ... 42
ПРИЛОЖЕНИЕ Б ... 43
ПРИЛОЖЕНИЕ В ... 44
ПРИЛОЖЕНИЕ Г ... 45

ПРИЛОЖЕНИЕ Д 46
ПРИЛОЖЕНИЕ Е 48
ПРИЛОЖЕНИЕ Ж 49
ПРИЛОЖЕНИЕ И 51
ПРИЛОЖЕНИЕ К 52
ПРИЛОЖЕНИЕ Л 53
ПРИЛОЖЕНИЕ М 54
ПРИЛОЖЕНИЕ Н 57
ПРИЛОЖЕНИЕ П 58

Термины, обозначения, сокращения

БПФ – быстрое преобразование Фурье;

ГСЧ – генератор случайных чисел;

н.р.р. – независимые и равномерно распределённые (случайные величины);

МБ – мегабайт;

МДРН – тест многомерной дискретной равномерности по непересекающимся отрезкам;

МДРП – тест многомерной дискретной равномерности по пересекающимся отрезкам;

ПО – программное обеспечение;

РСЛОС – регистр сдвига с линейной обратной связью;

СКЗИ – средств криптографической защиты информации;

ТНП – тестирование нескольких последовательностей;

ТОП – тестирование одной последовательности;

ФГСЧ – физический генератор случайных чисел;

$\boldsymbol{C_n^m}$– число сочетаний из n по m;

E{ ξ **}** – математическое ожидание случайной величины ξ;

I{$\mathscr{E}$**}** – индикатор наступления события $\mathscr{E}$;

$\boldsymbol{F_L(x)}$ – функция распределения закона L: $F_L(x)$= P $\{\xi \leqslant x\}$, где случайная величина ξ распределена по закону L;

$\boldsymbol{F_L^{-1}(p)}$ – квантиль уровня p ∈ [0, 1] закона L: такое x, что $F_L(x)$= p;

P{$\mathscr{E}$**}** – вероятность наступления события $\mathscr{E}$;

$\boldsymbol{X_m^2}$ – распределение хи-квадрат с m степенями свободы;

Ф(x) – функция стандартного нормального распределения;

[z] – целая часть числа z.

Реферат

Дипломная работа включает 62 страницы, 22 рисунка, 3 таблицы, 9 источников.

Ключевые слова: C++, ГЕНЕРАТОРЫ СЛУЧАЙНЫХ ЧИСЕЛ, СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ, ТЕСТИРОВАНИЕ СЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ, БАТАРЕИ ТЕСТОВ.

Объект исследования – методика тестирования выходных последовательностей генераторов случайных чисел МИ.10127.10.03.

Цель работы – разработка и экспериментальное применение программного комплекса реализующего алгоритмы тестирования методики МИ.10127.10.03.

Методы исследования – изучение литературы, посвящённой батареям тестов выходных последовательностей ГСЧ, проектирование архитектуры и разработка программного комплекса, проверка работоспособности полученного таким образом программного комплекса.

Полученные результаты и их новизна – получен полноценно функционирующий программный комплекс статистического тестирования выходных последовательностей ГСЧ.

Достоверность материалов и результатов дипломной работы. Результаты работы получены реализацией методики МИ.10127.10.03, разработанной НИИ ППМИ, в процессе разработки работа алгоритмов сверялась с эталонной реализацией, достоверность подтверждена реалистичными показателями результатов работы программного комплекса.

Область применения – оценка качества числовых последовательностей, генерируемых ГСЧ.

Рэферат

Дыпломная работа ўключае 62 старонкі, 22 малюнка, 3 табліцы, 9 крыніц.

Ключавыя словы: С++, ГЕНЕРАТАРЫ ВЫПАДКОВЫХ ЛІКАЎ, СРОДКІ КРЫПТАГРАФІЧНАЙ АБАРОНЫ ІНФАРМАЦЫІ, ТЭСТАВАННЕ ВЫПАДКОВЫХ НАСЛЕДЧАСЦЯЎ, БАТАРЭІ ТЭСТАЎ.

Аб'ект даследавання – методыка тэставання выходных паслядоўнасцяў ФГСЧ МИ.10127.10.03.

Мэта работы – распрацоўка і эксперыментальнае прымяненне праграмнага комплексу які рэалізуе алгарытмы тэсціравання методыкі «МИ.10127.10.03».

Метады даследавання – вывучэнне літаратуры, прысвечанай батарэям тэстаў выходных паслядоўнасцяў ГСЧ, праектаванне архітэктуры і распрацоўка праграмнага комплексу, праверка працаздольнасці атрыманага такім чынам праграмнага комплексу.

Атрыманыя вынікі і іх навізна – атрыманы паўнавартасна які функцыянуе праграмны комплекс статыстычнага тэставання выходных паслядоўнасцяў ГСЧ.

Дакладнасць матэрыялаў і вынікаў дыпломнай працы. Вынікі працы атрыманы рэалізацыяй методыкі «МИ.10127.10.03», распрацаванай НДІ ППМИ, у працэсе распрацоўкі праца алгарытмаў спраўджвалася з эталоннай рэалізацыяй, дакладнасць пацверджана рэалістычнымі паказчыкамі вынікаў працы праграмнага комплексу.

Вобласць прымянення – ацэнка якасці лікавых паслядоўнасцяў, якія генерыруюцца ГСЧ.

Summary

The thesis includes 62 pages, 22 images, 3 tables, 9 sources.

Keywords – C++, RANDOM NUMBER GENERATORS, CRYPTOGRAPHIC INFORMATION PROTECTION MEANS, RANDOM SEQUENCE TESTING, TEST BATTERIES.

Object of research – the methodology for testing the output sequences of the physical random number generators «МИ.10127.10.03».

Purpose of the work – development and experimental application of a software package implementing the testing algorithms of the «МИ.10127.10.03»methodology.

Research methods – study of the literature devoted to the RNG output sequence test batteries, architecture design and software package development, testing the operability of the resulting software package.

The results obtained and their novelty – a fully functioning software package for statistical testing of RNG output sequences.

Reliability of materials and results of the thesis. The results of the work were obtained by implementing the «МИ.10127.10.03» methodology developed by the Research Institute of applied problems of mathematics and computer science, during the development process, the algorithms were compared with the reference implementation, reliability is confirmed by realistic indicators of the results of the software.

Scope – assessment of the quality of numerical sequences generated by the random number generators.

ВВЕДЕНИЕ

В нашей стране широко развит рынок СКЗИ – программных продуктов или программно-аппаратных устройств, которые используются для обеспечения конфиденциальности, целостности и подлинности обрабатываемой информации. Согласно законодательству Республики Беларусь, в определенных случаях такие средства подлежат обязательной сертификации – процедуре, направленной на проверку соответствия сертифицируемого средства требованиям некоторого набора стандартов, основным из которых является СТБ 34.101.27-2022 [1] «Информационные технологии и безопасность. Средства криптографической защиты информации. Требования безопасности». Данный стандарт определяет основные требования безопасности, предъявляемые к СКЗИ. В случае, если в СКЗИ используется, то Стандарт предъявляет требования к качеству генерируемой таким генератором числовой последовательности.

Для оценки качества таких последовательностей сотрудниками НИИ прикладных проблем математики и информатики была разработана специализированная методика, имеющая идентификатор МИ.10127.10.03 [2]. Данная методика определяет порядок тестирования выходных последовательностей ФГСЧ, но не предоставляет программный инструментарий, который можно было был использовать для организации такого тестирования.

Цель данной работы – разработка программного комплекса для статистического тестирования выходных последовательностей генераторов случайных чисел с реализацией многопоточного режима работы.

Задачи данной работы:

1. Обзор алгоритмов статистического тестирования, описанных в МИ.10127.10.03.
2. Разработка программного обеспечения с реализацией алгоритмов методики МИ.10127.10.03, многопоточным режимом работы и графическим интерфейсом на языке программирования C++ [3].
3. Тестирование выходных последовательностей различных генераторов псевдослучайных чисел.
4. Тестирование последовательностей физического генератора случайных чисел с искусственно добавленными ошибками.

1. ОБЩАЯ ХАРАКТЕРИСТИКА ПРОБЛЕМЫ АНАЛИЗА ГЕНЕРАТОРОВ СЛУЧАЙНЫХ ЧИСЕЛ

1.1 Введение в генерацию случайных чисел

Генерация случайных чисел играет ключевую роль в современных информационных технологиях, криптографии, моделировании, статистике и многих других областях. Случайные числа используются для формирования криптографических ключей, моделирования процессов в системах массового обслуживания, проведения численных экспериментов, а также для реализации случайных выборок в научных исследованиях.

Одной из основных задач генерации случайных чисел является обеспечение их высокой степени непредсказуемости и соответствие статистическим свойствам случайности. Для достижения этих целей применяются как программные, так и аппаратные генераторы случайных чисел. Программные генераторы, или генераторы псевдослучайных чисел, обеспечивают воспроизводимость последовательностей, но подвержены предсказуемости при недостаточной криптографической защите. В свою очередь, аппаратные генераторы случайных чисел используют физические процессы, такие как тепловой шум или квантовые эффекты, для получения истинной случайности.

С учетом требований к безопасности информационных систем, разработка и тестирование генераторов случайных чисел стали важным направлением исследований. Генераторы должны соответствовать строгим критериям, определённым в национальных и международных стандартах. В Беларуси такие критерии формулируются в стандарте СТБ 34.101.27, который включает требования к свойствам случайных чисел, проверяемым через серии тестов (СЧ.1 – СЧ.8). Эти тесты охватывают основные характеристики случайности и являются обязательными для оценки генераторов, используемых в критически важных приложениях.

1.2 Требования СТБ 34.101.27

Требование СЧ.1 (1–4). Должны быть определены генераторы случайных чисел, которые используются для выработки ключей и других критических объектов. Для каждого генератора должны быть указаны источники случайности и методы обработки данных от источников случайности. Физические источники случайности должны быть включены в список КСК.

Требование СЧ.2 (1–4). Для каждого генератора случайных чисел должна быть проведена оценка энтропии всех его источников случайности. Способ обработки данных от источников случайности должен гарантировать, что

совокупная энтропия данных, использованных для генерации l-битового случайного числа, – не меньше l.

Требование СЧ.3 (1). Если в генераторе случайных чисел отсутствуют физические источники случайности, то должно использоваться не менее двух разных альтернативных источников.

Примечание – Можно использовать два разных системных источника или один системный источник и один источник, основанный на активности оператора.

Требование СЧ.4 (2–4). Генератор случайных чисел должен обязательно использовать хотя бы один физический источник случайности.

Требование СЧ.5 (1–4). Должна быть разработана и корректно реализована процедура статистического тестирования генератора случайных чисел, в котором используются физические источники случайности. Процедура должна быть направлена на выявление сбоев и изменений физических параметров при функционировании физических источников. Процедура должна быть включена в перечень тестов работоспособности СКЗИ. Тестирование генератора должно проводиться перед первым его использованием.

Требование СЧ.6 (1–4). Выходные данные генератора случайных чисел должны являться результатом применения криптографических алгоритмов к данным от источников случайности, возможно дополненным обновляемым внутренним состоянием или предыдущими случайными числами. Применяемые криптографические алгоритмы должны обеспечивать сложные зависимости между выходными данными генератора и данными от каждого из источников случайности. Внутреннее состояние генератора, если оно используется, должно быть отнесено к критическим объектам.

Требование СЧ.7 (1–4). Должна быть разработана и корректно реализована проверка работоспособности физических источников случайности как КСК.

Требование СЧ.8 (3, 4). Всякий раз перед созданием критических объектов с помощью генератора случайных чисел должна проверяться работоспособность его физических источников случайности. При отрицательном результате проверки создание критических объектов должно быть запрещено.

Итого тестирование выходных последовательностей ФГСЧ обязательно соответствии с требованием СЧ.5 стандарта СТБ 34.101.27.

1.3 Цели методики МИ.10127.10.03

В компьютерных системах распространены следующие источники случайности:

- физические источники, использующие процессы в физических устройствах (например, шум в радиоэлектронных приборах);
- системные источники, использующие состояния, процессы и события операционной системы (например, системное время, сетевая активность, прерывания);
- источники, основанные на активности операторов (например, движения мышью, нажатия клавиш).

В методике рассматриваются ГСЧ, в которых используются источники только первого типа.

Целью испытаний является проверка статистической гипотезы о н.р.р. (независимости и равномерном распределении) элементов выходной последовательности ГСЧ. В математической статистике эту гипотезу принято обозначать через H_0. Выполнение H_0 означает равновероятность, независимость и однородность наблюдаемых случайных чисел, отсутствие в них периодичности.

При выполнении H_0 выходные случайные числа генератора максимально трудно предсказать и поэтому их можно использовать для формирования ключей, синхропосылок, других критических или уникальных параметров криптографических алгоритмов и протоколов.

2. МЕТОДИКА СТАТИСТИЧЕСКОГО ТЕСТИРОВАНИЯ ВЫХОДНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ ГЕНЕРАТОРОВ СЛУЧАЙНЫХ ЧИСЕЛ

2.1 Тест многомерной дискретной равномерности по непересекающимся отрезкам

Общие сведения. Тест многомерной дискретной равномерности по непересекающимся отрезкам (МДРН) предназначен для проверки согласия распределения непересекающихся L-фрагментов последовательности X с L-мерным дискретным равномерным распределением. МДРН-тест позволяет выявлять отклонения от L-мерного дискретного равномерного распределения: тест отвергает гипотезу H_0 в случае, когда отклонения эмпирических частот встречаемости фрагментов длины L от их теоретических значений значимо.

Параметр теста: длина фрагмента L.

Обозначение: МДРН[L].

Входные данные: Последовательность X = $(x_1, \ldots, x_n)$.

Шаг 1. Вычислить число фрагментов разбиения: m = [n/L].

Шаг 2. Построить разбиение X на последовательные непересекающиеся L- фрагменты:

$$X_t = (x_{L(t-1)+1}, \ldots, x_{Lt}), \qquad t = \overline{1, m}.$$

Шаг 3. Вычислить частоты:

$$v_u^{(L)} = \sum_{t=1}^{m} \boldsymbol{I}\{X_t = u\}, u \in \{0,1\}^L.$$

Шаг 4. Вычислить статистику:

$$S_{x^2}(m) = \sum_{u\in\{0,1\}^L} \frac{(v_u^{(L)} - m2^{-L})^2}{m2^{-L}}.$$

Шаг 5. Вычислить и возвратить P-значение:

$$P = 1 - F_{x^2_{2^l-1}}(S_{x^2}(m)).$$

Условия применения. Должны выполняться условия:

$$m \geqslant 50,\ m \cdot \frac{1}{2^L} = \frac{n}{L2^L} \geqslant 10.$$

2.2 Тест многомерной дискретной равномерности по пересекающимся отрезкам

Общие сведения. Тест многомерной дискретной равномерности по пересекающимся отрезкам предназначен для проверки согласия распределения пересекающихся L-фрагментов наблюдаемой последовательности с L-мерным дискретным равномерным распределением.

МДРП-тест позволяет обнаруживать отклонения от гипотезы независимости и равновероятности типа рекуррентной, марковской зависимости порядка не больше, чем размерность вектора L, отклонения от L-мерного дискретного равномерного распределения.

Параметр теста: длина фрагмента L.

Обозначение: МДРП[L].

Входные данные: Последовательность $X = (x_1, \ldots, x_n)$.

Шаг 1. Построить разбиение X на пересекающиеся L-фрагменты:

$$X_1^{(L)} = (x_1, \ldots, x_L), X_2^{(L)} = (x_2, \ldots, x_{L+1}), \ldots,$$
$$X_{n-L+2}^{(L)} = (x_{n-L+2}, \ldots, x_1), \ \ldots, X_n^{(L)} = (x_n, \ldots, x_{L-1}).$$

Шаг 2. Для $u \in \{0,1\}^L$ вычислить частоты по всем фрагментам:

$$v_u^{(L)} = \sum_{t=1}^{n} \boldsymbol{I}\{X_t = u\}, u \in \{0,1\}^L.$$

Шаг 3. На основании частот $v_u^{(L)}$ вычислить статистику $\gamma_L(n)$:

$$\gamma_L(n) = \sum_{u\in\{0,1\}^L} \frac{(v_u^{(L)} - n2^{-L})^2}{n2^{-L}}.$$

Шаг 4. Построить разбиение X на пересекающиеся $(L-1)$-фрагменты:

$$X_1^{(L-1)} = (x_1, \ldots, x_{L-1}), X_2^{(L-1)} = (x_2, \ldots, x_L), \ldots,$$
$$X_{n-L+2}^{(L-1)} = (x_{n-L+2}, \ldots, x_1), \ \ldots, X_n^{(L-1)} = (x_n, \ldots, x_{L-2}).$$

Шаг 5. Вычислить частоты встречаемости векторов $v \in \{0,1\}^{L-1}$:

$$v_u^{(L-1)} = \sum_{t=1}^{n} \boldsymbol{I}\{X_t^{(L-1)} = u\}, u \in \{0,1\}^L.$$

Шаг 6. Вычислить статистику $\gamma_{L-1}(n)$:

$$\gamma_{L-1}(n) = \sum_{u\in\{0,1\}^{L-1}} \frac{(v_u^{(L-1)} - n2^{-(L-1)})^2}{n2^{-(L-1)}}.$$

Шаг 7. Вычислить статистику теста:

$$S_{\text{МДРП}} = \gamma_L(n) - \gamma_{L-1}(n).$$

Шаг 8. Вычислить и возвратить P-значение:

$$P = 1 - F_{x^2_{2^L-2^{L-1}}}(S_{\text{МДРП}}).$$

Условия применения. Должно выполняться условие:

$$\text{n} \geqslant 20 \cdot 2^L (\ \text{L} \leqslant log_2 \frac{n}{20}\ ).$$

Рекомендуется выбирать: $2 \leqslant \text{L} \leqslant 23$, причем при L = 20 необходимая длина последовательности $\text{n} = 20 \cdot 2^{20} = 20971520$ битов.

2.3 Тест «пустых ящиков»

Общие сведения. Применение статистических тестов, использующих частоты векторов из $\{0,1\}^L$, ограничено небольшими значениями L. С увеличением L информативной статистикой становится число векторов, не встретившихся в тестируемой последовательности ни разу (или же встретившихся более одного раза). Тестом, основанным на данной статистике, является тест «пустых ящиков». Тест «пустых ящиков» позволяет обнаруживать отклонения от L-мерного дискретного равномерного распределения.

Параметр теста: длина фрагмента L.

Обозначение: КПЯ[L].

Входные данные: Последовательность $X = (x_1, \ldots, x_n)$.

Шаг 1. Вычислить число фрагментов m = [n/L].

Шаг 2. Построить разбиение X на последовательные непересекающиеся L-фрагменты: $\text{X} = (X_1, \ldots, X_m)$, $X_t = (X_{L(t-1)+1}, \ldots, x_{Lt}) \in \{0,1\}^L$, $\text{t} = \overline{1,m}$.

Шаг 3. Для $v \in \{0,1\}^L$ вычислить частоты:

$$v_u^{(L)} = \sum_{t=1}^{m} \boldsymbol{I}\{X_t = u\}, u \in \{0,1\}^L.$$

Шаг 4. Вычислить статистику теста – число векторов, встретившихся более одного раза (число «конфликтов»):

$$S_{\text{ЕВ}} = \sum_{v\in\{0,1\}^L} (v_u^{(L)} - \mathbf{1})\boldsymbol{I}\{v_u^{(L)} > 1\} = m - \left(2^L - \sum_{v\in\{0,1\}^L} \boldsymbol{I}\left\{v_u^{(L)} = 0\right\}\right)$$
$$= m - (2^L - \mu_0),$$

где $\mu_0 = \mu_0(\mathrm{m}, 2^L)$ – число «пустых ящиков»: векторов из $\{0,1\}^L$, не встретившихся в X ни разу.

Шаг 5. Вычислить вспомогательные параметры:

$$\lambda = \frac{m}{2^L},\ \lambda_{\text{П}} = \frac{m^2}{2\cdot 2^L},\ \mu = 2^L e^{-\lambda},\ \sigma^2 = 2^L e^{-\lambda}(1 - (1+\lambda)e^{-\lambda}).$$

Шаг 6. Если $\lambda \leqslant 1/32$, то P-значение вычислить по формуле:

$$\mathrm{P} = 1 - F_{x^2_{2S_{EB}}}(2\lambda_{\text{П}}).$$

Шаг 7. Если $1/32 < \lambda < 5$, то P-значение вычислить по формуле:

$$\mathrm{P} = \Phi\left(-\frac{\mu_0 - \mu}{\sigma}\right).$$

Шаг 8. Возвратить P-значение.

Условия применения. Тест «пустых ящиков» следует применять для больших L в тех случаях, когда применить тесты типа хи-квадрат нельзя.

Размерность фрагментов L и число фрагментов $\mathrm{m} = \mathrm{n}/\mathrm{L}$ в X должны согласовано выбираться в соответствии с одной из указанных выше асимптотик (см. шаги 5 и 6 алгоритма).

В случае $\lambda \geqslant 5$ тест не применяется: рекомендуется использовать критерий хи-квадрат, описание которого приведено в разделе 2.10.

При $\lambda < 1/32$ рекомендуется использовать $\mathrm{L} \geq 19$ и $\mathrm{m} = 2^{L/2+3}$ векторов.

2.4 Тест серий

Общие сведения. Цель теста – проверить, соответствует ли число серий из нулей и единиц различной длины в наблюдаемой последовательности теоретически ожидаемому числу для равномерно распределенных случайных последовательностей.

Тест серий позволяет выявлять последовательности, которые с вероятностью больше 0,5 сохраняют свое предыдущее значение.

Обозначение: КС.

Входные данные: Последовательность $\mathrm{X} = (x_1, \ldots, x_n)$.

Фрагмент $\mathrm{X} = (x_t, \ldots, x_{t+l-1})$ последовательности X называется серией длины l, если $x_t = x_{t+1} = \cdots = x_{t+l-1}$, но $x_{t-1} \neq x_t$ (или $\mathrm{t} = 1$) и $x_{t+l-1} \neq x_{t+l}$ (или $\mathrm{t} + \mathrm{l} - 1 = \mathrm{n}$).

Шаг 1. Вычислить теоретические частоты серий:

μ_i = (n − i + 3) / 2^{i+2}, i = k, где k определяется из условия: $\mu_k \geqslant 5$ и $\mu_{k+1} < 5$.

Шаг 2. Для i = 1, . . . , k вычислить частоты $v_i^{(0)}$ и $v_i^{(1)}$. Здесь $v_i^{(b)}$ – частота серий из символов b длины i в последовательности X.

Шаг 3. Вычислить значение статистики теста S_{runs}:

$$S_{runs} = \sum_{i=1}^{k} \left(\frac{(v_i^{(0)} - \mu_i)^2}{\mu_i} \right) + \sum_{i=1}^{k} \left(\frac{(v_i^{(1)} - \mu_i)^2}{\mu_i} \right).$$

Шаг 4. Вычислить и возвратить P-значение:

$$P = 1 - F_{x^2_{2k-2}}(S_{runs}).$$

Условия применения теста. Рекомендуется использовать n $\geqslant$ 20000 и k $\geqslant$ 6.

2.5 Тест длинных серий

Общие сведения. Тест длинных серий в отличии от теста серий проверяет, соответствует ли максимальная длина серии из единиц (или из нулей) в L-битном фрагменте наблюдаемой последовательности теоретически ожидаемому значению.

Тест длинных серий позволяет выявлять такие же зависимости, как и в тесте серий (см. п. 2.4).

Параметр теста: длина фрагмента L ∈ {8, 128, 512, 1000, 10000}.

Обозначение: КДС[L].

Входные данные: Последовательность X = (x_1, . . . , x_n).

Шаг 1. Разбить X на m = [n/L] фрагментов X_i по L бит:

$$X_t = (X_{(i-1)L+1}, \ldots, x_{(i-1)L+L}) \in \{0,1\}^L, \; i = \overline{1, m}.$$

Шаг 2. По L, используя таблицу 2, определить разбиение множества длин серий на подмножества (классы) $D_0, \ldots, D_K$.

Шаг 3. Для i = 1, . . . , m найти длину b_i максимальной серии из единиц в фрагменте X_i .

Шаг 4. Для j = 1, . . . , K:

1) используя таблицу 3, определить теоретические вероятности попадания статистик b_i в класс D_j :

$$\pi_j = \mathbf{P}\{b_i \in D_j\};$$

2) вычислить частоты попадания статистик b_i в класс D_j :

$$v_j = \sum_{i=1}^{K} \boldsymbol{I}\{b_i \in D_j\}.$$

Шаг 5. Вычислить статистику теста:

$$S_{long} = \sum_{j=0}^{K} \left(\frac{(v_j - m\pi_j)^2}{m\pi_j}\right).$$

Шаг 6. Вычислить и возвратить P-значение теста:

$$\mathrm{P} = 1 - F_{x_K^2}(S_{long}).$$

Условия применения. Должно выполняться условие: m $min_{0\leqslant \mathrm{i} \leqslant \mathrm{K}}\, \pi_j \geqslant 10$.

Откуда следует, что минимальная длина n последовательности X для различных значений параметра L должна удовлетворять значениям из таблицы 2.1.

Таблица 2.1 – Минимальные значения n теста длинных серий

L	8	128	512	1000	10000
n	427	12464	50688	100000	1490000

Примечание. Если верна H_0 и в фрагменте X_i встречается r единиц и L − r нулей, то теоретические вероятности равны:

$$\mathbf{P}\{b_i \leqslant q\} = \sum_{r=0}^{L} C_r^L\, \mathbf{P}\{b_i \leqslant q | r\} \frac{1}{2^L}, \quad \mathbf{P}\{b_i \leqslant q | r\} = \frac{1}{C_L^r} \sum_{j=0}^{U} (-1)^j \frac{1}{2^L} C_{L-r+1}^{j} C_{L-j(q+1)}^{L-r},$$

где $0 \leqslant \mathrm{q} \leqslant \mathrm{L}$, $U = \min(\mathrm{L} - \mathrm{r} + 1, [\mathrm{r}/(\mathrm{q} + 1)])$, $0 \leqslant \mathrm{r} \leqslant \mathrm{L}$.

При больших значениях L вычисление вероятностей π_i по данным формулам требует больших вычислительных и временных ресурсов, поэтому в методике предлагается значения длин фрагментов L выбирать из множества $\Lambda = \{8, 128, 512, 1000, 10000\}$, для которого сделаны предвычисления. В таблице 2 представлены классы D_i для различных значений $\mathrm{L} \in \Lambda$, а в таблице 3 – соответствующие им вероятности π_i.

Таблица 2.2 – Разбиение множества значений статистики b_i на классы

L	8	128	512	1000	10000
K	3	5	5	5	6
D_0	$\leqslant 1$	$\leqslant 4$	$\leqslant 6$	$\leqslant 7$	$\leqslant 10$
D_1	2	5	7	8	11
D_2	3	6	8	9	12
D_3	$\geqslant 4$	7	9	10	13
D_4	—	8	10	11	14
D_5	—	$\geqslant 9$	$\geqslant 11$	$\geqslant 12$	15
D_6	—	—	—	—	$\geqslant 16$

Таблица 2.3 – Вероятности попадания статистики b_i в классы

L	8	128	512	1000	10000
K	3	5	5	5	6
π_0	0.2148	0.1174	0.1170	0.1307	0.0882
π_1	0.3672	0.2430	0.2460	0.2437	0.2092
π_2	0.2305	0.2493	0.2523	0.2452	0.2483
π_3	0.1875	0.1752	0.1755	0.1714	0.1933
π_4	—	0.1027	0.1015	0.1002	0.1208
π_5	—	0.1124	0.1077	0.1088	0.0675
π_6	—	—	—	—	0.0727

2.6 Тест аппроксимации энтропии

Общие сведения. Тест аппроксимации энтропии основан на статистических оценках энтропии случайных фрагментов размерностей L и L − 1.

Тест позволяет обнаруживать отклонения от свойств бинарных равномерно распределенных случайных последовательностей, связанные со значимыми изменениями количества информации при переходе от фрагментов размерности L − 1 к фрагментам размерности L.

Параметр теста: длина фрагмента L.

Обозначение: КАЭ[L].

Входные данные: Последовательность X = $(x_1, \ldots, x_n)$.

Шаг 1. Для l ∈ {L − 1, L} построить разбиения X на пересекающиеся l-фрагменты: $X_t^{(l)} = (x_t, x_{t+1}, \ldots, x_{t+l-1})$, t = $\overline{1, n + 1 - l}$.

Шаг 2. Для l ∈ {L − 1, L} вычислить относительные частоты $\{\pi_t^{(l)}\}$ встречаемости всевозможных векторов $v \in \{0,1\}^L$:

$$\pi_t^{(l)} = \frac{1}{n+1-l} \sum_{t=1}^{n+1-l} \boldsymbol{I}\{X_t^{(l)} = v\}.$$

Шаг 3. Для l ∈ {L − 1, L} вычислить выборочные энтропии:

$$F^{(l)} = -\sum_{v \in \{0,1\}^L} \pi_v^{(l)} \ln \pi_v^{(l)}.$$

Шаг 4. Вычислить приращение энтропии:

$$ApEn(L) = F^{(L)} - F^{(L-1)}.$$

Шаг 5. Вычислить статистику теста $S_{AE}(n)$:

$$S_{AE}(n) = 2n|ln2 - ApEn(L)|.$$

Шаг 6. Вычислить и возвратить P-значение теста:

$$\text{P} = 1 - F_{x^2_{2^L - 2^{L-1}}}(S_{AE}(n)).$$

Условия применения. Параметр L рекомендуется выбирать максимально возможным, удовлетворяющим условию $n/2^L \geqslant 100$.

2.7 Тест скалярного произведения

Общие сведения. Тест основан на экстремальной статистике скалярного произведения. Тест позволяет обнаруживать отклонения от гипотезы независимости и равновероятности.

Параметры теста: длина подфрагмента m, число подфрагментов K ⩾ 2. По параметрам m и K определяется длина фрагмента L = m · K.

Обозначение: КСКП[m, K].

Входные данные: Последовательность $X = (x_1, \ldots, x_n)$.

Шаг 1. Разбить X на $M = [n/L]$ фрагментов длины $L = m \cdot K$, а каждый из этих фрагментов в свою очередь разбить на K подфрагментов длины m:

$$X^{(l)} = \left(X_1^{(l)}, X_2^{(l)}, \ldots, X_K^{(l)}\right), l = \overline{1, M},$$

$$X_i^{(l)} = \left(x_{(l-1)L+(i-1)m+1}, \ldots, x_{(l-1)L+im}\right), i = \overline{1, K}.$$

Шаг 2. По каждому фрагменту $X^{(l)}$ ($l = \overline{1,M}$) вычислить K − 1 статистик $Y_2^{(l)}, \ldots, Y_K^{(l)}$: $Y_o^{(l)} = (X_1^{(l)}, X_o^{(l)}) = \sum_{k=1}^{m} x_{(l-1)L+k} * x_{(l-1)L+(j-1)m+k}, j = \overline{2,K}$.

Шаг 3. Для каждого фрагмента $X^{(l)}$ $l = \overline{1,M}$ по статистикам $Y_2^{(l)}, \ldots, Y_K^{(l)}$ определить экстремальную статистику $Y_{max}^{(l)}$:

$$Y_{max}^{(l)} = \max\{Y_2^{(l)}, \ldots, Y_K^{(l)}\}.$$

Статистика Y_j характеризует «степень похожести» 1-го и j-го фрагментов.

Шаг 4. По набору статистик $\{Y_{max}^{(l)} : l = 1, M\}$ вычислить эмпирические частоты:

$$f_k = \sum_{l=1}^{M} \boldsymbol{I}\left\{Y_{max}^{(l)} = k\right\}.$$

Шаг 5. Вычислить теоретические частоты:

$$q_0 = \boldsymbol{P}\{Y_{max} = 0\} = 2^{-m} \sum_{l=0}^{m} 2^{-l(K-l)}\, C_m^l = 2^{-m}(1 + 2^{1-K})^m,$$

$$q_k = \boldsymbol{P}\{Y_{max} = k\} = 2^{-m} \sum_{l=0}^{m} C_m^l \Big(\sum_{l=0}^{k} 2^{-l} C_l^y\Big)^{K-1} - 2^{-m} \sum_{l=0}^{m} C_m^l \Big(\sum_{l=0}^{k-1} 2^{-l} C_l^y\Big)^{K-1},$$

$k > 0$.

Шаг 6. Вычислить статистику теста S_{x^2}:

$$S_{x^2} = \sum_{k=0}^{m} \Big(\frac{(f_k - Mq_k)^2}{Mq_k}\Big).$$

Шаг 7. Вычислить и возвратить P-значение теста:

$$\mathrm{P} = 1 - F_{x_m^2}(S_{x^2}).$$

Условия применения. При выборе параметров m и K требуется обеспечить условия:

1) $m \in \{2, 3, \ldots, 8\}$;

2) $K \in \{2, 3, \ldots, 11\}$;

3) $\frac{K}{q_{min}} \leqslant \frac{n}{10m}$, где $q_{min} = min_k q_k$.

2.8 Спектральный тест

Общие сведения. Спектральный тест предназначен для выявления периодичности в бинарных случайных последовательностях.

Параметры теста: нет.

Обозначение: СК-БПФ.

Входные данные. Последовательность X = (x_1, . . . , x_n).

Шаг 1. По X сформировать последовательность Y, заменяя x_t = 0 на y_t = −1, а x_t = 1 на y_t = 1:

$$Y = (y_1, y_2, \ldots, y_n),\ y_t = 2x_t - 1,\ t = \overline{1, n}.$$

Шаг 2. К Y применить алгоритм быстрого преобразования Фурье (БПФ) и сформировать комплекснозначную последовательность S = (s_0, s_1, . . . , s_{n-1}):

$$s_t = \sum_{k=1}^{n} y_k \exp\left(\frac{2\pi i(k-1)t}{n}\right),\ t = \overline{0, n-1},$$

где

$$\exp\left(\frac{2\pi ikt}{n}\right) = \cos\left(\frac{2\pi kt}{n}\right) + isin\left(\frac{2\pi kt}{n}\right),$$

и i =$\sqrt{-1}$ – мнимая единица.

Шаг 3. По S сформировать последовательность {|s_t |}:

$$|s_t| = \sqrt{(s_t^{\Re})^2 + (s_t^{\Im})^2},$$

где $\{s_t^{\Re}\}$ и $\{s_t^{\Im}\}$ – действительная и мнимая части последовательности S (в силу симметричности суммируем половину членов последовательности).

Шаг 4. Определить порог h = $\sqrt{3n}$ и построить последовательность бинарных случайных величин – индикаторов превышения значения |s_t | порога h:

$$u_t = I\{|s_t| < h\},\ t = \overline{0, n/2 - 1}.$$

При выполнении H_0 95% значений |s_t| должно лежать ниже порогового значения $h = \sqrt{3n}$. При этом величины u_t имеют распределение Бернулли:

$$\mathcal{L}\{u_t\} = B_i\ (1, p),\ \ p = 0.95,\ \ t = \overline{0, n/2 - 1}.$$

Шаг 5. Вычислить значение статистики S_{SC}:

$$S_{SC} = \sum_{t=0}^{n/2-1} u_t$$

Шаг 6. Вычислить математическое ожидание и дисперсию:

$$\mu = \frac{n}{2} p,\ \sigma^2 = \frac{n}{2} p(1-p)$$

Шаг 7. Вычислить и возвратить P-значение теста:

$$\text{P} = 2\,(\,1 - \Phi\,(\,|\,\frac{S_{SC} - \mu}{\sigma}\,|\,)).$$

Условия применения. Требуется выполнение условия: $\frac{n}{2} \cdot 0.95 \cdot 0.05 \geqslant 9$, то есть $n \geqslant 400$. Рекомендуется выбирать $n \geqslant 1000$ битов.

2.9 Батарея Бернулли

Общие сведения.

При использовании батареи Бернулли по набору входных P-значений фрагментов для каждого теста строится новая случайная бинарная последовательность, элементы которой равны 1 при принятии H_0 и равны 0 при отклонении H_0 на уровне значимости теста α. При выполнении H_0 новая последовательность соответствует реализации схемы независимых испытаний Бернулли. Длина новой последовательности совпадает с числом фрагментов тестирования. Частота принятия H_0 по всем фрагментам определяет еще одну статистику, которая и является статистикой интегрального P-значения. При выполнении H_0 данная статистика имеет распределение Бернулли.

Параметры: уровень значимости тестов α.

Обозначение: Б1[α].

Входные данные: Набор P-значений (P_{ij}), i = $\overline{1,q}$, j = $\overline{1,m}$ (q – число тестов; m – число фрагментов).

Шаг 1. Для i = 1, . . . , q и j = 1, . . . , m по набору P-значений построить индикаторы:

$$v_{ij} = \mathbf{I}\{P_{ij} \geqslant \alpha\}\ .$$

Шаг 2. Для i = 1, . . . , q вычислить частоту принятия H_0:

$$\overline{v_i} = \sum_{j=1}^{m} v_{ij}.$$

Шаг 3. Для i = 1, . . . , q вычислить интегральные P-значения тестов:

$$P_i^1 = 1 - \Phi\left(-\frac{\frac{\overline{v_i}}{m} - (1-\alpha)}{\sqrt{\frac{\alpha(1-\alpha)}{m}}}\right).$$

Шаг 4. Возвратить (P_1^1 , . . . , P_q^1).

Условия применения. Должно выполняться условие: m $\geqslant 1/\alpha$. Рекомендуемые значения: β = 0.0001; m = 104 для выборки 1 МБ и m = 200 для выборки 10 МБ.

2.10 Батарея хи-квадрат

Общие сведения. При использовании батареи хи-квадрат к входному набору P-значений фрагментов для каждого теста в отдельности применяется критерий согласия хи-квадрат, при этом набор P-значений интерпретируется как новые статистические наблюдения. Длина новой последовательности совпадает с числом фрагментов тестирования. По значению статистики хи-квадрат строится интегральное P-значение теста.

Параметры: k – число интервалов группировки для критерия хи-квадрат.

Обозначение: Б2[k].

Входные данные. Набор P-значений (P_{ij}), i = $\overline{1,q}$, j = $\overline{1,m}$ (q – число тестов; m – число фрагментов).

Шаг 1. Разбить отрезок [0; 1] на k равных интервалов:

I_j= [(j − 1)/k; j/k), j = $\overline{1,k\ -\ 1}$, I_k= [(k − 1)/k; 1].

Шаг 2. Для каждого теста K_i (i = 1, . . . , q) вычислить частоты попадания P- значений данного теста в интервалы I_j :

$$\nu_{ij}=\sum_{t=1}^{m} \boldsymbol{I}\{P_{ij} \in I_j\},\ j = 1, \ldots, k.$$

Теоретические вероятности попадания в каждый из интервалов I_j равняются 1/k.

Шаг 3. Для i = 1, . . . , q вычислить статистику критерия хи-квадрат i-го теста:

$$S_{x^2}^{(i)} = \sum_{j=1}^{m} \frac{(v_{ij} - m/k)^2}{m/k}.$$

где m/k – теоретическая частота попадания наблюдений в j-й интервал.

Шаг 4. Для i = 1, . . . , q вычислить интегральные P-значения тестов:

$$P_i^2 = 1 - F_{x_{m-1}^2}(S_{x^2}^{(i)}).$$

Шаг 5. Возвратить (P_1^2 , . . . , P_q^2). Условия применения. Должны выполняться условия: m ⩾ $1/\alpha$, k ⩾ 10, m/k ⩾ 10.

Рекомендуемые значения: = 0.0001; m= 104, k = 10 для выборки 1 МБ; m = 200, k = 10 для выборки 10 МБ.

2.11 Тестирование одной последовательности

Общие сведения. Методика принятия H_0 основана на принципе гарантированного результата: H_0 принимается, если все интегральные P-значения каждой из батарей свидетельствуют в ее пользу.

Параметры: Набор тестов $\mathscr{B} = \{K_1, K_2, \ldots, K_q\}$, где q – число тестов в батарее; число фрагментов m; число интервалов группировки k; уровень значимости тестов α; уровень значимости батарей β.

Обозначение: ТОП[$\mathscr{B}, m, k, \alpha, \beta$].

Входные данные. Последовательность X = ($x_1, \ldots, x_N$).

Шаг 1. Разбить X на m фрагментов по n = [N/m] битов:

$X^{(j)}$= ($x_{(j-1)n+1}, x_{(j-1)n+2}, \ldots, x_{jn}$), j = 1, . . . , m.

При таком разбиении последние биты X могут игнорироваться.

Шаг 2. Для i = 1, . . . , q и j = 1, . . . , m по батарее тестов $\mathscr{B}$ вычислить наборы P-значений:

$$P_{ij} = K_i(X^{(j)}).$$

Шаг 3. Вычислить интегральные P -значения батареи Бернулли:

$$(P_1^1, \ldots, P_q^1) = Б1[\alpha](P_{11}, \ldots, P_{1m}, P_{21}, \ldots, P_{2m}, \ldots, P_{q1}, \ldots, P_{qm}).$$

Шаг 4. Вычислить интегральные P -значения батареи хи-квадрат:

$$(P_1^2, \ldots, P_q^2) = = Б2[k](P_{11}, \ldots, P_{1m}, P_{21}, \ldots, P_{2m}, \ldots, P_{q1}, \ldots, P_{qm}).$$

Шаг 5. Найти минимальное P-значение среди интегральных P -значений обеих батарей Б1 и Б2:

$$P_{min} = \min\{P_1^1, \ldots, P_q^1, P_1^2, \ldots, P_q^2\}.$$

Шаг 6. Возвратить I{$P_{min} < \beta$ }: 0 означает принятие H_0, 1 – отклонение.

Шаг 7. Если блок «Тестирование одной последовательности» используется в составе блока «Тестирование нескольких последовательности», то дополнительно возвратить набор ($P_1^1, \ldots, P_q^1, P_1^2, \ldots, P_q^2$).

Схема тестирования одной последовательности представлена на рисунке 2.1.

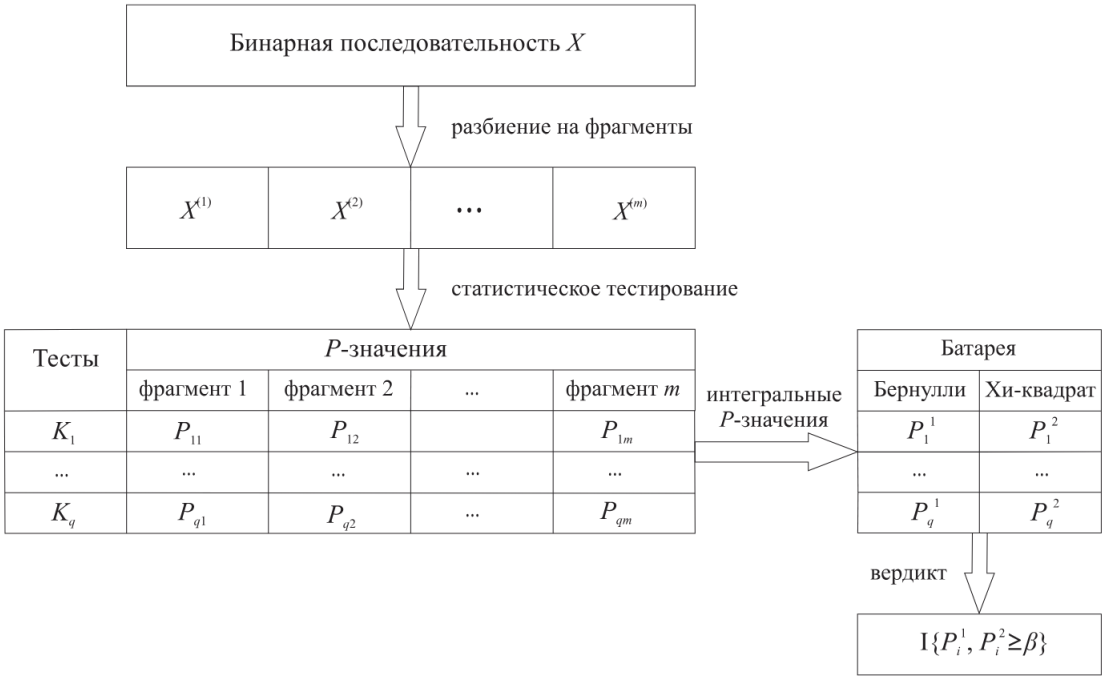


Рисунок 2.1 – Схема тестирования одной последовательности

2.12 Тестирование нескольких последовательностей

Общие сведения. Пусть тестируется не одна, а несколько последовательностей, и к каждой из них применяется блок ТОП с фиксированным уровнем значимости β. Тогда с ростом числа последовательностей вероятность браковки хотя бы одной из них будет увеличиваться даже при справедливости H_0. Чтобы избежать чересчур жестких вердиктов, процедура принятия решения модифицируется. Во-первых, анализируется уровень браковки каждого из тестов по-отдельности. Гипотеза H_0 отвергается, только если некоторый тест явно свидетельствует против нее. Во-вторых, при обработке уровней браковки отдельных тестов используется дополнительный уровень значимости. Этот уровень уменьшается с ростом числа последовательностей, и, таким образом, требования к уровням браковки ослабляются.

Параметры: Набор тестов $\mathscr{B} = \{K_1, K_2, \ldots, K_q\}$, где q – число тестов в батарее; число фрагментов m; число интервалов группировки k; уровень значимости тестов α; уровень значимости батареи β; уровень значимости прохождения тестов в составе батарей γ.

Обозначение: ТНП[$\mathscr{B}$, m, k, α, β, γ].

Входные данные. Последовательности X[1], X[2], . . . , X[M] длины N каждая (M $\geqslant$ 2).

Шаг 1. Для j = 1, . . . , M вычислить:

$$(P_1^1[j], \ldots, P_q^1[j], P_1^2[j], \ldots, P_q^2[j]) = \text{ТОП}[\mathscr{B}, m, k, \alpha, \beta](X[j]).$$

Полученный набор P-значений ($P_i^{1,2}$) называется статистическим портретом.

Шаг 2. Для b = 1, 2, i = 1, . . . , q и j = 1, . . . , M вычислить результаты прохождения теста K_i в составе батарей Б1 и Б2:

$$\Delta_i^b = \text{I}\{P_i^b[j] \geqslant \beta\}.$$

Шаг 3. Для b = 1, 2 и i = 1, . . . , q вычислить относительные частоты прохождения теста K_i в составе батарей Б1 и Б2:

$$\text{v}_i^b = \frac{1}{M}\sum_{j=1}^{M}\Delta_i^b.$$

Шаг 4. По M и γ построить доверительный интервал $[\Delta_{\min}, \Delta_{\max}]$ для относительных частот прохождения тестов:

$$\Delta_{\min} = 1 - \gamma - 3\sqrt{\frac{\gamma(1-\gamma)}{M}},\ \Delta_{\max} = 1 - \gamma + 3\sqrt{\frac{\gamma(1-\gamma)}{M}}.$$

Если полученное значение $\Delta_{\min} < 0$, то его значение следует переопределить: $\Delta_{\min} = 0$. Если $\Delta_{\max} > 1$, то его также следует переопределить: $\Delta_{\max} = 1$.

Например, при M = 50 и γ = 0.01 будет получен доверительный интервал [0.948, 1]; при M = 50 и γ = 0.001 – интервал [0.986, 1].

Шаг 5. Найти минимальное значение относительных частот прохождения тестов для обеих батарей:

$$\nu_{\min} = \min_{\text{b,i}}\{\text{v}_i^b\}$$

Минимум берется по $\text{b} \in \{1, 2\}$ и $\text{i} \in \{1, \ldots, \text{q}\}$.

Шаг 6. Возвратить $\text{I}\{\text{v}_{\min} \notin [\Delta_{\min},\Delta_{\max}]\}$: 0 означает принятие H_0, 1 – отклонение.

Схема тестирования нескольких последовательностей представлена на рисунке 2.2.

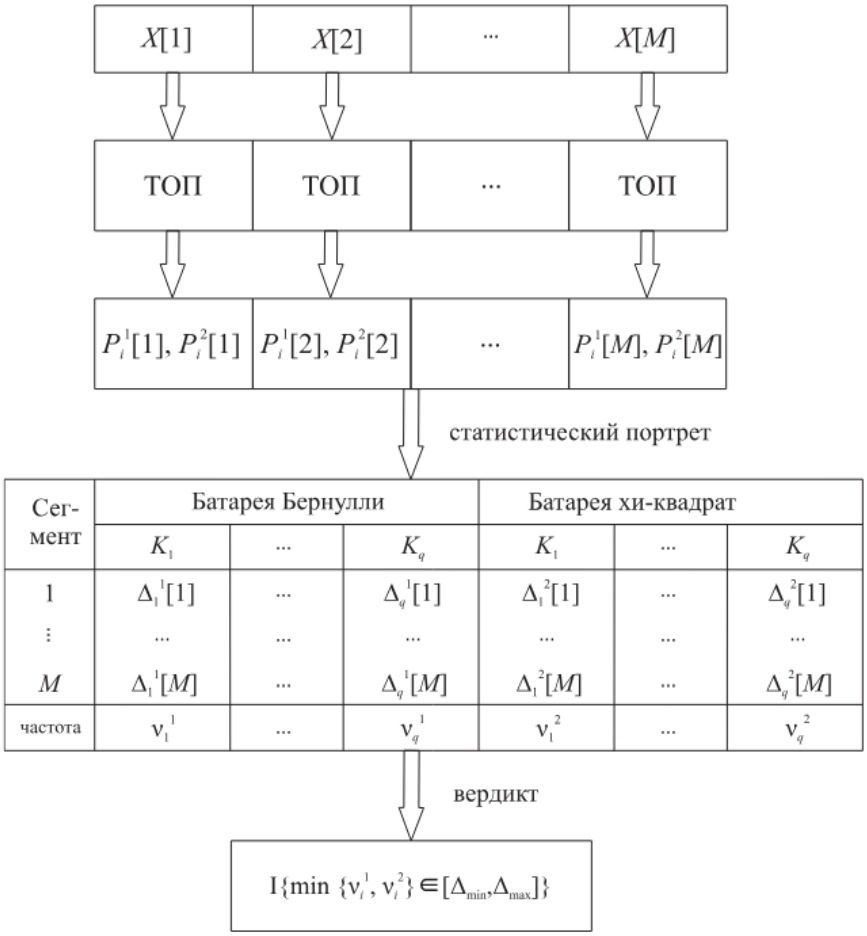


Рисунок 2.2 – Схема тестирования нескольких последовательностей

2.13 Неточности документа методики

В процессе разработки программы требовалось внимательно вчитываться в текст методики, в итоге было обнаружено несколько неточностей и опечаток:

1. В методике указано (раздел 5.14 методики) для тестирования одной последовательности (ТОП) длины 10 мегабайт рекомендуемое разбиение 200 фрагментов (m = 200), тогда получается 200 фрагментов по 419 тысяч символов. При этом в тестах МДРП и МДРН рекомендуются параметры L = 20 и L = 17 соответственно, в самих тестах также имеются проверки условий применения тестов.

   Для МДРН[17] условие $419430/17/2^{17} \geq 10$, ($0.18 \geq 10$) не выполняется, то есть рекомендуемые параметры не подходят. По формуле условия должно выполняться $419430 \geq 10*L*2^L$, тогда подходящее значение параметра для последовательности длины 10 мегабайт L = 11: $419430/11/2^{11} \geq 10$ ($18.6 \geq 10$).

   Для МДРП[20] условие $419430 \geq 20 * 2^{20}$ ($419430 \geq 20971000$) не выполняется, то есть рекомендуемые параметры не подходят. По формуле условия должно выполняться $419430 \geq 20 * 2^L$, тогда

подходящее значение параметра для последовательности длины 10 мегабайт L = 14: $419430 \geq 20 * 2^{14}$ ($419430 \geq 327680$).

2. В тесте аппроксимации энтропии [3, пункт 5.7] на шаге 3 используется логарифм по основанию 2, затем в шаге 5 используется натуральный, из-за чего в итоговой статистике появляется большое число, не отражающее результаты теста. Необходимо на шаге 3 логарифм исправить на натуральный, в таком случае ошибки не возникнет. В процессе изучения ошибки шаги алгоритма сверялись с аналогичным алгоритмом из батареи тестов NIST [4].
3. При тестировании нескольких последователей [3, пункт 5.13] на шаге 5 требуется найти минимальную относительную частоту прохождения теста $\nu_{\min}$ (среди посчитанных на шаге 3), однако в методике показан поиск не среди относительных частот, а среди Δ_i^b из шага 2 – вероятно допущена опечатка, ведь они обозначают результаты прохождения теста K_i в составе батарей, а не искомые частоты, вместо Δ_i^b должно быть v_i^b.

3. ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ТЕСТИРОВАНИЯ ПОСЛЕДОВАТЕЛЬНОСТЕЙ СЛУЧАЙНЫХ ВЕЛИЧИН

3.1 Основные компоненты программного комплекса

В программном комплексе реализовано 13 файлов:

1. Тест многомерной дискретной равномерности по непересекающимся отрезкам (Приложение А).
2. Тест многомерной дискретной равномерности по пересекающимся отрезкам (Приложение Б).
3. Тест «пустых ящиков» (Приложение В).
4. Тест серий (Приложение Г).
5. Тест длинных серий (Приложение Д).
6. Тест аппроксимации энтропии (Приложение Е).
7. Тест скалярного произведения (Приложение Ж).
8. Спектральный тест (Приложение И).
9. Батарея Бернулли (Приложение К).
10. Батарея хи-квадрат (Приложение Л).
11. Логика тестирования одной последовательности (Приложение М).
12. Логика тестирования нескольких последовательностей (Приложение Н).
13. Общая логика разбора входных параметров и запуска процедуры тестирования с соответствующими параметрами (Приложение П).

Разработанный программный комплекс опубликован в открытом доступе на ресурсе GitHub по адресу: https://github.com/Ananasiig/Best/.

3.2 Демонстрация работы программного комплекса

В процессе работы были разработаны реализации всех задуманных 13 файлов на языке программирования C++, также для корректной работы алгоритмов были реализованы функции, вычисляющие значения неполной гамма-функции [5][6] и распределения Хи-квадрат [7], функции нормального распределения, а также функция, реализующая БПФ [8] (быстрое преобразование Фурье).

Для ускорения работы программы была добавлена многопоточность с помощью OpenMP, позволяющая в разы ускорить тестирование крупных файлов. В программе предполагается обработка 10 мегабайт одним потоком, программа использует не более максимального числа потоков процессора минус два для сохранения общей работоспособности компьютера. В случае, если файл достаточно большой, программа читает его частями, кратными количеству доступных потоков, после чтения определённой части файла потоки разбирают прочитанное по 10 мегабайт и начинают обработку, после чего, если конец файла не был достигнут, продолжается чтение.

Также в процессе был разработан графический интерфейс с использованием Qt6[9] (Рисунок 3.1).

Разработанный интерфейс позволяет выбирать необходимый для тестирования бинарный файл с помощью обзора каталога файлов (Рисунок 3.2), а также задания пути в строке (Рисунок 3.3).

При нажатии кнопки «Старт» запускается окно выбора места сохранения выходного файла и его имени, после чего начинается алгоритм тестирования файла, запускается таймер. При нажатии кнопки «Стоп» можно в любой момент прервать тестирование.

В процессе тестирования заполняется шкала прогресса, отображающая протестированную долю файла.

По завершению тестирования в окне графического интерфейса заполняется таблица, состоящая из интегральных P-значений, полученных в батареях Бернулли и хи-квадрат: минимальное из значений двух батарей для 8 тестов, а так же P-значение для каждого из тестов, итого 17 столбцов, каждая строка представляет собой подстроку файла длиной 10 мегабайт или 1 мегабайт, если размер всего файла менее 10 мегабайт. В случае, если P-значение менее требуемого 0.0001, то в таблице оно отображается красным цветом текста.

Над таблицей имеется поле, в котором отображается была принята или отвергнута гипотеза H_0. При 1 строке вывода таблицы в поле отображаются результаты батарей Бернулли и хи-квадрат для тестирования одной последовательности, иначе отображается единственный результат для тестирования нескольких последовательностей.

Ниже представлены результаты тестирования последовательностей длиной 100 мегабайт и 1 мегабайт (Рисунок 3.4, Рисунок 3.5).

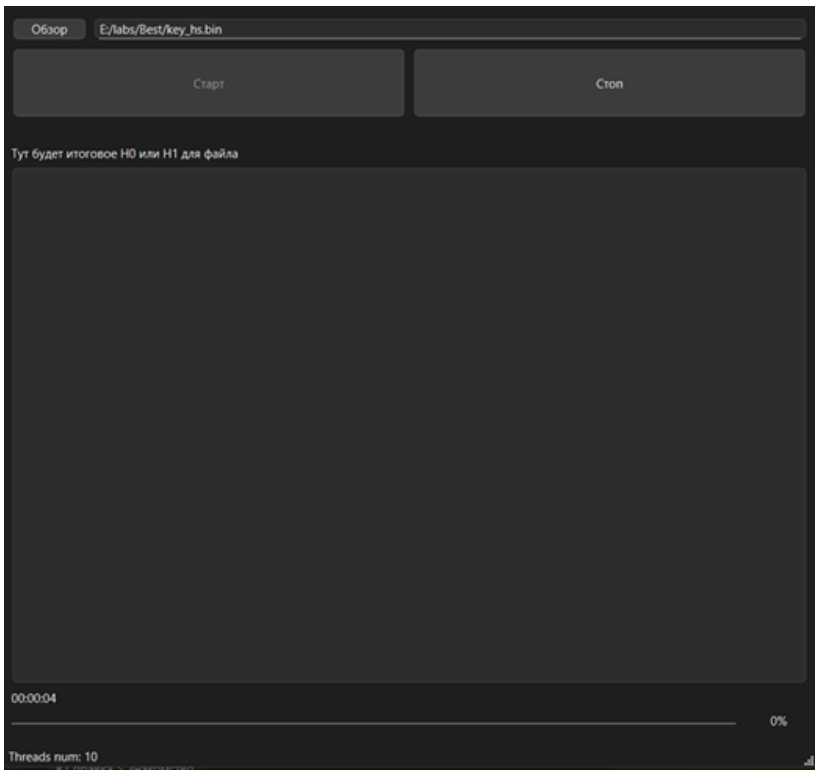


Рисунок 3.1 – Графический интерфейс приложения

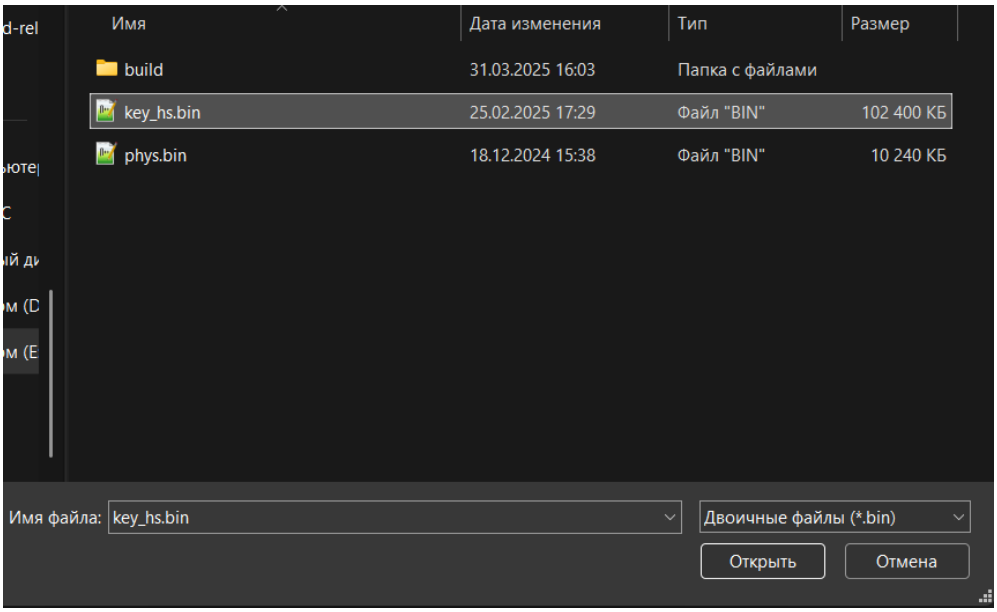


Рисунок 3.2 – Выбор файла через обозреватель.

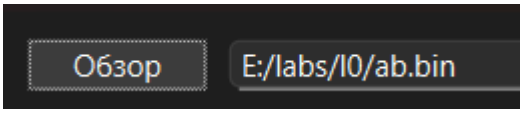


Рисунок 3.3 – кнопка «Обзор» и редактируемая строка пути к файлу.

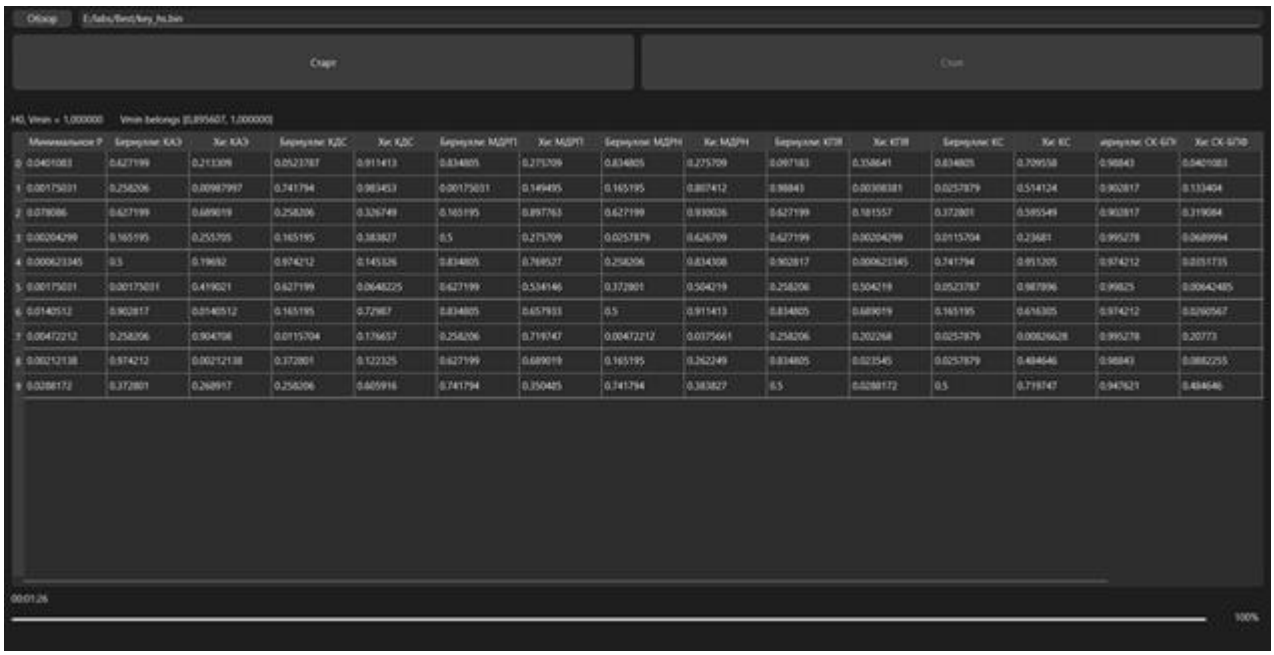


Рисунок 3.4 – Выходные P-значения батарей и принятие H_0
для файла длиной 100 мегабайт

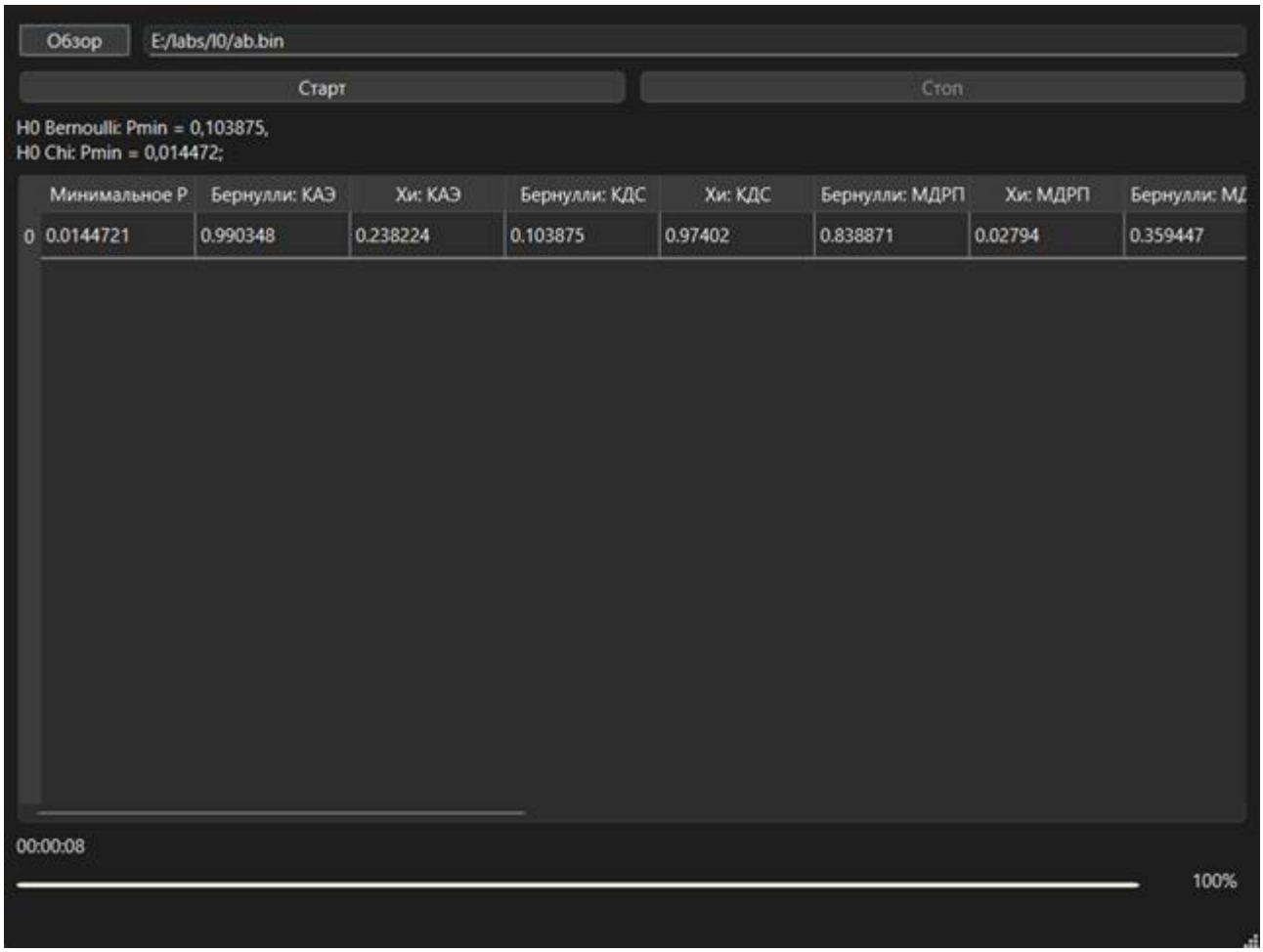
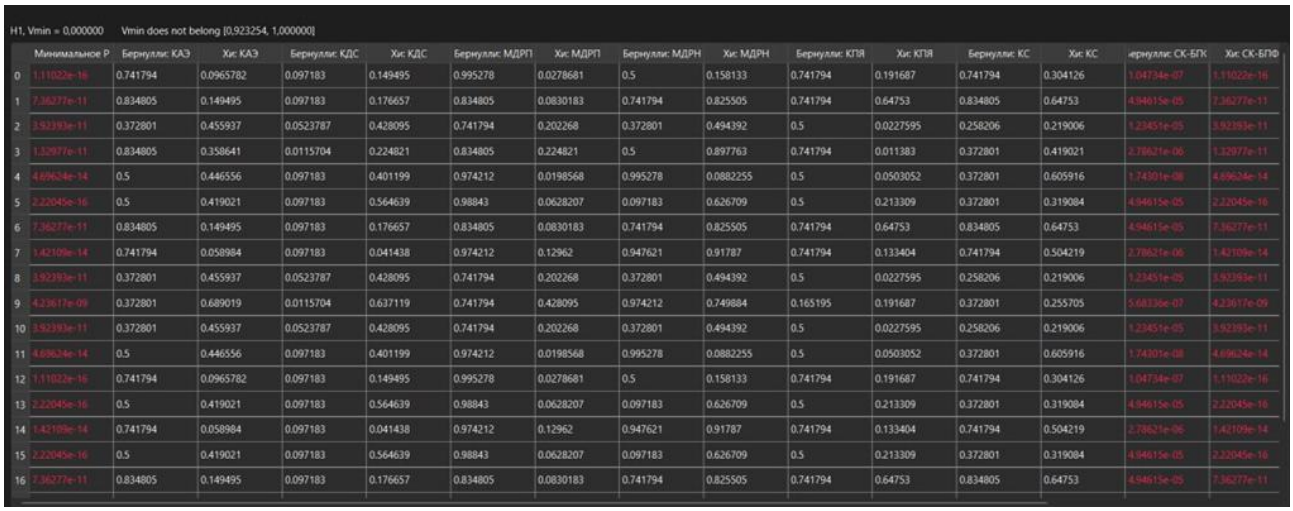


Рисунок 3.5 – Выходные P-значения батарей и принятие H_0
для файла длиной 1 мегабайт

3.3 Результаты тестирования некоторых генераторов случайных чисел

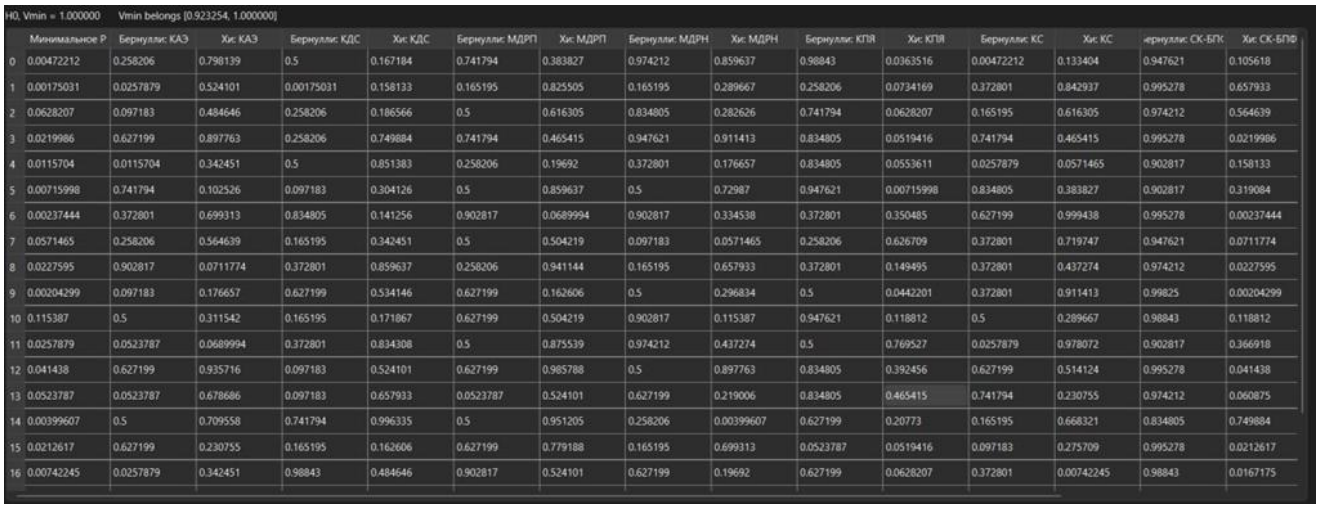
Для тестирования были выбраны несколько разных алгоритмов, один из них – базовая функция rand() языка C++, на рисунке 3.6 можно увидеть результат тестирования файла длиной 200 мегабайт, сгенерированного с её помощью. По результатам выявлен недостаточно большой период функции.


H1, Vmin = 0,000000 Vmin does not belong [0,923254, 1,000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПК	Хи: СК-БПФ
0	1.11022e-16	0.741794	0.0965782	0.097183	0.149495	0.995278	0.0278681	0.5	0.158133	0.741794	0.191687	0.741794	0.304126	1.04734e-07	1.11022e-16
1	7.36277e-11	0.834805	0.149495	0.097183	0.176657	0.834805	0.0830183	0.741794	0.825505	0.741794	0.64753	0.834805	0.64753	4.94615e-05	7.36277e-11
2	3.92393e-11	0.372801	0.455937	0.0523787	0.428095	0.741794	0.202268	0.372801	0.494392	0.5	0.0227595	0.258206	0.219006	1.23451e-05	3.92393e-11
3	1.32977e-11	0.834805	0.358641	0.0115704	0.224821	0.834805	0.224821	0.5	0.897763	0.741794	0.011383	0.372801	0.419021	2.78621e-06	1.32977e-11
4	4.69634e-14	0.5	0.446556	0.097183	0.401199	0.974212	0.0198568	0.995278	0.0882255	0.5	0.0503052	0.372801	0.605916	1.74301e-08	4.69624e-14
5	2.22045e-16	0.5	0.419021	0.097183	0.564639	0.98843	0.0628207	0.097183	0.626709	0.5	0.213309	0.372801	0.319084	4.94615e-05	2.22045e-16
6	7.36277e-11	0.834805	0.149495	0.097183	0.176657	0.834805	0.0830183	0.741794	0.825505	0.741794	0.64753	0.834805	0.64753	4.94615e-05	7.36277e-11
7	1.42109e-14	0.741794	0.058984	0.097183	0.041438	0.974212	0.12962	0.947621	0.91787	0.741794	0.133404	0.741794	0.504219	2.78621e-06	1.42109e-14
8	3.92393e-11	0.372801	0.455937	0.0523787	0.428095	0.741794	0.202268	0.372801	0.494392	0.5	0.0227595	0.258206	0.219006	1.23451e-05	3.92393e-11
9	4.23617e-09	0.372801	0.689019	0.0115704	0.637119	0.741794	0.428095	0.974212	0.749884	0.165195	0.191687	0.372801	0.255705	5.68336e-07	4.23617e-09
10	3.92393e-11	0.372801	0.455937	0.0523787	0.428095	0.741794	0.202268	0.372801	0.494392	0.5	0.0227595	0.258206	0.219006	1.23451e-05	3.92393e-11
11	4.69634e-14	0.5	0.446556	0.097183	0.401199	0.974212	0.0198568	0.995278	0.0882255	0.5	0.0503052	0.372801	0.605916	1.74301e-08	4.69624e-14
12	1.11022e-16	0.741794	0.0965782	0.097183	0.149495	0.995278	0.0278681	0.5	0.158133	0.741794	0.191687	0.741794	0.304126	1.04734e-07	1.11022e-16
13	2.22045e-16	0.5	0.419021	0.097183	0.564639	0.98843	0.0628207	0.097183	0.626709	0.5	0.213309	0.372801	0.319084	4.94615e-05	2.22045e-16
14	1.42109e-14	0.741794	0.058984	0.097183	0.041438	0.974212	0.12962	0.947621	0.91787	0.741794	0.133404	0.741794	0.504219	2.78621e-06	1.42109e-14
15	2.22045e-16	0.5	0.419021	0.097183	0.564639	0.98843	0.0628207	0.097183	0.626709	0.5	0.213309	0.372801	0.319084	4.94615e-05	2.22045e-16
16	7.36277e-11	0.834805	0.149495	0.097183	0.176657	0.834805	0.0830183	0.741794	0.825505	0.741794	0.64753	0.834805	0.64753	4.94615e-05	7.36277e-11

Рисунок 3.6 – Тестирование генерации чисел с использованием базовой функции rand() языка программирования C++

Далее был протестирован файл такой же длины, сгенерированный с помощью алгоритма, который не является криптографически стойким, – вихря Мерсенна, тесты он успешно прошёл (Рисунок 3.7).


H0, Vmin = 1.000000 Vmin belongs [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПК	Хи: СК-БПФ
0	0.00472212	0.258206	0.798139	0.5	0.167184	0.741794	0.383827	0.974212	0.859637	0.98843	0.0363516	0.00472212	0.133404	0.947621	0.105618
1	0.00175031	0.0257879	0.524101	0.00175031	0.158133	0.165195	0.825505	0.165195	0.289667	0.258206	0.0734169	0.372801	0.842937	0.995278	0.657933
2	0.0628207	0.097183	0.484646	0.258206	0.186566	0.5	0.616305	0.834805	0.282626	0.741794	0.0628207	0.165195	0.616305	0.974212	0.564639
3	0.0219986	0.627199	0.897763	0.258206	0.749884	0.741794	0.465415	0.947621	0.911413	0.834805	0.0519416	0.741794	0.465415	0.995278	0.0219986
4	0.0115704	0.0115704	0.342451	0.5	0.851383	0.258206	0.19692	0.372801	0.176657	0.834805	0.0553611	0.0257879	0.0571465	0.902817	0.158133
5	0.00715998	0.741794	0.102526	0.097183	0.304126	0.5	0.859637	0.5	0.72987	0.947621	0.00715998	0.834805	0.383827	0.902817	0.319084
6	0.00237444	0.372801	0.699313	0.834805	0.141256	0.902817	0.0689994	0.902817	0.334538	0.372801	0.350485	0.627199	0.999438	0.995278	0.00237444
7	0.0571465	0.258206	0.564639	0.165195	0.342451	0.5	0.504219	0.097183	0.0571465	0.258206	0.626709	0.372801	0.719747	0.947621	0.0711774
8	0.0227595	0.902817	0.0711774	0.372801	0.859637	0.258206	0.941144	0.165195	0.657933	0.372801	0.149495	0.372801	0.437274	0.974212	0.0227595
9	0.00204299	0.097183	0.176657	0.627199	0.534146	0.627199	0.162606	0.5	0.296834	0.5	0.0442201	0.372801	0.911413	0.99825	0.00204299
10	0.115387	0.5	0.311542	0.165195	0.171867	0.627199	0.504219	0.902817	0.115387	0.947621	0.118812	0.5	0.289667	0.98843	0.118812
11	0.0257879	0.0523787	0.0689994	0.372801	0.834308	0.5	0.875539	0.974212	0.437274	0.5	0.769527	0.0257879	0.978072	0.902817	0.366918
12	0.041438	0.627199	0.935716	0.097183	0.524101	0.627199	0.985788	0.5	0.897763	0.834805	0.392456	0.627199	0.514124	0.995278	0.041438
13	0.0523787	0.0523787	0.678686	0.097183	0.657933	0.0523787	0.524101	0.627199	0.219006	0.834805	0.465415	0.741794	0.230755	0.974212	0.060875
14	0.00399607	0.5	0.709558	0.741794	0.996335	0.5	0.951205	0.258206	0.00399607	0.627199	0.20773	0.165195	0.668321	0.834805	0.749884
15	0.0212617	0.627199	0.230755	0.165195	0.162606	0.627199	0.779188	0.165195	0.699313	0.0523787	0.0519416	0.097183	0.275709	0.995278	0.0212617
16	0.00742245	0.0257879	0.342451	0.98843	0.484646	0.902817	0.524101	0.627199	0.19692	0.627199	0.0628207	0.372801	0.00742245	0.98843	0.0167175

Рисунок 3.7 – Тестирование последовательности чисел, сгенерированной вихрем Мерсенна с равномерным распределением и реализацией на языке программирования C++

Также в рамках эксперимента были протестированы РСЛОС с разными периодами:

- РСЛОС 8-бит: $x^8+x^4+x^3+x^2+1$ (Рисунок 3.8);
- РСЛОС 16-бит: $x^{16}+x^{14}+x^{13}+x^{11}+1$ (Рисунок 3.9);
- РСЛОС 24-бит: $x^{24}+x^{22}+x^{20}+x^{19}+1$ (Рисунок 3.10);
- РСЛОС 32-бит: $x^{32}+x^{22}+x^2+x+1$ (Рисунок 3.11).

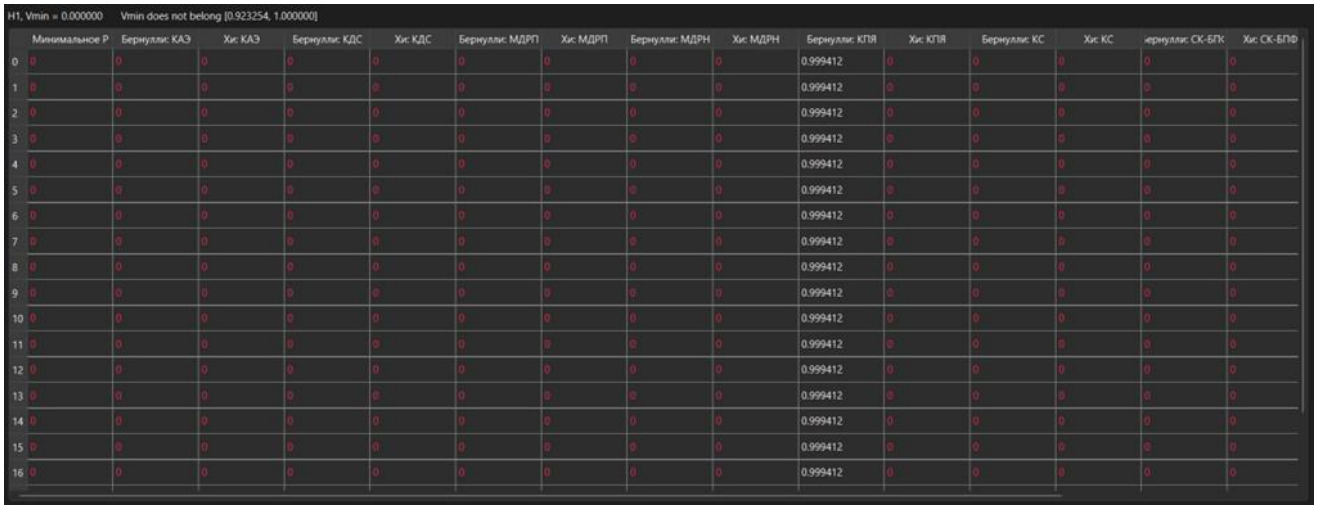


Рисунок 3.8 – Тестирование последовательности чисел, сгенерированной слабым РСЛОС на 8 бит с реализацией на языке программирования C++

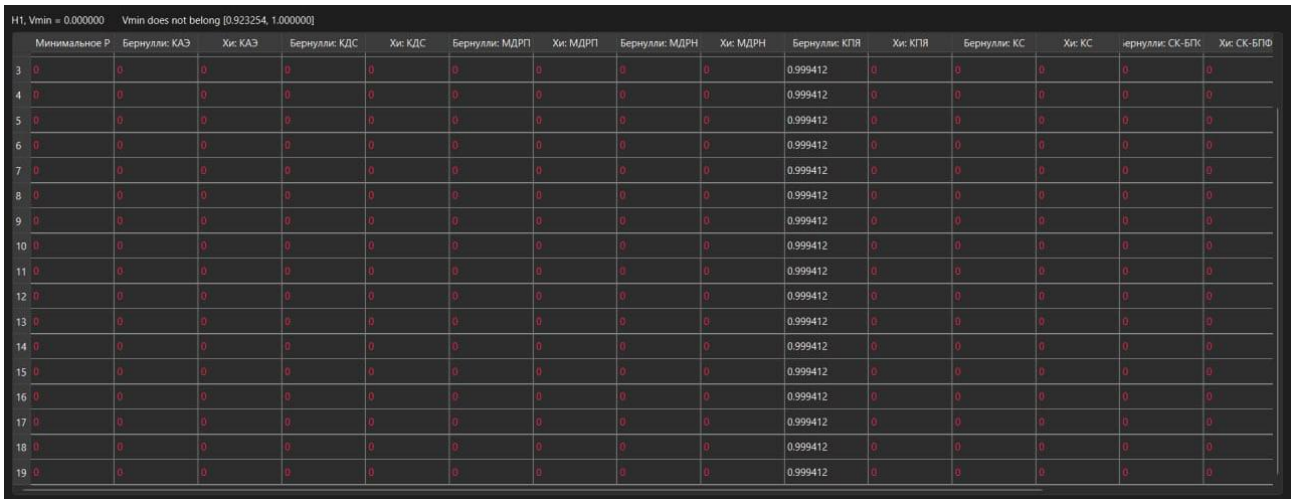
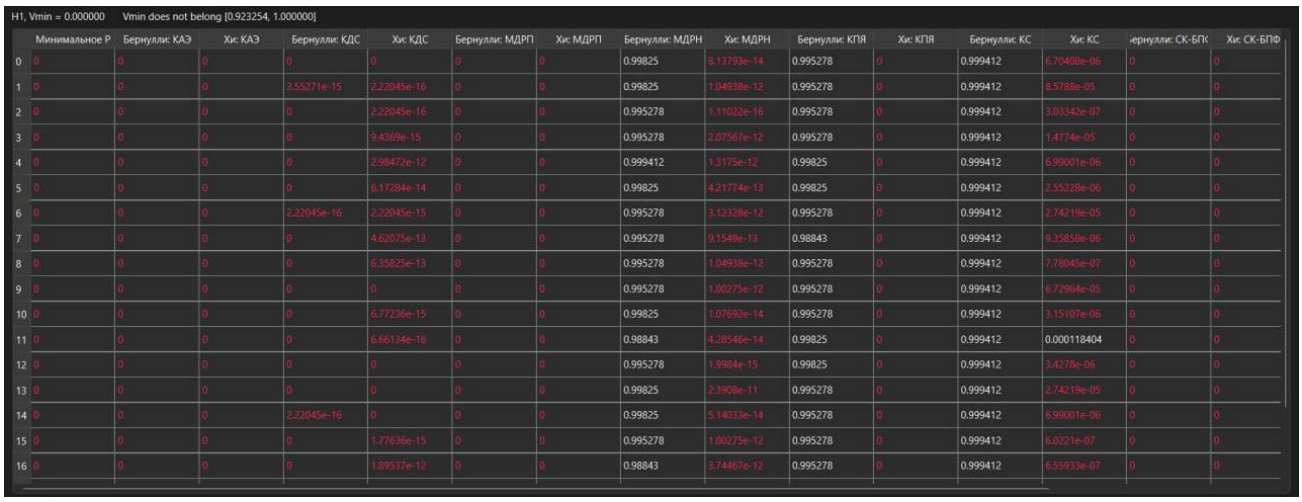


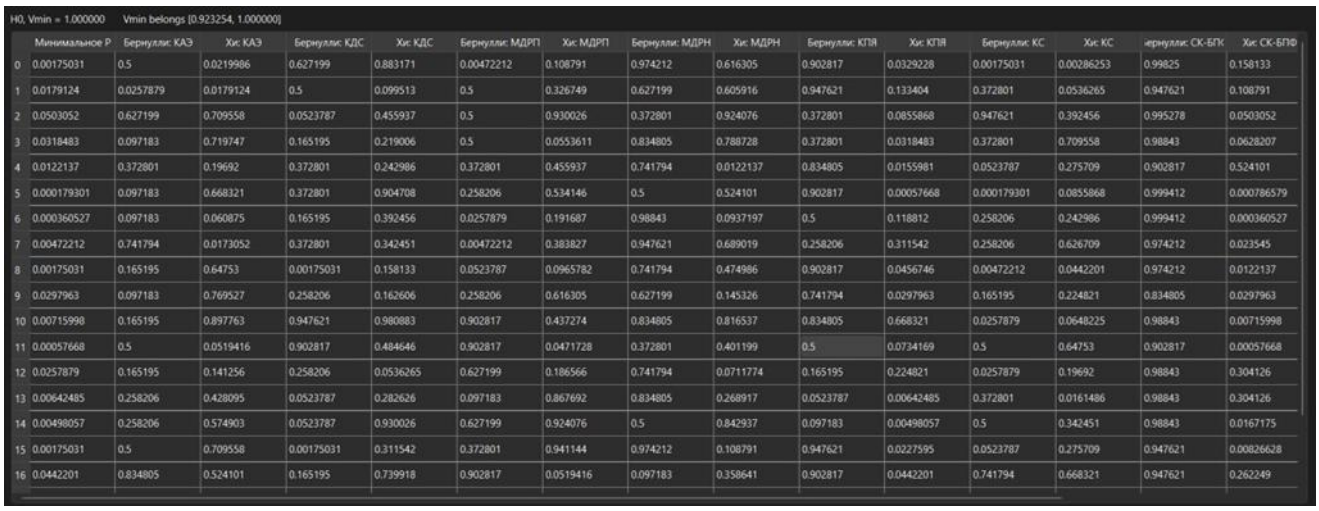
Рисунок 3.9 – Тестирование последовательности чисел, сгенерированной РСЛОС на 16 бит с реализацией на языке программирования C++



H1, Vmin = 0.000000 Vmin does not belong [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	0	0	0	0	0	0	0	0.99825	8.13793e-14	0.995278	0	0.999412	6.70408e-06	0	0
1	0	0	0	3.55271e-15	2.22045e-16	0	0	0.99825	1.04938e-12	0.995278	0	0.999412	8.5788e-05	0	0
2	0	0	0	0	2.22045e-16	0	0	0.995278	1.11022e-16	0.995278	0	0.999412	3.03342e-07	0	0
3	0	0	0	0	9.4369e-15	0	0	0.995278	2.07567e-12	0.995278	0	0.999412	1.4774e-05	0	0
4	0	0	0	0	2.98472e-12	0	0	0.999412	1.3175e-12	0.99825	0	0.999412	6.99001e-06	0	0
5	0	0	0	0	6.17284e-14	0	0	0.99825	4.21774e-13	0.99825	0	0.999412	2.55228e-06	0	0
6	0	0	0	2.22045e-16	2.22045e-15	0	0	0.995278	3.12328e-12	0.995278	0	0.999412	2.74219e-05	0	0
7	0	0	0	0	4.62075e-13	0	0	0.995278	9.1549e-13	0.98843	0	0.999412	9.35858e-06	0	0
8	0	0	0	0	6.35825e-13	0	0	0.995278	1.04938e-12	0.995278	0	0.999412	7.78045e-07	0	0
9	0	0	0	0	0	0	0	0.995278	1.00275e-12	0.995278	0	0.999412	6.72964e-05	0	0
10	0	0	0	0	6.77236e-15	0	0	0.99825	1.07692e-14	0.995278	0	0.999412	3.15107e-06	0	0
11	0	0	0	0	6.66134e-16	0	0	0.98843	4.28546e-14	0.99825	0	0.999412	0.000118404	0	0
12	0	0	0	0	0	0	0	0.995278	1.9984e-15	0.99825	0	0.999412	3.4278e-06	0	0
13	0	0	0	0	0	0	0	0.99825	2.3908e-11	0.995278	0	0.999412	2.74219e-05	0	0
14	0	0	0	2.22045e-16	0	0	0	0.99825	5.14033e-14	0.995278	0	0.999412	6.99001e-06	0	0
15	0	0	0	0	1.77636e-15	0	0	0.995278	1.00275e-12	0.995278	0	0.999412	6.0221e-07	0	0
16	0	0	0	0	1.89537e-12	0	0	0.98843	3.74467e-12	0.995278	0	0.999412	6.55933e-07	0	0

Рисунок 3.10 – Тестирование последовательности чисел, сгенерированной РСЛОС на 24 бит с реализацией на языке программирования C++



H0, Vmin = 1.000000 Vmin belongs [0.923254, 1.000000]

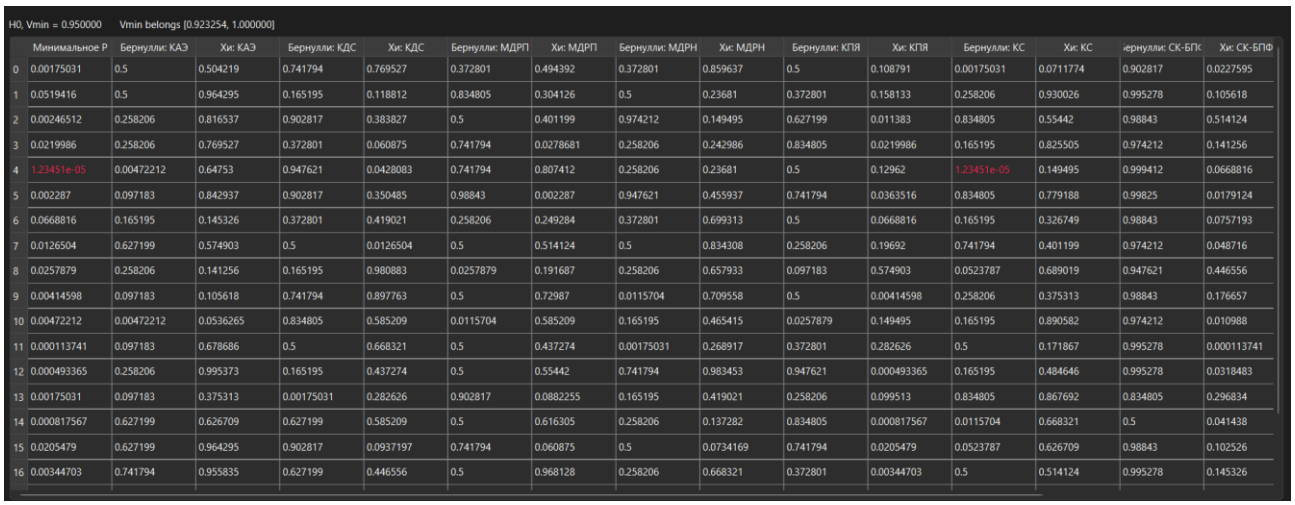
	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	0.00175031	0.5	0.0219986	0.627199	0.883171	0.00472212	0.108791	0.974212	0.616305	0.902817	0.0329228	0.00175031	0.00286253	0.99825	0.158133
1	0.0179124	0.0257879	0.0179124	0.5	0.099513	0.5	0.326749	0.627199	0.605916	0.947621	0.133404	0.372801	0.0536265	0.947621	0.108791
2	0.0503052	0.627199	0.709558	0.0523787	0.455937	0.5	0.930026	0.372801	0.924076	0.372801	0.0855868	0.947621	0.392456	0.995278	0.0503052
3	0.0318483	0.097183	0.719747	0.165195	0.219006	0.5	0.0553611	0.834805	0.788728	0.372801	0.0318483	0.372801	0.709558	0.98843	0.0628207
4	0.0122137	0.372801	0.19692	0.372801	0.242986	0.372801	0.455937	0.741794	0.0122137	0.834805	0.0155981	0.0523787	0.275709	0.902817	0.524101
5	0.000179301	0.097183	0.668321	0.372801	0.904708	0.258206	0.534146	0.5	0.524101	0.902817	0.00057668	0.000179301	0.0855868	0.999412	0.000786579
6	0.000360527	0.097183	0.060875	0.165195	0.392456	0.0257879	0.191687	0.98843	0.0937197	0.5	0.118812	0.258206	0.242986	0.999412	0.000360527
7	0.00472212	0.741794	0.0173052	0.372801	0.342451	0.00472212	0.383827	0.947621	0.689019	0.258206	0.311542	0.258206	0.626709	0.974212	0.023545
8	0.00175031	0.165195	0.64753	0.00175031	0.158133	0.0523787	0.0965782	0.741794	0.474986	0.902817	0.0456746	0.00472212	0.0442201	0.974212	0.0122137
9	0.0297963	0.097183	0.769527	0.258206	0.162606	0.258206	0.616305	0.627199	0.145326	0.741794	0.0297963	0.165195	0.224821	0.834805	0.0297963
10	0.00715998	0.165195	0.897763	0.947621	0.980883	0.902817	0.437274	0.834805	0.816537	0.834805	0.668321	0.0257879	0.0648225	0.98843	0.00715998
11	0.00057668	0.5	0.0519416	0.902817	0.484646	0.902817	0.0471728	0.372801	0.401199	0.5	0.0734169	0.5	0.64753	0.902817	0.00057668
12	0.0257879	0.165195	0.141256	0.258206	0.0536265	0.627199	0.186566	0.741794	0.0711774	0.165195	0.224821	0.0257879	0.19692	0.98843	0.304126
13	0.00642485	0.258206	0.428095	0.0523787	0.282626	0.097183	0.867692	0.834805	0.268917	0.0523787	0.00642485	0.372801	0.0161486	0.98843	0.304126
14	0.00498057	0.258206	0.574903	0.0523787	0.930026	0.627199	0.924076	0.5	0.842937	0.097183	0.00498057	0.5	0.342451	0.98843	0.0167175
15	0.00175031	0.5	0.709558	0.00175031	0.311542	0.372801	0.941144	0.974212	0.108791	0.947621	0.0227595	0.0523787	0.275709	0.947621	0.00826628
16	0.0442201	0.834805	0.524101	0.165195	0.739918	0.902817	0.0519416	0.097183	0.358641	0.902817	0.0442201	0.741794	0.668321	0.947621	0.262249

Рисунок 3.11 – Тестирование последовательности чисел, сгенерированной РСЛОС на 32 бит с реализацией на языке C++

По результатам тестирования можем заметить, что программный комплекс распознаёт последовательности с низким минимальным периодом, пусть и недостаточно строго оценивает криптографическую стойкость получившейся последовательности.

3.4 Результаты тестирования выходной последовательности генератора случайных чисел с искусственно добавленными ошибками

С помощью приложения был протестирован файл длины 200 мегабайт, сгенерированной ФГСЧ «Ключ-ВС» (Рисунок 3.12).



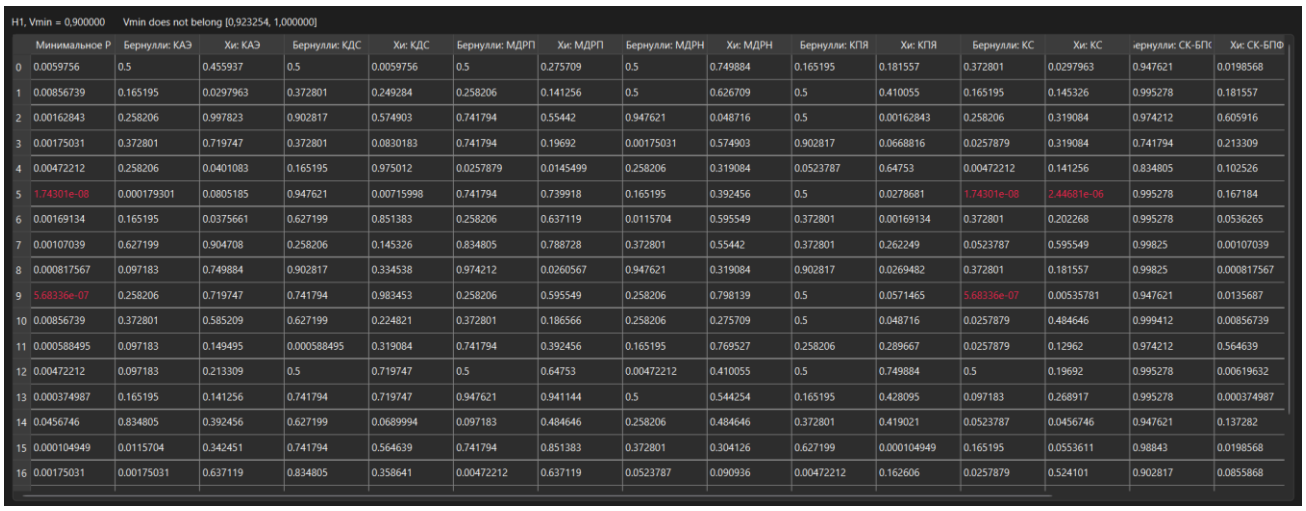
H0, Vmin = 0.950000 Vmin belongs [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	0.00175031	0.5	0.504219	0.741794	0.769527	0.372801	0.494392	0.372801	0.859637	0.5	0.108791	0.00175031	0.0711774	0.902817	0.0227595
1	0.0519416	0.5	0.964295	0.165195	0.118812	0.834805	0.304126	0.5	0.23681	0.372801	0.158133	0.258206	0.930026	0.995278	0.105618
2	0.00246512	0.258206	0.816537	0.902817	0.383827	0.5	0.401199	0.974212	0.149495	0.627199	0.011383	0.834805	0.55442	0.98843	0.514124
3	0.0219986	0.258206	0.769527	0.372801	0.060875	0.741794	0.0278681	0.258206	0.242986	0.834805	0.0219986	0.165195	0.825505	0.974212	0.141256
4	1.23451e-05	0.00472212	0.64753	0.947621	0.0428083	0.741794	0.807412	0.258206	0.23681	0.5	0.12962	1.23451e-05	0.149495	0.999412	0.0668816
5	0.002287	0.097183	0.842937	0.902817	0.350485	0.98843	0.002287	0.947621	0.455937	0.741794	0.0363516	0.834805	0.779188	0.99825	0.0179124
6	0.0668816	0.165195	0.145326	0.372801	0.419021	0.258206	0.249284	0.372801	0.699313	0.5	0.0668816	0.165195	0.326749	0.98843	0.0757193
7	0.0126504	0.627199	0.574903	0.5	0.0126504	0.5	0.514124	0.5	0.834308	0.258206	0.19692	0.741794	0.401199	0.974212	0.048716
8	0.0257879	0.258206	0.141256	0.165195	0.980883	0.0257879	0.191687	0.258206	0.657933	0.097183	0.574903	0.0523787	0.689019	0.947621	0.446556
9	0.00414598	0.097183	0.105618	0.741794	0.897763	0.5	0.72987	0.0115704	0.709558	0.5	0.00414598	0.258206	0.375313	0.98843	0.176657
10	0.00472212	0.00472212	0.0536265	0.834805	0.585209	0.0115704	0.585209	0.165195	0.465415	0.0257879	0.149495	0.165195	0.890582	0.974212	0.010988
11	0.000113741	0.097183	0.678686	0.5	0.668321	0.5	0.437274	0.00175031	0.268917	0.372801	0.282626	0.5	0.171867	0.995278	0.000113741
12	0.000493365	0.258206	0.995373	0.165195	0.437274	0.5	0.55442	0.741794	0.983453	0.947621	0.000493365	0.165195	0.484646	0.995278	0.0318483
13	0.00175031	0.097183	0.375313	0.00175031	0.282626	0.902817	0.0882255	0.165195	0.419021	0.258206	0.099513	0.834805	0.867692	0.834805	0.296834
14	0.000817567	0.627199	0.626709	0.627199	0.585209	0.5	0.616305	0.258206	0.137282	0.834805	0.000817567	0.0115704	0.668321	0.5	0.041438
15	0.0205479	0.627199	0.964295	0.902817	0.0937197	0.741794	0.060875	0.5	0.0734169	0.741794	0.0205479	0.0523787	0.626709	0.98843	0.102526
16	0.00344703	0.741794	0.955835	0.627199	0.446556	0.5	0.968128	0.258206	0.668321	0.372801	0.00344703	0.5	0.514124	0.995278	0.145326

Рисунок 3.12 – Тестирование последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»

После проверки данного файла были произведены различные его модификации и их тестирование:

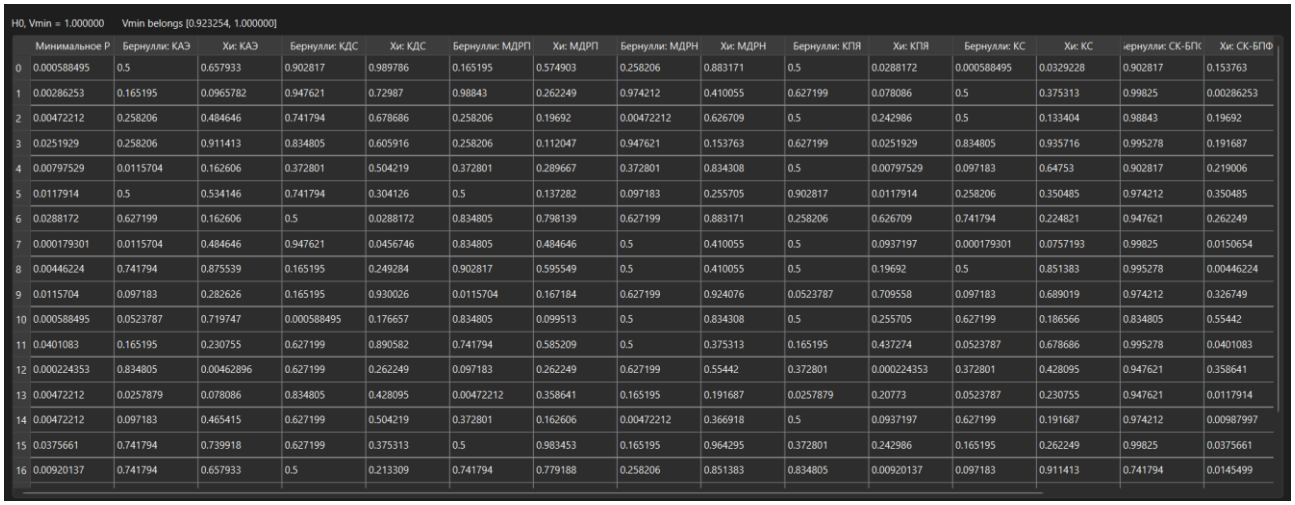
1. Фиксированный байт 0x55 каждые 256 байт (Рисунок 3.13);
2. Инвертирование 7-й и 15-й бит каждые 512 байт (Рисунок 3.14);
3. Повторение 2-байтовой последовательности, 0xDE и 0xAD каждые 2048 байт (Рисунок 3.15);
4. Дублирование предыдущего байта каждые 256 байт (Рисунок 3.17);
5. Повторяющийся каждые 740 байт паттерн из 3 разных байтов: 0xAA 0xBB 0xCC (Рисунок 3.18);
6. Фиксированная 4-байтовая метка 0xCAFEBABE в начале и каждые 1024 байт (Рисунок 3.19);
7. Чередование двух значений 0x33 и 0xCC каждые 256 байт (Рисунок 3.20);
8. Каждые 1024 байт меняются 5 идущих подряд байтов:1,2,3,4,5 → 2,2,1,1,1 (Рисунок 3.22).



H1, Vmin = 0,900000 Vmin does not belong [0,923254, 1,000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	0.0059756	0.5	0.455937	0.5	0.0059756	0.5	0.275709	0.5	0.749884	0.165195	0.181557	0.372801	0.0297963	0.947621	0.0198568
1	0.00856739	0.165195	0.0297963	0.372801	0.249284	0.258206	0.141256	0.5	0.626709	0.5	0.410055	0.165195	0.145326	0.995278	0.181557
2	0.00162843	0.258206	0.997823	0.902817	0.574903	0.741794	0.55442	0.947621	0.048716	0.5	0.00162843	0.258206	0.319084	0.974212	0.605916
3	0.00175031	0.372801	0.719747	0.372801	0.0830183	0.741794	0.19692	0.00175031	0.574903	0.902817	0.0668816	0.0257879	0.319084	0.741794	0.213309
4	0.00472212	0.258206	0.0401083	0.165195	0.975012	0.0257879	0.0145499	0.258206	0.319084	0.0523787	0.64753	0.00472212	0.141256	0.834805	0.102526
5	1.74301e-08	0.000179301	0.0805185	0.947621	0.00715998	0.741794	0.739918	0.165195	0.392456	0.5	0.0278681	1.74301e-08	2.44681e-06	0.995278	0.167184
6	0.00169134	0.165195	0.0375661	0.627199	0.851383	0.258206	0.637119	0.0115704	0.595549	0.372801	0.00169134	0.372801	0.202268	0.995278	0.0536265
7	0.00107039	0.627199	0.904708	0.258206	0.145326	0.834805	0.788728	0.372801	0.55442	0.372801	0.262249	0.0523787	0.595549	0.99825	0.00107039
8	0.000817567	0.097183	0.749884	0.902817	0.334538	0.974212	0.0260567	0.947621	0.319084	0.902817	0.0269482	0.372801	0.181557	0.99825	0.000817567
9	5.68336e-07	0.258206	0.719747	0.741794	0.983453	0.258206	0.595549	0.258206	0.798139	0.5	0.0571465	5.68336e-07	0.00535781	0.947621	0.0135687
10	0.00856739	0.372801	0.585209	0.627199	0.224821	0.372801	0.186566	0.258206	0.275709	0.5	0.048716	0.0257879	0.484646	0.999412	0.00856739
11	0.000588495	0.097183	0.149495	0.000588495	0.319084	0.741794	0.392456	0.165195	0.769527	0.258206	0.289667	0.0257879	0.12962	0.974212	0.564639
12	0.00472212	0.097183	0.213309	0.5	0.719747	0.5	0.64753	0.00472212	0.410055	0.5	0.749884	0.5	0.19692	0.995278	0.00619632
13	0.000374987	0.165195	0.141256	0.741794	0.719747	0.947621	0.941144	0.5	0.544254	0.165195	0.428095	0.097183	0.268917	0.995278	0.000374987
14	0.0456746	0.834805	0.392456	0.627199	0.0689994	0.097183	0.484646	0.258206	0.484646	0.372801	0.419021	0.0523787	0.0456746	0.947621	0.137282
15	0.000104949	0.0115704	0.342451	0.741794	0.564639	0.741794	0.851383	0.372801	0.304126	0.627199	0.000104949	0.165195	0.0553611	0.98843	0.0198568
16	0.00175031	0.00175031	0.637119	0.834805	0.358641	0.00472212	0.637119	0.0523787	0.090936	0.00472212	0.162606	0.0257879	0.524101	0.902817	0.0855868

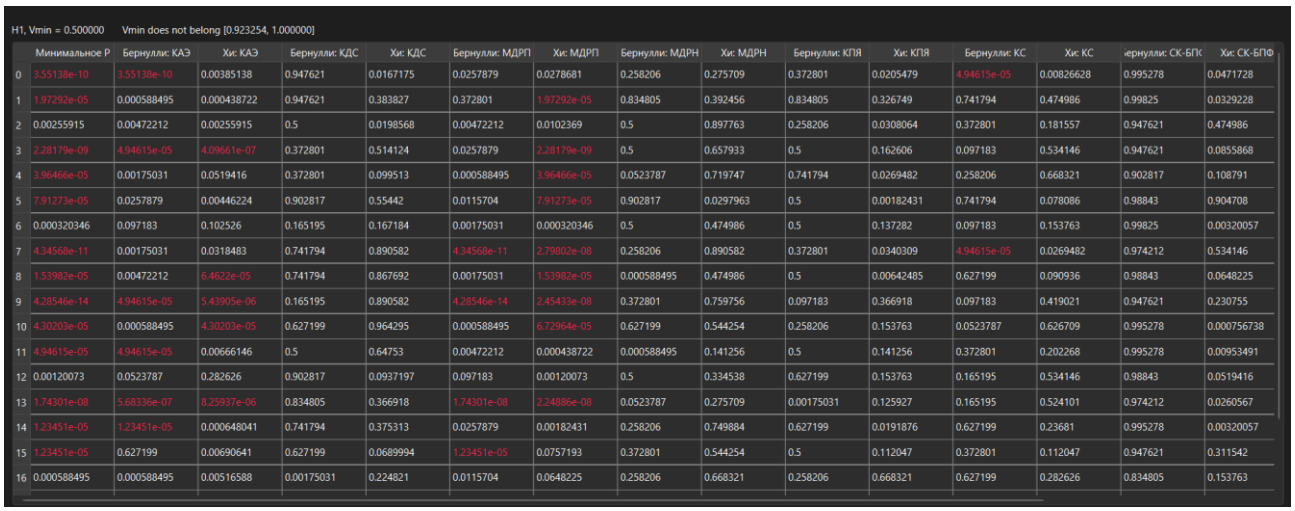
Рисунок 3.13 – Тестирование 1-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»



H0, Vmin = 1.000000 Vmin belongs [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	0.000588495	0.5	0.657933	0.902817	0.989786	0.165195	0.574903	0.258206	0.883171	0.5	0.0288172	0.000588495	0.0329228	0.902817	0.153763
1	0.00286253	0.165195	0.0965782	0.947621	0.72987	0.98843	0.262249	0.974212	0.410055	0.627199	0.078086	0.5	0.375313	0.99825	0.00286253
2	0.00472212	0.258206	0.484646	0.741794	0.678686	0.258206	0.19692	0.00472212	0.626709	0.5	0.242986	0.5	0.133404	0.98843	0.19692
3	0.0251929	0.258206	0.911413	0.834805	0.605916	0.258206	0.112047	0.947621	0.153763	0.627199	0.0251929	0.834805	0.935716	0.995278	0.191687
4	0.00797529	0.0115704	0.162606	0.372801	0.504219	0.372801	0.289667	0.372801	0.834308	0.5	0.00797529	0.097183	0.64753	0.902817	0.219006
5	0.0117914	0.5	0.534146	0.741794	0.304126	0.5	0.137282	0.097183	0.255705	0.902817	0.0117914	0.258206	0.350485	0.974212	0.350485
6	0.0288172	0.627199	0.162606	0.5	0.0288172	0.834805	0.798139	0.627199	0.883171	0.258206	0.626709	0.741794	0.224821	0.947621	0.262249
7	0.000179301	0.0115704	0.484646	0.947621	0.0456746	0.834805	0.484646	0.5	0.410055	0.5	0.0937197	0.000179301	0.0757193	0.99825	0.0150654
8	0.00446224	0.741794	0.875539	0.165195	0.249284	0.902817	0.595549	0.5	0.410055	0.5	0.19692	0.5	0.851383	0.995278	0.00446224
9	0.0115704	0.097183	0.282626	0.165195	0.930026	0.0115704	0.167184	0.627199	0.924076	0.0523787	0.709558	0.097183	0.689019	0.974212	0.326749
10	0.000588495	0.0523787	0.719747	0.000588495	0.176657	0.834805	0.099513	0.5	0.834308	0.5	0.255705	0.627199	0.186566	0.834805	0.55442
11	0.0401083	0.165195	0.230755	0.627199	0.890582	0.741794	0.585209	0.5	0.375313	0.165195	0.437274	0.0523787	0.678686	0.995278	0.0401083
12	0.000224353	0.834805	0.00462896	0.627199	0.262249	0.097183	0.262249	0.627199	0.55442	0.372801	0.000224353	0.372801	0.428095	0.947621	0.358641
13	0.00472212	0.0257879	0.078086	0.834805	0.428095	0.00472212	0.358641	0.165195	0.191687	0.0257879	0.20773	0.0523787	0.230755	0.947621	0.0117914
14	0.00472212	0.097183	0.465415	0.627199	0.504219	0.372801	0.162606	0.00472212	0.366918	0.5	0.0937197	0.627199	0.191687	0.974212	0.00987997
15	0.0375661	0.741794	0.739918	0.627199	0.375313	0.5	0.983453	0.165195	0.964295	0.372801	0.242986	0.165195	0.262249	0.99825	0.0375661
16	0.00920137	0.741794	0.657933	0.5	0.213309	0.741794	0.779188	0.258206	0.851383	0.834805	0.00920137	0.097183	0.911413	0.741794	0.0145499

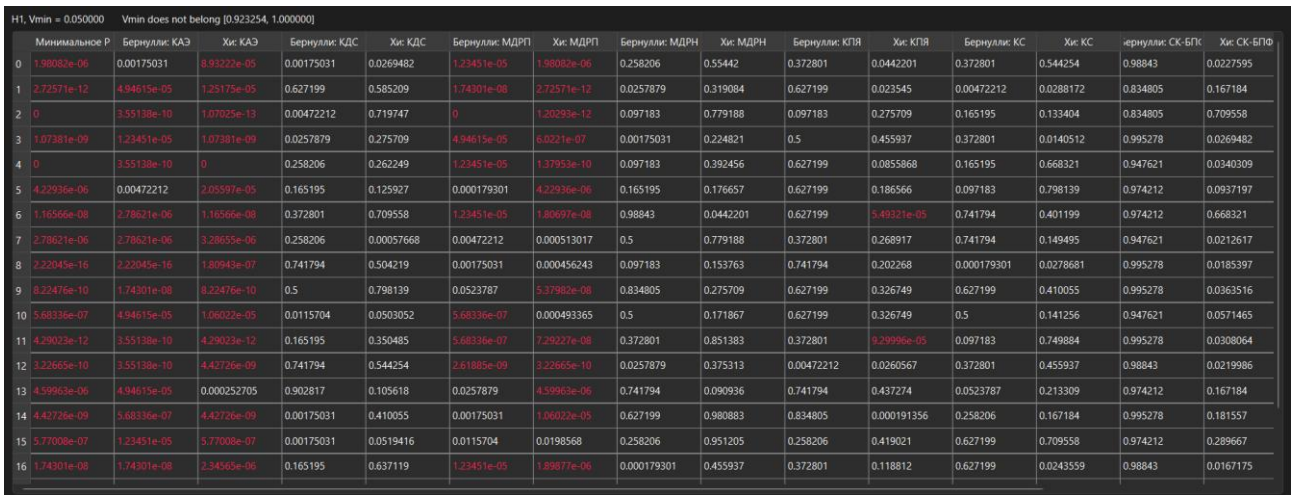
Рисунок 3.14 – Тестирование 2-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»



H1, Vmin = 0.500000 Vmin does not belong [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	3.55138e-10	3.55138e-10	0.00385138	0.947621	0.0167175	0.0257879	0.0278681	0.258206	0.275709	0.372801	0.0205479	4.94615e-05	0.00826628	0.995278	0.0471728
1	1.97292e-05	0.000588495	0.000438722	0.947621	0.383827	0.372801	1.97292e-05	0.834805	0.392456	0.834805	0.326749	0.741794	0.474986	0.99825	0.0329228
2	0.00255915	0.00472212	0.00255915	0.5	0.0198568	0.00472212	0.0102369	0.5	0.897763	0.258206	0.0308064	0.372801	0.181557	0.947621	0.474986
3	2.28179e-09	4.94615e-05	4.09661e-07	0.372801	0.514124	0.0257879	2.28179e-09	0.5	0.657933	0.5	0.162606	0.097183	0.534146	0.947621	0.0855868
4	3.96466e-05	0.00175031	0.0519416	0.372801	0.099513	0.000588495	3.96466e-05	0.0523787	0.719747	0.741794	0.0269482	0.258206	0.668321	0.902817	0.108791
5	7.91273e-05	0.0257879	0.00446224	0.902817	0.55442	0.0115704	7.91273e-05	0.902817	0.0297963	0.5	0.00182431	0.741794	0.078086	0.98843	0.904708
6	0.000320346	0.097183	0.102526	0.165195	0.167184	0.00175031	0.000320346	0.5	0.474986	0.5	0.137282	0.097183	0.153763	0.99825	0.00320057
7	4.34568e-11	0.00175031	0.0318483	0.741794	0.890582	4.34568e-11	2.79802e-08	0.258206	0.890582	0.372801	0.0340309	4.94615e-05	0.0269482	0.974212	0.534146
8	1.53982e-05	0.00472212	6.4622e-05	0.741794	0.867692	0.00175031	1.53982e-05	0.000588495	0.474986	0.5	0.00642485	0.627199	0.090936	0.98843	0.0648225
9	4.28546e-14	4.94615e-05	5.43905e-06	0.165195	0.890582	4.28546e-14	2.45433e-08	0.372801	0.759756	0.097183	0.366918	0.097183	0.419021	0.947621	0.230755
10	4.30203e-05	0.000588495	4.30203e-05	0.627199	0.964295	0.000588495	6.72964e-05	0.627199	0.544254	0.258206	0.153763	0.0523787	0.626709	0.995278	0.000756738
11	4.94615e-05	4.94615e-05	0.00666146	0.5	0.64753	0.00472212	0.000438722	0.000588495	0.141256	0.5	0.141256	0.372801	0.202268	0.995278	0.00953491
12	0.00120073	0.0523787	0.282626	0.902817	0.0937197	0.097183	0.00120073	0.5	0.334538	0.627199	0.153763	0.165195	0.534146	0.98843	0.0519416
13	1.74301e-08	5.68336e-07	8.25937e-06	0.834805	0.366918	1.74301e-08	2.24886e-08	0.0523787	0.275709	0.00175031	0.125927	0.165195	0.524101	0.974212	0.0260567
14	1.23451e-05	1.23451e-05	0.000648041	0.741794	0.375313	0.0257879	0.00182431	0.258206	0.749884	0.627199	0.0191876	0.627199	0.23681	0.995278	0.00320057
15	1.23451e-05	0.627199	0.00690641	0.627199	0.0689994	1.23451e-05	0.0757193	0.372801	0.544254	0.5	0.112047	0.372801	0.112047	0.947621	0.311542
16	0.000588495	0.000588495	0.00516588	0.00175031	0.224821	0.0115704	0.0648225	0.258206	0.668321	0.258206	0.668321	0.627199	0.282626	0.834805	0.153763

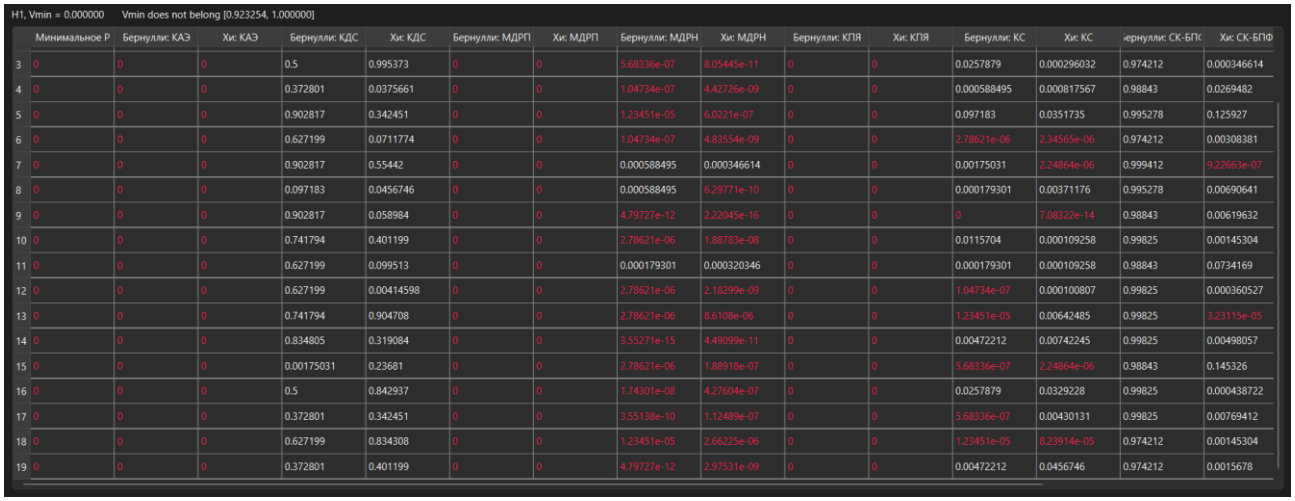
Рисунок 3.15 – Тестирование 3-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»



H1, Vmin = 0.050000 Vmin does not belong [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	1.98082e-06	0.00175031	8.93222e-05	0.00175031	0.0269482	1.23451e-05	1.98082e-06	0.258206	0.55442	0.372801	0.0442201	0.372801	0.544254	0.98843	0.0227595
1	2.72571e-12	4.94615e-05	1.25175e-05	0.627199	0.585209	1.74301e-08	2.72571e-12	0.0257879	0.319084	0.627199	0.023545	0.00472212	0.0288172	0.834805	0.167184
2	0	3.55138e-10	1.07025e-13	0.00472212	0.719747	0	1.20293e-12	0.097183	0.779188	0.097183	0.275709	0.165195	0.133404	0.834805	0.709558
3	1.07381e-09	1.23451e-05	1.07381e-09	0.0257879	0.275709	4.94615e-05	6.0221e-07	0.00175031	0.224821	0.5	0.455937	0.372801	0.0140512	0.995278	0.0269482
4	0	3.55138e-10	0	0.258206	0.262249	1.23451e-05	1.37953e-10	0.097183	0.392456	0.627199	0.0855868	0.165195	0.668321	0.947621	0.0340309
5	4.22936e-06	0.00472212	2.05597e-05	0.165195	0.125927	0.000179301	4.22936e-06	0.165195	0.176657	0.627199	0.186566	0.097183	0.798139	0.974212	0.0937197
6	1.16566e-08	2.78621e-06	1.16566e-08	0.372801	0.709558	1.23451e-05	1.80697e-08	0.98843	0.0442201	0.627199	5.49321e-05	0.741794	0.401199	0.974212	0.668321
7	2.78621e-06	2.78621e-06	3.28655e-06	0.258206	0.00057668	0.00472212	0.000513017	0.5	0.779188	0.372801	0.268917	0.741794	0.149495	0.947621	0.0212617
8	2.22045e-16	2.22045e-16	1.80943e-07	0.741794	0.504219	0.00175031	0.000456243	0.097183	0.153763	0.741794	0.202268	0.000179301	0.0278681	0.995278	0.0185397
9	8.22476e-10	1.74301e-08	8.22476e-10	0.5	0.798139	0.0523787	5.37982e-08	0.834805	0.275709	0.627199	0.326749	0.627199	0.410055	0.995278	0.0363516
10	5.68336e-07	4.94615e-05	1.06022e-05	0.0115704	0.0503052	5.68336e-07	0.000493365	0.5	0.171867	0.627199	0.326749	0.5	0.141256	0.947621	0.0571465
11	4.29023e-12	3.55138e-10	4.29023e-12	0.165195	0.350485	5.68336e-07	7.29227e-08	0.372801	0.851383	0.372801	9.29996e-05	0.097183	0.749884	0.995278	0.0308064
12	3.22665e-10	3.55138e-10	4.42726e-09	0.741794	0.544254	2.61885e-09	3.22665e-10	0.0257879	0.375313	0.00472212	0.0260567	0.372801	0.455937	0.98843	0.0219986
13	4.59963e-06	4.94615e-05	0.000252705	0.902817	0.105618	0.0257879	4.59963e-06	0.741794	0.090936	0.741794	0.437274	0.0523787	0.213309	0.974212	0.167184
14	4.42726e-09	5.68336e-07	4.42726e-09	0.00175031	0.410055	0.00175031	1.06022e-05	0.627199	0.980883	0.834805	0.000191356	0.258206	0.167184	0.995278	0.181557
15	5.77008e-07	1.23451e-05	5.77008e-07	0.00175031	0.0519416	0.0115704	0.0198568	0.258206	0.951205	0.258206	0.419021	0.627199	0.709558	0.974212	0.289667
16	1.74301e-08	1.74301e-08	2.34565e-06	0.165195	0.637119	1.23451e-05	1.89877e-06	0.000179301	0.455937	0.372801	0.118812	0.627199	0.0243559	0.98843	0.0167175

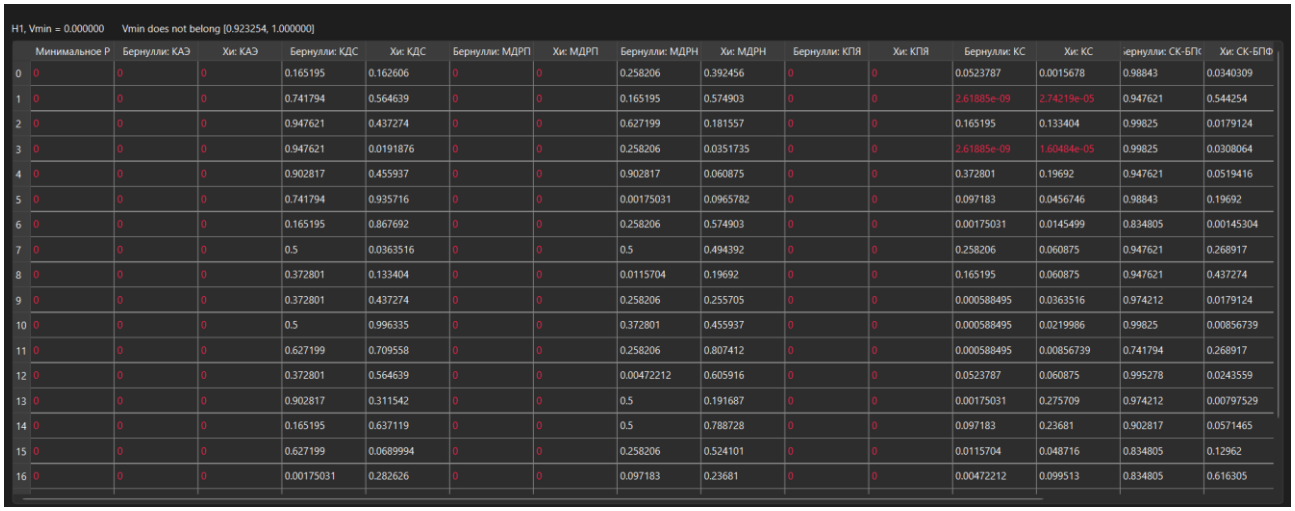
Рисунок 3.16 – Тестирование 4-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»



H1, Vmin = 0.000000 Vmin does not belong [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
3	0	0	0	0.5	0.995373	0	0	5.68336e-07	8.05445e-11	0	0	0.0257879	0.000296032	0.974212	0.000346614
4	0	0	0	0.372801	0.0375661	0	0	1.04734e-07	4.42726e-09	0	0	0.000588495	0.000817567	0.98843	0.0269482
5	0	0	0	0.902817	0.342451	0	0	1.23451e-05	6.0221e-07	0	0	0.097183	0.0351735	0.995278	0.125927
6	0	0	0	0.627199	0.0711774	0	0	1.04734e-07	4.83554e-09	0	0	2.78621e-06	2.34565e-06	0.974212	0.00308381
7	0	0	0	0.902817	0.55442	0	0	0.000588495	0.000346614	0	0	0.00175031	2.24864e-06	0.999412	9.22663e-07
8	0	0	0	0.097183	0.0456746	0	0	0.000588495	6.29771e-10	0	0	0.000179301	0.00371176	0.995278	0.00690641
9	0	0	0	0.902817	0.058984	0	0	4.79727e-12	2.22045e-16	0	0	0	7.08322e-14	0.98843	0.00619632
10	0	0	0	0.741794	0.401199	0	0	2.78621e-06	1.88783e-08	0	0	0.0115704	0.000109258	0.99825	0.00145304
11	0	0	0	0.627199	0.099513	0	0	0.000179301	0.000320346	0	0	0.000179301	0.000109258	0.98843	0.0734169
12	0	0	0	0.627199	0.00414598	0	0	2.78621e-06	2.18299e-09	0	0	1.04734e-07	0.000100807	0.99825	0.000360527
13	0	0	0	0.741794	0.904708	0	0	2.78621e-06	8.6108e-06	0	0	1.23451e-05	0.00642485	0.99825	3.23115e-05
14	0	0	0	0.834805	0.319084	0	0	3.55271e-15	4.49099e-11	0	0	0.00472212	0.00742245	0.99825	0.00498057
15	0	0	0	0.00175031	0.23681	0	0	2.78621e-06	1.88918e-07	0	0	5.68336e-07	2.24864e-06	0.98843	0.145326
16	0	0	0	0.5	0.842937	0	0	1.74301e-08	4.27604e-07	0	0	0.0257879	0.0329228	0.99825	0.000438722
17	0	0	0	0.372801	0.342451	0	0	3.55138e-10	1.12489e-07	0	0	5.68336e-07	0.00430131	0.99825	0.00769412
18	0	0	0	0.627199	0.834308	0	0	1.23451e-05	2.66225e-06	0	0	1.23451e-05	8.23914e-05	0.974212	0.00145304
19	0	0	0	0.372801	0.401199	0	0	4.79727e-12	2.97531e-09	0	0	0.00472212	0.0456746	0.974212	0.0015678

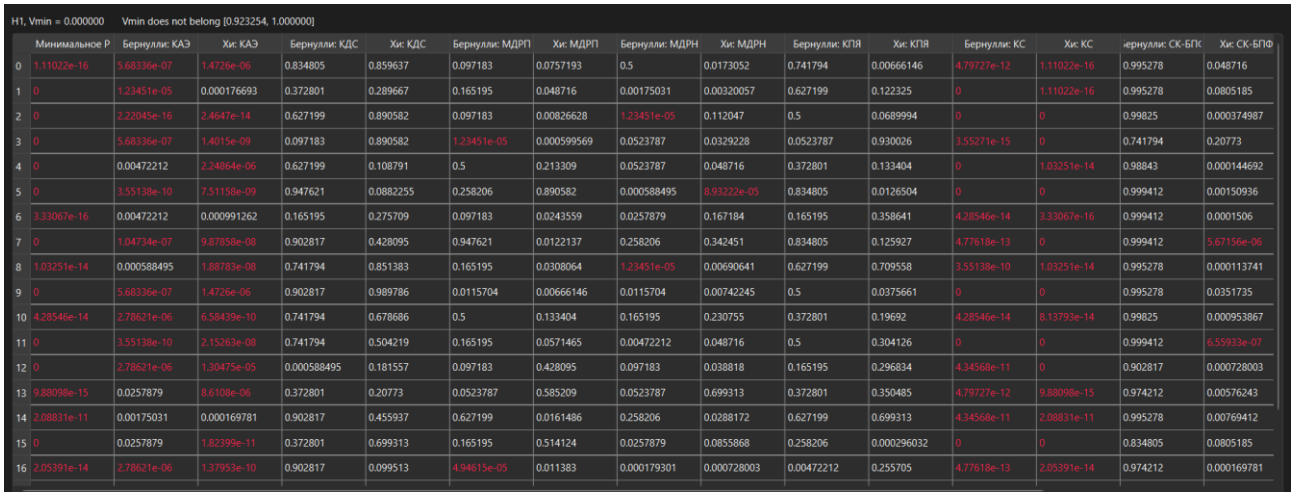
Рисунок 3.17 – Тестирование 5-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»



H1, Vmin = 0.000000 Vmin does not belong [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	0	0	0	0.165195	0.162606	0	0	0.258206	0.392456	0	0	0.0523787	0.0015678	0.98843	0.0340309
1	0	0	0	0.741794	0.564639	0	0	0.165195	0.574903	0	0	2.61885e-09	2.74219e-05	0.947621	0.544254
2	0	0	0	0.947621	0.437274	0	0	0.627199	0.181557	0	0	0.165195	0.133404	0.99825	0.0179124
3	0	0	0	0.947621	0.0191876	0	0	0.258206	0.0351735	0	0	2.61885e-09	1.60484e-05	0.99825	0.0308064
4	0	0	0	0.902817	0.455937	0	0	0.902817	0.060875	0	0	0.372801	0.19692	0.947621	0.0519416
5	0	0	0	0.741794	0.935716	0	0	0.00175031	0.0965782	0	0	0.097183	0.0456746	0.98843	0.19692
6	0	0	0	0.165195	0.867692	0	0	0.258206	0.574903	0	0	0.00175031	0.0145499	0.834805	0.00145304
7	0	0	0	0.5	0.0363516	0	0	0.5	0.494392	0	0	0.258206	0.060875	0.947621	0.268917
8	0	0	0	0.372801	0.133404	0	0	0.0115704	0.19692	0	0	0.165195	0.060875	0.947621	0.437274
9	0	0	0	0.372801	0.437274	0	0	0.258206	0.255705	0	0	0.000588495	0.0363516	0.974212	0.0179124
10	0	0	0	0.5	0.996335	0	0	0.372801	0.455937	0	0	0.000588495	0.0219986	0.99825	0.00856739
11	0	0	0	0.627199	0.709558	0	0	0.258206	0.807412	0	0	0.000588495	0.00856739	0.741794	0.268917
12	0	0	0	0.372801	0.564639	0	0	0.00472212	0.605916	0	0	0.0523787	0.060875	0.995278	0.0243559
13	0	0	0	0.902817	0.311542	0	0	0.5	0.191687	0	0	0.00175031	0.275709	0.974212	0.00797529
14	0	0	0	0.165195	0.637119	0	0	0.5	0.788728	0	0	0.097183	0.23681	0.902817	0.0571465
15	0	0	0	0.627199	0.0689994	0	0	0.258206	0.524101	0	0	0.0115704	0.048716	0.834805	0.12962
16	0	0	0	0.00175031	0.282626	0	0	0.097183	0.23681	0	0	0.00472212	0.099513	0.834805	0.616305

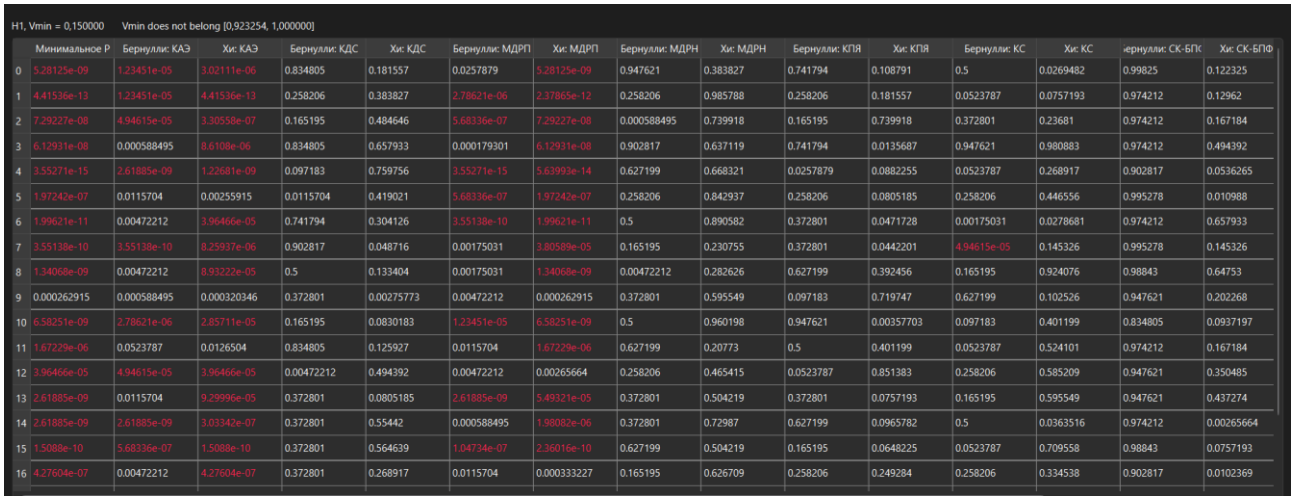
Рисунок 3.18 – Тестирование 6-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»



H1, Vmin = 0.000000 Vmin does not belong [0.923254, 1.000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	1.11022e-16	5.68336e-07	1.4726e-06	0.834805	0.859637	0.097183	0.0757193	0.5	0.0173052	0.741794	0.00666146	4.79727e-12	1.11022e-16	0.995278	0.048716
1	0	1.23451e-05	0.000176693	0.372801	0.289667	0.165195	0.048716	0.00175031	0.00320057	0.627199	0.122325	0	1.11022e-16	0.995278	0.0805185
2	0	2.22045e-16	2.4647e-14	0.627199	0.890582	0.097183	0.00826628	1.23451e-05	0.112047	0.5	0.0689994	0	0	0.99825	0.000374987
3	0	5.68336e-07	1.4015e-09	0.097183	0.890582	1.23451e-05	0.000599569	0.0523787	0.0329228	0.0523787	0.930026	3.55271e-15	0	0.741794	0.20773
4	0	0.00472212	2.24864e-06	0.627199	0.108791	0.5	0.213309	0.0523787	0.048716	0.372801	0.133404	0	1.03251e-14	0.98843	0.000144692
5	0	3.55138e-10	7.51158e-09	0.947621	0.0882255	0.258206	0.890582	0.000588495	8.93222e-05	0.834805	0.0126504	0	0	0.999412	0.00150936
6	3.33067e-16	0.00472212	0.000991262	0.165195	0.275709	0.097183	0.0243559	0.0257879	0.167184	0.165195	0.358641	4.28546e-14	3.33067e-16	0.999412	0.0001506
7	0	1.04734e-07	9.87858e-08	0.902817	0.428095	0.947621	0.0122137	0.258206	0.342451	0.834805	0.125927	4.77618e-13	0	0.999412	5.67156e-06
8	1.03251e-14	0.000588495	1.88783e-08	0.741794	0.851383	0.165195	0.0308064	1.23451e-05	0.00690641	0.627199	0.709558	3.55138e-10	1.03251e-14	0.995278	0.000113741
9	0	5.68336e-07	1.4726e-06	0.902817	0.989786	0.0115704	0.00666146	0.0115704	0.00742245	0.5	0.0375661	0	0	0.995278	0.0351735
10	4.28546e-14	2.78621e-06	6.58439e-10	0.741794	0.678686	0.5	0.133404	0.165195	0.230755	0.372801	0.19692	4.28546e-14	8.13793e-14	0.99825	0.000953867
11	0	3.55138e-10	2.15263e-08	0.741794	0.504219	0.165195	0.0571465	0.00472212	0.048716	0.5	0.304126	0	0	0.999412	6.55933e-07
12	0	2.78621e-06	1.30475e-05	0.000588495	0.181557	0.097183	0.428095	0.097183	0.038818	0.165195	0.296834	4.34568e-11	0	0.902817	0.000728003
13	9.88098e-15	0.0257879	8.6108e-06	0.372801	0.20773	0.0523787	0.585209	0.0523787	0.699313	0.372801	0.350485	4.79727e-12	9.88098e-15	0.974212	0.00576243
14	2.08831e-11	0.00175031	0.000169781	0.902817	0.455937	0.627199	0.0161486	0.258206	0.0288172	0.627199	0.699313	4.34568e-11	2.08831e-11	0.995278	0.00769412
15	0	0.0257879	1.82399e-11	0.372801	0.699313	0.165195	0.514124	0.0257879	0.0855868	0.258206	0.000296032	0	0	0.834805	0.0805185
16	2.05391e-14	2.78621e-06	1.37953e-10	0.902817	0.099513	4.94615e-05	0.011383	0.000179301	0.000728003	0.00472212	0.255705	4.77618e-13	2.05391e-14	0.974212	0.000169781

Рисунок 3.19 – Тестирование 7-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»



H1, Vmin = 0,150000 Vmin does not belong [0,923254, 1,000000]

	Минимальное P	Бернулли: КАЭ	Хи: КАЭ	Бернулли: КДС	Хи: КДС	Бернулли: МДРП	Хи: МДРП	Бернулли: МДРН	Хи: МДРН	Бернулли: КПЯ	Хи: КПЯ	Бернулли: КС	Хи: КС	ернулли: СК-БПС	Хи: СК-БПФ
0	5.28125e-09	1.23451e-05	3.02111e-06	0.834805	0.181557	0.0257879	5.28125e-09	0.947621	0.383827	0.741794	0.108791	0.5	0.0269482	0.99825	0.122325
1	4.41536e-13	1.23451e-05	4.41536e-13	0.258206	0.383827	2.78621e-06	2.37865e-12	0.258206	0.985788	0.258206	0.181557	0.0523787	0.0757193	0.974212	0.12962
2	7.29227e-08	4.94615e-05	3.30558e-07	0.165195	0.484646	5.68336e-07	7.29227e-08	0.000588495	0.739918	0.165195	0.739918	0.372801	0.23681	0.974212	0.167184
3	6.12931e-08	0.000588495	8.6108e-06	0.834805	0.657933	0.000179301	6.12931e-08	0.902817	0.637119	0.741794	0.0135687	0.947621	0.980883	0.974212	0.494392
4	3.55271e-15	2.61885e-09	1.22681e-09	0.097183	0.759756	3.55271e-15	5.63993e-14	0.627199	0.668321	0.0257879	0.0882255	0.0523787	0.268917	0.902817	0.0536265
5	1.97242e-07	0.0115704	0.00255915	0.0115704	0.419021	5.68336e-07	1.97242e-07	0.258206	0.842937	0.258206	0.0805185	0.258206	0.446556	0.995278	0.010988
6	1.99621e-11	0.00472212	3.96466e-05	0.741794	0.304126	3.55138e-10	1.99621e-11	0.5	0.890582	0.372801	0.0471728	0.00175031	0.0278681	0.974212	0.657933
7	3.55138e-10	3.55138e-10	8.25937e-06	0.902817	0.048716	0.00175031	3.80589e-05	0.165195	0.230755	0.372801	0.0442201	4.94615e-05	0.145326	0.995278	0.145326
8	1.34068e-09	0.00472212	8.93222e-05	0.5	0.133404	0.00175031	1.34068e-09	0.00472212	0.282626	0.627199	0.392456	0.165195	0.924076	0.98843	0.64753
9	0.000262915	0.000588495	0.000320346	0.372801	0.00275773	0.00472212	0.000262915	0.372801	0.595549	0.097183	0.719747	0.627199	0.102526	0.947621	0.202268
10	6.58251e-09	2.78621e-06	2.85711e-05	0.165195	0.0830183	1.23451e-05	6.58251e-09	0.5	0.960198	0.947621	0.00357703	0.097183	0.401199	0.834805	0.0937197
11	1.67229e-06	0.0523787	0.0126504	0.834805	0.125927	0.0115704	1.67229e-06	0.627199	0.20773	0.5	0.401199	0.0523787	0.524101	0.974212	0.167184
12	3.96466e-05	4.94615e-05	3.96466e-05	0.00472212	0.494392	0.00472212	0.00265664	0.258206	0.465415	0.0523787	0.851383	0.258206	0.585209	0.947621	0.350485
13	2.61885e-09	0.0115704	9.29996e-05	0.372801	0.0805185	2.61885e-09	5.49321e-05	0.372801	0.504219	0.372801	0.0757193	0.165195	0.595549	0.947621	0.437274
14	2.61885e-09	2.61885e-09	3.03342e-07	0.372801	0.55442	0.000588495	1.98082e-06	0.372801	0.72987	0.627199	0.0965782	0.5	0.0363516	0.974212	0.00265664
15	1.5088e-10	5.68336e-07	1.5088e-10	0.372801	0.564639	1.04734e-07	2.36016e-10	0.627199	0.504219	0.165195	0.0648225	0.0523787	0.709558	0.98843	0.0757193
16	4.27604e-07	0.00472212	4.27604e-07	0.372801	0.268917	0.0115704	0.000333227	0.165195	0.626709	0.258206	0.249284	0.258206	0.334538	0.902817	0.0102369

Рисунок 3.20 – Тестирование 8-й модификации последовательности чисел, сгенерированной ФГСЧ «Ключ-ВС»

В результате проведенного тестирования последовательностей, выработанных генератором случайных чисел с искусственно добавленными ошибками, была подтверждена высокая эффективность разработанного программного комплекса в обнаружении сбоев и аномалий в выходных данных.

Полученные результаты демонстрируют, что разработанное программное обеспечение соответствует требованиям СТБ 34.101.27 и может быть использовано для проверки качества выходных последовательностей физических генераторов случайных чисел. Таким образом, проведенные эксперименты подтвердили корректность реализации алгоритмов статистического тестирования и практическую применимость комплекса для задач анализа и верификации генераторов случайных чисел.

ЗАКЛЮЧЕНИЕ

Основные результаты работы:

1. Изучена методика МИ.10127.10.03. Выявлены неточности и опечатки в документе методики.
2. Разработан программный комплекс с многопоточным режимом работы, в котором реализованы алгоритмы МДРН, МДРП, КПЯ, КС, КДС, АЭ, КСКП, СК-БПФ, батарей Бернулли и Хи-квадрат, ТОП, ТНП.
3. В разработанном программном комплексе реализована общая логика разбора входных параметров и запуска процедуры тестирования с соответствующими параметрами; графический интерфейс со следующими элементами: кнопки «Старт», «Стоп», «Обзор», индикатор прогресса, таймер от начала тестирования файла.

Выводы, которые могут быть сделаны по результатам работы:

1. Были изучены основные статистические тесты, такие как тесты из батареи Бернулли и батареи Хи-квадрат, а также требования СТБ 34.101.27, предъявляемые к генераторам случайных чисел.
2. Установлено, что алгоритмы методики МИ.10127.10.03 позволяют эффективно оценить ключевые параметры генераторов: равновероятность, независимость и однородность наблюдаемых случайных чисел, отсутствие в них периодичности.
3. В ходе выполнения работы была достигнута основная цель – выполнена реализация алгоритмов из МИ.10127.10.03 для оценки качества выходных последовательностей генераторов случайных чисел.
4. Программный комплекс полностью соответствует требованиям СТБ 34.101.27 и подходит для тестирования качества выходных последовательностей ФГСЧ.
5. Использование многопоточного режима работы позволяет значительно сократить время анализа больших объемов данных без потери точности и достоверности результатов.

Разработанный программный комплекс был опубликован в открытом доступе на ресурсе GitHub, что позволит получить всем желающим проводить тестирование выходных последовательностей генераторов случайных чисел по методике МИ.10127.10.03. В будущем программный комплекс может быть доработан за счёт применения других батарей тестов, а также оптимизации скорости работы реализации.

Реализованный программный комплекс успешно применяется в производственном процессе Государственного предприятия «НИИ ТЗИ», что отражено в прилагаемом акте о внедрении.

Список используемой литературы

1. СТБ 34.101.27: [Электронный ресурс] - https://apmi.bsu.by/assets/files/std/skzi-spec35.pdf – получен доступ: 21.05.2025
2. Методика статистического тестирования выходных последовательностей генераторов случайных чисел: [Электронный ресурс] - https://www.oac.gov.by/public/content/files/files/met-10127-10-03.pdf – Дата доступа: 16.05.2025
3. Справочник по стандартной библиотеке C++: [Электронный ресурс] - https://learn.microsoft.com/ru-ru/cpp/standard-library/cmath?view=msvc-170 – Дата доступа: 09.05.2025
4. NIST Special Publication 800-22 Rev 1a: [Электронный ресурс] - https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic – получен доступ: 26.05.2025.
5. Вычисление гамма-функции на языке C++ с помощью алгоритма asa032: [Электронный ресурс] - https://people.sc.fsu.edu/~jburkardt/cpp_src/asa032/asa032.html – Дата доступа: 11.03.2025
6. Гамма-функция: [Электронный ресурс] - https://ru.wikipedia.org/wiki/Гамма-функция – получен доступ: 14.03.2025
7. Распределение Хи-квадрат: [Электронный ресурс] - https://ru.wikipedia.org/wiki/Распределение_хи-квадрат – Дата доступа: 16.05.2025
8. The NIST Statistical Test Suite: [Электронный ресурс] - https://github.com/terrillmoore/NIST-Statistical-Test-Suite/tree/master. - Дата доступа: 22.05.2025.
9. Qt документация: [Электронный ресурс] - https://doc.qt.io/ – Дата доступа: 18.04.2025

ПРИЛОЖЕНИЕ А

Код теста МДРН на C++

```
#pragma once

#include "2igamc.h"
#include <cmath>
#include <vector>
#include <string>

double mdu(std::string &x) {
      int l = 1, m = x.length(), u = 2, k;
      double svalue = 0, m2_l = 1.0*m/u;
      while (m >= 50 && m2_l >= 10) {
            l++;
            m = x.length() / l;
            u *= 2;
            m2_l = (double)m / u;
      }
      --l;
      m = x.length() / l;
      u /= 2;
      m2_l = (double)m / u;

    std::string sub;
    std::vector<int> vec(u);
      for (int t = 0; t < m; t++) {

            k = 0;
            for (int j = 0; j < l; j++) {
                  k <<= 1;
                  if (x[t*l + j] == '1')
                        k++;
            }
            vec[k]++;

      }
      for (int i = 0; i < u; ++i) {
            svalue += pow(m2_l - vec[i], 2);
      }

    svalue /= m2_l;
      return 1 - gamain(svalue/2, 0.5*(u - 1), 1);
}
```

ПРИЛОЖЕНИЕ Б

Код теста МДРП на C++

```
#pragma once

#include "2igamc.h"
#include <cmath>
#include <vector>
#include <string>

double mdu2(std::string &x) {
      int l = 1;
      while (l <= log2(0.05 * x.length())) {
            ++l;
      }
      --l;
      int u = pow(2, l), u2 = u / 2, k;

    if (x.length() < 20 * u) { return -1; }

    std::vector<int> vec(u), vec1(u2);
    std::string sub;
      double svalue = 0, gvalue = 0,gvalue2 = 0, n2_l = 1.0 * x.length() / u,
n2_l1 = 1.0 * x.length() / u2;

      x = x + x.substr(0, l);

      if (x.length() < 20 * u) { return -1; }
      for (int t = 0; t < x.length()-l+1; t++) {
            k = 0;
            for (int j = 0; j < l; j++) {
                  k <<= 1;
                  if (x[t + j] == '1')
                        k++;
            }
            vec[k]++;

            k = 0;
            for (int j = 0; j < l - 1; j++) {
                  k <<= 1;
                  if (x[t + j] == '1')
                        k++;
            }
            vec1[k]++;
      }


      for (int i = 0; i < u; i++) {
            gvalue += pow(n2_l - vec[i], 2);
      }
      gvalue /= n2_l;

      for (int i = 0; i < u2; i++) {
            gvalue2 += pow(n2_l1 - vec1[i], 2);
      }
      gvalue2 /= n2_l1;

      svalue = gvalue - gvalue2;

      x.erase(x.length() - l, l);
      return 1 - gamain(svalue/2, 0.5*u2, 1);
}
```

ПРИЛОЖЕНИЕ В

Код теста ПЯ на C++

```
#pragma once

#include "2igamc.h"
#include "normal.h"
#include <vector>
#include <string>
#include <cmath>

double empbo(int l, std::string &x) {

      double u = pow(2, l), p = -1, e = M_E;
      int m = x.length() / l;
      int nu0 = 0, k;
      double lam, lamp, nu, sig2;
    std::vector<int> vec(u);

      for (int t = 0; t < m; t++) {
            k = 0;
            for (int j = 0; j < l; j++) {
                  k <<= 1;
                  if (x[l*t + j] == '1')
                        k++;
            }
            vec[k]++;
      }
      for (int i = 0; i<u; ++i){
            if (vec[i] == 0) nu0++;
      }
      double s = m - (u-nu0);
      lam = 1.0 * m / u;
      lamp = lam * m / 2;
      nu = pow(e, - lam) * u;
      sig2 = nu * (1.0 - (1.0 + lam)*pow(e, -lam));
      if (lam <= 1.0 / 32) {
            p = gamain(lamp, s, 1);
      }
      else if (lam > 1.0 / 32 && lam < 5.0) {
            p = norm_fun(-((u-nu0)-nu) / sqrt(sig2),0,1);
      }
      return p;
}
```

ПРИЛОЖЕНИЕ Г

Код теста серий на C++

```
#pragma once

#include "2igamc.h"
#include <cmath>
#include <vector>
#include <string>
#include <map>

double series(std::string &x) {
    std::string exam = "";
    std::vector<double> nui;
      int k = 0;
    map<std::string, int> v0, v1;
      double nu = 10, s = 0;

      while(nu >= 5) {
            k++;
            nu = (x.length() - k + 3) / pow(2, k + 2);
            nui.push_back(nu);
      }
      k--;
      nui.pop_back();

      for (int i = 0; i < k; i++) {
            exam += '0';
      }
      for (int i = 0; i < k; i++) {
            exam += '1';
      }

      size_t cur = 0, next = 0;
    while (next != std::string::npos) {
            if (x[cur] == '0') {
                  next = x.find('1', cur);
                  ++v0[x.substr(cur, next - cur)];
            }
            else if (x[cur] == '1') {
                  next = x.find('0', cur);
                  ++v1[x.substr(cur, next - cur)];
            }
            cur = next;
      }
      for (int i = 0; i < k; i++) {
      //    cout << nui[i] << ' ' << v0[exam.substr(0, i + 1)] << ' ' <<
v1[exam.substr(k, i+1)] << endl;
            s += (pow(v0[exam.substr(0, i + 1)] - nui[i], 2) +
(pow(v1[exam.substr(k, i + 1)] - nui[i], 2)) )/ nui[i];
            }
      return 1-gamain(s/2, k-1, 1);
}
```

ПРИЛОЖЕНИЕ Д

Код теста длинных серий на C++

```
#pragma once

#include "2igamc.h"
#include <cmath>
#include <vector>
#include <string>

double lseries(std::string& x) {
      int l = 0;
      if (x.length() < 149000) l = 128;
      else l = 10000;

      int ij, k;
      if (l == 8) {
            ij = 0;
            k = 3;
      }
      else if (l == 128) {
            ij = 4;
            k = 5;
      }
      else if (l == 512) {
            ij = 10;
            k = 5;
      }
      else if (l == 1000) {
            ij = 16;
            k = 5;
      }
      else if (l == 10000) {
            ij = 22;
            k = 6;
      }
      else { return -1; }

      double pi[29] = { 0.2148, 0.3672, 0.2305, 0.1875 ,
                        0.1174, 0.2430, 0.2493, 0.1752, 0.1027, 0.1124,
                        0.1170, 0.2460, 0.2523, 0.1755, 0.1015, 0.1077 ,
                        0.1307, 0.2437, 0.2452, 0.1714, 0.1002, 0.1088 ,
                        0.0822, 0.2092, 0.2483, 0.1933, 0.1208, 0.0675, 0.0727
};

      int d[29] = { 1, 2, 3, 4,
                        4, 5, 6, 7, 8, 9,
                        6, 7, 8, 9, 10, 11,
                        7, 8, 9, 10, 11, 12,
                        10, 11, 12, 13, 14, 15, 16};

      double s = 0;
    std::vector<int> b, v;
      int m = x.length()/l;
      size_t cur = 0, next = 0;
      for (int i = 0; i < m; i++) {
            b.push_back(0);
      }

      for (int i = 0; i < k+1; i++) {
            v.push_back(0);
      }
```

```
	for (int i = 0; i < m; i++) {
		cur = i*l;
		next = i*l;
		while (next < i*l + l) {

		cur = next;

		if (x[cur] == '0') {
			next = x.find('1', cur);
			cur = next;
		}
		else if (x[cur] == '1') {
			next = x.find('0', cur);
			if (next > i * l + l) {
				next = i * l + l;
			}
		b[i] = std::max(b[i], int(next - cur));
		}
	}
	}

	for (int i = 0; i < m; i++) {
		if (b[i] <= d[ij]) {
			v[0]++;
		}
		else if (b[i] >= d[ij + k]) {
			v[k]++;
		}
		else
			for (int j = 1; j < k; j++) {
				if (b[i] == d[ij + j]) {
				v[j]++;
			}
		}
	}

	for (int i = 0; i < k+1; i++) {
		s += pow(v[i] - m * pi[ij + i], 2)/(m*pi[ij+i]);
	}
	return 1 - gamain(0.5*s, 0.5*k, 1);
}
```

ПРИЛОЖЕНИЕ Е

Код теста аппроксимации энтропии на C++

```
#pragma once

#include "2igamc.h"
#include <cmath>
#include <vector>
#include <string>

double appent(std::string& x) {
    int l = 1, u = 2, n = x.length(), k = 0;
      double f = 0, f1 = 0, apen = 0, s = 0;


      while (u < 0.01 * n) {
            ++l;
            u *= 2;
      }
      --l;
      u /= 2;
      x += x.substr(0, l-1);
      n = x.length();

    std::vector<int> vec(u), vec1(0.5*u);

      for (int i = 0; i <= n - l; i++) {
            k = 0;
            for (int j = 0; j < l; j++) {
            k <<= 1;
            if (x[i+j] == '1')
                  k++;
            }
            vec[k]++;
            k = 0;
            for (int j = 0; j < l-1; j++) {
                  k <<= 1;
                  if (x[i + j] == '1')
                        k++;
            }
            vec1[k]++;
      }
      for (int i = 0; i < u; i++) {

            if (vec[i] != 0) {
                  f -= log(1.0 * vec[i] / (n + 1 - l)) * vec[i];
            }
      }

      for (int i = 0; i < 0.5*u; i++) {
            if (vec1[i] != 0) {
                  f1 -= log(1.0 * vec1[i] / (n + 1 - l)) * vec1[i];
            }
      }
      f /= (n + 1 - l);
      f1 /= (n + 1 - l);
      apen = f - f1;

      s = 2 * (n-l+1) * fabs(log(2) - apen);
      return 1 - gamain(s/2, 0.25*u, 1);
}
```

ПРИЛОЖЕНИЕ Ж

Код скалярного теста на C++

```
#pragma once

#include "2igamc.h"
#include <cmath>
#include <vector>
#include <string>

double comb(int l, int m) {
      if (l > m) return 0;
    int r = std::max(l, m - l);
      if (r == m) { return 1; }
      double res = 1;
      for (int i = 0; i < m - r; i++) {
            res *= m - i;
            res /= i + 1;
      }
      return res;
}

double scapro(int m, int K, std::string& x) {
    std::vector<int> fk, ymax;
    std::vector<double> qk;
      int l = m * K, M = x.length()/l;
    double y = 0, sum1, sum2, s = 0, left = 0, right = 0;
      for (int p = 0; p < m+1; p++) {
            fk.push_back(0);
            qk.push_back(0);
      }
      for (int p = 0; p < M; p++) {
            ymax.push_back(0);
      }
      qk[0] = pow(2, -m) * pow(1 + pow(2, 1 - K), m);


      for (int k = 1; k <= m; k++) {
            for (int i = 0; i <= m; i++) {
                  sum1 = 0;
                  sum2 = 0;
                  for (int j = 0; j < k; j++) {
                        sum2 += comb(j, i);
                  }
                  sum1 = sum2 + comb(k, i);
                  left += comb(i, m) * (pow(pow(2, -i)*sum1, K - 1));
                  right += comb(i, m) * (pow(pow(2, -i)*sum2, K - 1));
            }
            qk[k] = pow(2, -m) * (left - right);
            left = 0;
            right = 0;
      }

      double qmin = qk[0];
      for (int k = 0; k <= m; k++) {
        qmin = fmin(qmin, qk[k]);
      }
    if (1.0 * K / qmin > 0.1 * x.length() / m) { return -1; }
    if (m < 2 || m > 8) { return -1; }
    if (K < 2 || K > 11) { return -1; }

      for (int i = 0; i < M; i++) {
```

```
            for (int j = 1; j < K; j++) {
                  y = 0;
                  for (int p = 0; p < m; p++) {
                        y += (x[i * l + p] - '0') * (x[i * l + j * m + p] -
'0');
                  }
                  if (y > ymax[i]) ymax[i] = y;
        }
      }

      for (int i = 0; i < M; i++) {
        ++fk[ymax[i]];
      }
    for (int i = 0; i < m+1; i++) {
      }


      for (int k = 0; k <= m; k++) {
        s += pow(fk[k] - M * qk[k], 2)/(M*qk[k]);
    }

      return 1 - gamain(s/2, 0.5*m, 1);
}
```

ПРИЛОЖЕНИЕ И

Код спектрального теста на C++

```
#pragma once

#include <string>
#include "dfft.h"
#include "normal.h"

void  __ogg_fdrffti(int n, double* wsave, int* ifac);
void  __ogg_fdrfftf(int n, double* X, double* wsave, int* ifac);

double spect(std::string& x) {
    double  p_value, upperBound, N_l, N_o, d, * m = NULL, * X = NULL, * wsave =
NULL;
      int         i, count, ifac[15], n = x.length();

      try {
            X = new double[n]();
            wsave = new double[2 * n]();
            m = new double[n / 2 + 1]();
      }
    catch (const std::bad_alloc& e) {
            delete[] X;
            delete[] wsave;
            delete[] m;
            return -1;
      }
      for (i = 0; i < n; i++)
        X[i] = 2 * (x[i]-'0') - 1;

      __ogg_fdrffti(n, wsave, ifac);
      __ogg_fdrfftf(n, X, wsave, ifac);

      m[0] = sqrt(X[0] * X[0]);

      for (i = 0; i < n / 2; i++)
            m[i + 1] = sqrt(pow(X[2 * i + 1], 2) + pow(X[2 * i + 2], 2));
      count = 0;
      upperBound = sqrt(3 * n);
      for (i = 0; i < n / 2; i++) {
            if (m[i] < upperBound)
                  count++;
      }
      N_l = (double)count;       */
      N_o = (double)0.95 * n / 2.0;
      d = sqrt(N_o*0.05);

      p_value = 2*(1 - norm_fun(fabs(N_l - N_o)/d, 0, 1));


      delete[] X;
      delete[] wsave;
      delete[] m;

      return p_value;
}
```

ПРИЛОЖЕНИЕ К

Код батареи Бернулли на C++

```
#pragma once

#include "normal.h"
#include <cmath>
#include <vector>

void ber(std::vector<std::vector <double> > p, double a, std::vector<std::vector
<double> >&res) {
      int q = p.size(), m = p[0].size();
    std::vector<int> v(q);

      for (int i = 0; i < q; ++i) {
            for (int j = 0; j < m; ++j) {
                  if (p[i][j] > a) ++v[i];
            }
      }

      for (int i = 0; i < q; ++i) {
            res[0][i] = 1 - norm_fun(-((double)v[i] / m - (1 - a)) / sqrt(a * (1
- a) / m), 0, 1);
      }
}
```

ПРИЛОЖЕНИЕ Л

Код батареи Хи-квадрат на C++

```
#pragma once


#include "2igamc.h"
#include <cmath>
#include <vector>

void chi(std::vector<std::vector <double> > p, double k, std::vector<std::vector
<double> >& res) {
      double q = p.size(), m = p[0].size();
    std::vector<std::vector <int> > v(q, std::vector<int>(k));
    std::vector<double> stat(q);

      for (int i = 0; i < q; ++i) {
            for (int j = 0; j < m; ++j) {
                  for (int t = 0; t < k; ++t) {
                        if (p[i][j] < (1.0 + t) / k) {
                              ++v[i][t];
                              break;
                        }
                  }
                  if (p[i][j] == 1) ++v[i][k - 1];
            }
      }

      for (int i = 0; i < q; ++i) {
            for (int j = 0; j < k; ++j) {
                  stat[i] += pow(v[i][j] - m / k, 2);
            }
            stat[i] /= m / k;
      }

      for (int i = 0; i < q; ++i) {
            res[1][i] = 1 - gamain(stat[i]/2, 0.5*(k-1 ), 1);
      }
}
```

ПРИЛОЖЕНИЕ М

Код тестирования одной последовательности на C++

```
#pragma once

#include <QProgressBar>
#include <QMetaObject>
#include <QDebug>
#include <iomanip>
#include <vector>
#include <string>
#include <fstream>
#include "mduonis.h"
#include "mduois.h"
#include "empbo.h"
#include "series.h"
#include "lseries.h"
#include "appent.h"
#include "scapro.h"
#include "spect.h"
#include "ber.h"
#include "chi.h"
#include "testing.h"

void updateProgressCallback(QProgressBar* progressBar, int progressValue) {
    QMetaObject::invokeMethod(progressBar, "setValue", Qt::QueuedConnection,
                              Q_ARG(int, progressValue));
}

extern int fullOutputFlag;
extern unsigned long long int inFileSize;
extern double progress;
extern QString result;

void test_one(std::ofstream& out, std::string& bitString, int m,
              int k, double a, double b, std::vector<std::vector <double>>& res,
TestLogic* logicObj, QProgressBar* progressBar) {

    double iterationProgress = (double)bitString.length() / inFileSize * 100 /
m;
    double lastEmittedProgress = 0;
    std::string test = "";
    double pvalue1 = 0, pmin = 1, bmin = 1, cmin = 1;
    int ll;
    if (bitString.length() / m < 100000) ll = 18;
    else ll = 22;
    std::vector <std::vector <double>> pvalue(8, std::vector<double>());

    int l = bitString.length() / m;


    for (int i = 0; i < m; ++i) {
        if (logicObj->isInterruptionRequested()) {
            return;
        }
        test = bitString.substr(i * l, l);

        pvalue1 = lseries(test);
        pvalue[1].push_back(pvalue1);


        pvalue1 = mdu2(test);
```

```
        pvalue[2].push_back(pvalue1);

        pvalue1 = mdu(test);
        pvalue[3].push_back(pvalue1);

        pvalue1 = empbo(ll, test);
        pvalue[4].push_back(pvalue1);

        pvalue1 = series(test);
        pvalue[5].push_back(pvalue1);

        pvalue1 = spect(test);
        pvalue[6].push_back(pvalue1);

        pvalue1 = scapro(4, 10, test);
        pvalue[7].push_back(pvalue1);

        pvalue1 = appent(test);
        pvalue[0].push_back(pvalue1);

        progress += iterationProgress;

        if (progress - lastEmittedProgress >= 1.0 || i == m - 1) {
            if (logicObj) {
            updateProgressCallback(progressBar, (int)progress);
            }
                lastEmittedProgress = progress;
            }
        }

    ber(pvalue, a, res);
    chi(pvalue, k, res);

    for (int i = 0; i < 8; ++i) {
        for (int j = 0; j < m; ++j) {
            pmin = min(pmin, pvalue[i][j]);
        }
        bmin = min(res[0][i], bmin);
        cmin = min(res[1][i], cmin);
    }

    if(inFileSize == 8388608){
        if (bmin < b) {
            result +=
                "H1 Bernoulli: Pmin < B\t( "
                + QString::fromStdString(std::to_string(bmin)) + " < "
                     + QString::fromStdString(std::to_string(b))+" ),\n";
        }
        else result += "H0 Bernoulli: Pmin = " +
QString::fromStdString(std::to_string(bmin))+",\n";

        if (cmin < b) {
            result +=
                "H1 Chi: Pmin < B\t( "
                + QString::fromStdString(std::to_string(cmin)) + " < "
                + QString::fromStdString(std::to_string(b))+" );";
        }
        else result += "H0 Chi: Pmin = " +
QString::fromStdString(std::to_string(cmin))+";";
      }
    if (bmin < b) {
        out <<
            "\n----\nH1, Bernoulli - P1min < B: \t "
```

```
            << bmin << " < " << b << std::endl <<
            "----\n";
    }
    else out << "\n----\nH0, Bernoulli - P1min = " << bmin << std::endl <<
        "----\n";

    if (cmin < b) {
        out <<
            "\n----\nH1, Chi - P2min < B: \t "
            << cmin << " < " << b << std::endl <<
            "----\n\n";
    }
    else out << "\n----\nH0, Chi - P2min = " << cmin << std::endl <<
        "----\n\n";

    if(fullOutputFlag != 0)
    out << "\t\ttest Entropy\n\nP-value(s):" << fixed << setprecision(6) <<
std::endl;

    if (fullOutputFlag != 0) {
        for (int i = 0; i < 8; ++i) {
            for (int j = 0; j < m; ++j) {
                out << '\t' << pvalue[i][j];
            }
            out << "\nBernoulli: " << res[0][i] << "\nChi:" << res[1][i] <<
"\n\n";

            if (i == 0) out << "\t\ttest Long Series\n\nP-value(s):" <<
std::endl;

            else if (i == 1) out << "\t\ttest MDU2\n\nP-value(s):" << std::endl;
            else if (i == 2) out << "\t\ttest MDU\n\nP-value(s):" << std::endl;
            else if (i == 3) out << "\t\ttest Empty Boxes\n\nP-value(s):" <<
std::endl;
            else if (i == 4) out << "\t\ttest Series\n\nP-value(s):" <<
std::endl;
            else if (i == 5) out << "\t\ttest Spectral\n\nP-value(s):" <<
std::endl;
            else if (i == 6) out << "\t\ttest Scalar\n\nP-value(s):" <<
std::endl;
        }
    }
}
```

ПРИЛОЖЕНИЕ Н

Код тестирования нескольких последовательностей на C++

```
#pragma once

#include <QString>
#include <cmath>
#include <vector>
#include <fstream>

extern QString result;

void test_n(std::ofstream& out, std::vector<std::vector<std::vector <double>>>&
results, double b, double ga) {
      int M = results.size();
      double dmin, dmax, vmin = 1;
    std::vector <std::vector <double>> v(2, std::vector<double>(8));

      for (int mk = 0; mk < 2; ++mk) {
            for (int i = 0; i < 8; ++i) {
                  for (int j = 0; j < M; ++j) {
                        if (results[j][mk][i] >= b) v[mk][i] += 1;
                  }
                  v[mk][i] /= M;
            }
      }

    dmin = fmax(0.0, 1 - ga - 3 * sqrt(ga * (1 - ga) / M));
    dmax = fmin(1.0, 1 - ga + 3 * sqrt(ga * (1 - ga) / M));

      for (int mk = 0; mk < 2; ++mk) {
            for (int i = 0; i < 8; ++i) {
            vmin = fmin(vmin, v[mk][i]);
            }
      }

    if (vmin >= dmin && vmin <= dmax) {
        result = "H0, Vmin = "
            + QString::fromStdString(std::to_string(vmin)) + "\tVmin belongs ["
            + QString::fromStdString(std::to_string(dmin)) + ", "
            + QString::fromStdString(std::to_string(dmax))+"]";
        out << "\n\n\n----------------------------------------\nH0, Vmin = "
        << vmin << "\tVmin принадлежит [" << dmin << ", " << dmax << "]" <<
std::endl
            << "----------------------------------------\n";
    }
    else {
        result = "H1, Vmin = "
                + QString::fromStdString(std::to_string(vmin)) + "\tVmin does
not belong ["
                + QString::fromStdString(std::to_string(dmin)) + ", "
                + QString::fromStdString(std::to_string(dmax))+"]";
        out << "\n\n\n----------------------------------------\nH1, Vmin = " <<
vmin << "\tVmin не принадлежит[" << dmin << ", " << dmax << "]" << std::endl
            << "----------------------------------------\n";
    }
}
```

ПРИЛОЖЕНИЕ П

Код общей логики разбора входных параметров и запуска процедуры тестирования с соответствующими параметрами на C++

```
#include <iostream>
#include <vector>
#include <QThread>
#include <QHeaderView>
#include <stdlib.h>
#include <cstdlib>
#include <fstream>
#include <string>
#include <math.h>
#include <ctime>
#include <bitset>
#include <omp.h>
#include "test_one.h"
#include "test_n.h"
#include "testing.h"

#define _USE_MATH_DEFINES
int fullOutputFlag = 1;
unsigned long long int inFileSize = 0;
double progress = 0;
QString result = "";


using namespace std;

TestLogic::TestLogic(QObject* parent, QProgressBar* progressBar1) :
    QObject(parent),
    m_interruptionRequested(false)
{
    this->progressBar = progressBar1;
}

void TestLogic::requestInterruption() {
    m_interruptionRequested.store(true);
    if (m_inputStream && m_inputStream->is_open()) {
        m_inputStream->close();
        delete m_inputStream;
        m_inputStream = nullptr;
      }
    if (m_outputStream && m_outputStream->is_open()) {
        m_outputStream->close();
        delete m_outputStream;
        m_outputStream = nullptr;
    }
}

bool TestLogic::isInterruptionRequested() const {
    return m_interruptionRequested.load();
}


void TestLogic::testFile(const QString& inputFile, const QString& outputFile) {
    m_interruptionRequested.store(false);
    m_inputStream = new std::ifstream(inputFile.toStdString(),
std::ios::binary);
    m_outputStream = new std::ofstream(outputFile.toStdString());
```

```
    if (!m_inputStream->is_open()) {
        emit messageLogged("Error: cant open input file");
        m_interruptionRequested.store(true);
        emit testingFinished(isInterruptionRequested());
        return;
      }
    if (!m_outputStream->is_open()) {
        emit messageLogged("Error: cant open output file");
        m_interruptionRequested.store(true);
        emit testingFinished(isInterruptionRequested());
        return;
    }

    setlocale(LC_ALL, ".1251");
    vector<vector <double>> res(2, vector<double>(8));
    vector<vector<vector <double>>> results;
    vector <string> bitStrings;
    string bitString, bitString1 = "", output;
    char byte;
    int mb10 = 83886080, mb1 = 8388608;


    m_inputStream->seekg(0, ios::end);

        streampos fileSize = m_inputStream->tellg();
    inFileSize = fileSize * 8;

    m_inputStream->seekg(0, ios::beg);

    emit messageLogged("Input file size: " +
QString::fromStdString(to_string(inFileSize)));

    int numthr = 1, num = inFileSize / mb10, part = 1;
    if(num == 0) {
        num = inFileSize / mb1;
        if(num == 0) return;
    }

    if (num < omp_get_max_threads() - 2) numthr = num;
    else numthr = omp_get_max_threads() - 2;
    while (num % numthr != 0) numthr -= 1;
    part = numthr;

    emit messageLogged("Threads num: " +
QString::fromStdString(to_string(numthr)));


    while (m_inputStream->get(byte)) {
        if (isInterruptionRequested()) {
            emit messageLogged("The operation was interrupted by the user");
            emit testingFinished(isInterruptionRequested());
            return;
        }

        bitString1 = bitset<8>(static_cast<unsigned char>(byte)).to_string();
        reverse(bitString1.begin(), bitString1.end());
        bitString += bitString1;

        if (bitString.length() == mb10) {

            bitStrings.push_back(bitString);
            bitString.clear();
```

```
            if (bitStrings.size() == part) {

#pragma omp parallel for num_threads(numthr)
                for (int i = 0; i < part; ++i) {
                    if (isInterruptionRequested()) {
                        continue;
                    }
#pragma omp critical
                    {
                        output = "Thread " + to_string(omp_get_thread_num()) +
": processes the ["
                                 + to_string(results.size() * mb10 + i * mb10) +
", "
                                 + to_string(results.size() * mb10 + (i + 1) *
mb10) + "]\n";
                        emit messageLogged(QString::fromStdString(output));
                    }

                    vector<vector <double>> res1(2, vector<double>(8));
                    test_one(*m_outputStream, bitStrings[i], 200, 10, 0.05,
0.0001, res1, this, progressBar);

                    if (isInterruptionRequested()) {
                        continue;
                    }

#pragma omp critical
                    {
                        results.push_back(res1);
                    }
                }
                bitStrings.clear();
            }
            if (isInterruptionRequested()) {
                emit messageLogged("The operation was interrupted by the user");
                emit testingFinished(isInterruptionRequested());
                return;
            }
        }
      }
    if (isInterruptionRequested()) {
        emit messageLogged("The operation was interrupted by the user");
        emit testingFinished(isInterruptionRequested());
        return;
    }

    if (bitStrings.size() > 0) {

#pragma omp parallel for num_threads(bitStrings.size())
        for (int i = 0; i < bitStrings.size(); ++i) {
#pragma omp critical
            {
                output = "Thread " + to_string(omp_get_thread_num()) + ":
processes the ["
                    + to_string(results.size() * mb10 + i * mb10) + ", "
                    + to_string(results.size() * mb10 + (i + 1) * mb10) + "]\n";
                emit messageLogged(QString::fromStdString(output));
            }

            vector<vector <double>> res1(2, vector<double>(8));
            test_one(*m_outputStream, bitStrings[i], 200, 10, 0.05, 0.0001,
res1, this, progressBar);
```

```
#pragma omp critical
            {
                results.push_back(res1);
            }
        }
        bitStrings.clear();
    }

    if (results.size() == 0) {

            if (bitString.length() == mb1) {
                test_one(*m_outputStream, bitString, 104, 10, 0.05, 0.0001, res,
this, progressBar);

                if (isInterruptionRequested()) {
                    emit messageLogged("The operation was interrupted by the
user");
                    emit testingFinished(isInterruptionRequested());
                    return;
                }

                results.push_back(res);
                bitString = "";
            }
            else if(bitString.length() > mb1){
                bitString1 = bitString.substr(0, mb1);
                test_one(*m_outputStream, bitString1, 104, 10, 0.05, 0.0001,
res, this, progressBar);

                if (isInterruptionRequested()) {
                    emit messageLogged("The operation was interrupted by the
user");
                    emit testingFinished(isInterruptionRequested());
                    return;
                }

                bitString.erase(0, mb1);
            }
        if (results.size() > 1) {
            test_n(*m_outputStream, results, 0.0001, 0.01);
        }
    }
    else if (results.size() > 1)
        test_n(*m_outputStream, results, 0.0001, 0.01);


    if (isInterruptionRequested()) {
        emit messageLogged("The operation was interrupted by the user");
        emit testingFinished(isInterruptionRequested());
        return;
    }

    m_inputStream->close();
    if (!m_inputStream->is_open()) {
        emit messageLogged("file " + inputFile + " was closed.");
    }

    m_outputStream->close();
    if (!m_outputStream->is_open()) {
        emit messageLogged("file " + outputFile + " was closed.");
    }
```

```
    if (!results.empty()) {
        setTable(results);
    }

    progressUpdated(100.0);
    progress = 0;
    emit resultReady(result);
    result = "";
    emit testingFinished(isInterruptionRequested());
    return;
}
```