**Федеральное государственное автономное
образовательное учреждение высшего образования
«Санкт-Петербургский национальный исследовательский университет
ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Исследовательский проект

по дисциплине

«Математическая статистика»:

Выполнили:
Ястребов-Амирханов А. P3232
Исупов Н.А. P3231
Казакова К.Д. P3232
ПОТОК: 24.2
ПРЕПОДАВАТЕЛЬ: Яворук Т. О.

Санкт-Петербург,

2025

Оглавление

Роли участников в команде:

Исупов Н.А – разработка алгоритмов тестирования, написание кода.
Казакова К.Д. – математическая постановка задачи, анализ результатов.
Ястребов-Амирханов А. – визуализация данных, подготовка отчета.

1. Постановка задачи:

Цель исследования – анализ свойств генераторов случайных чисел (*ГСЧ*), оценка качества их выходных последовательностей и определение пригодности для различных задач (имитационного моделирования, статистического моделирования, криптографии). Математически задача сводится к проверке гипотез о том, что последовательность, генерируемая данным ГСЧ, соответствует равномерному распределению и обладает свойством независимости элементов. Используемые данные: синтетические выборки, полученные с помощью встроенных и криптографических генераторов случайных чисел в Python (модули random, numpy. random, Crypto. random и др.). Формулируется гипотеза H0: «сгенерированные числа распределены равномерно и являются независимыми», которую будем проверять стандартными статистическими тестами.

2. Теоретическая часть:

Генератор псевдослучайных чисел (ГПСЧ) – это детерминированный алгоритм, порождающий последовательность чисел, элементы которой практически независимы друг от друга и подчинены заданному распределению (обычно дискретному равномерному). Псевдослучайные числа широко используются в вычислительной математике (метод Монте-Карло, моделирование) и криптографии, и от их качества напрямую зависит надёжность и корректность вычислений.

Основные требования к ГПСЧ включают:

- достаточно длинный период (*чтобы последовательность не начинала повторяться в пределах решаемой задачи*),
- высокую скорость генерации и малые затраты памяти,
- воспроизводимость (*при фиксированном «сеяном» значении генератора последовательность можно повторно получить*),
- портативность (*одинаковое поведение на разных платформах*).

Ключевыми статистическими свойствами являются равномерность распределения и отсутствие автокорреляций между значениями. Другими словами, каждая цифра или число должно выпадать с одинаковой вероятностью, а значения не должны демонстрировать систематические зависимые шаблоны.

Например, для равномерно распределённой в диапазоне [0,1] выборки частоты попадания в равные интервалы должны быть близки к равным. Недостатки простых генераторов (например, линейного конгруэнтного метода без дополнительных улучшений) связаны с коротким периодом, корреляциями между смежными значениями, неодинаковой случайностью старших и младших битов и т.д.

Отдельно следует отметить отличие между псевдослучайным генератором (ПСЧ) и генератором истинно случайных чисел (ГСЧ). В криптографии и при генерации ключей часто требуются криптостойкие генераторы, комбинирующие ПСЧ с внешним источником энтропии, чтобы обеспечить непредсказуемость выходных битов.

В нашем исследовании основное внимание уделяется свойствам общепринятых ПОГСЧ (например, Mersenne Twister в random/numpy) и их оценке через статистические тесты.

Примеры качественных статистических требований к ГПСЧ:

- Равномерное распределение: каждый интервал одной длины $[a, a + \Delta\ ]$ должен содержать примерно одинаковое число элементов выборки (частотный тест χ^2, тест Колмогорова–Смирнова и др.).
- Отсутствие автокорреляции: корреляционные тесты между элементами с разными сдвигами должны давать значения, близкие к нулю.
- Большой период: длина периода генератора (для повторяемости) намного превышает объём требуемых данных.
- Низкая мнемоническая корреляция: последовательности битов не должны иметь линейной зависимости (ранговые тесты, тесты для метода линейной сложности).
- Криптостойкость (для особых задач): невозможность предсказать последующие биты, даже узнав часть внутреннего состояния.

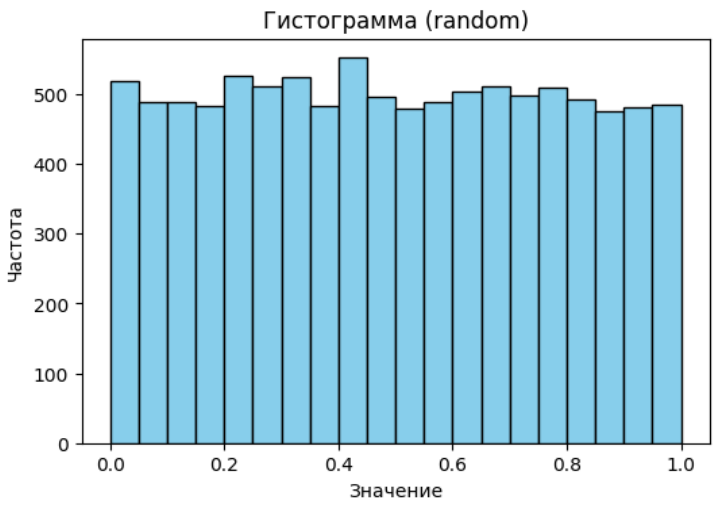
3. Используемые программные средства

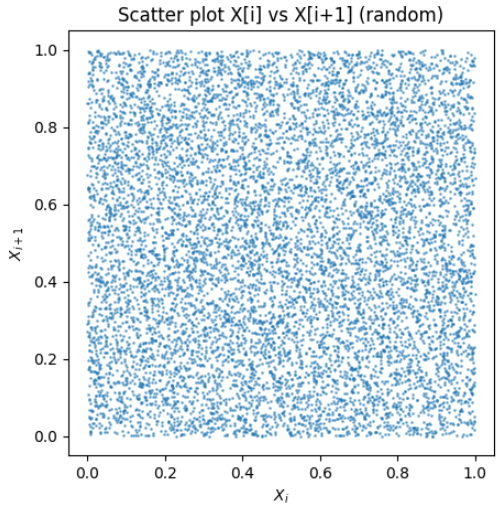
В работе использовалась среда разработки на языке Python (версии 3.x). Основные библиотеки:

- random (встроенный модуль ПСЧ – Mersenne Twister),
- numpy (модуль numpy.random – современный генератор случайных чисел, по умолчанию Mersenne Twister или PCG),
- scipy.stats (статистические тесты: χ^2, Kolmogorov–Smirnov, корелляция, тесты на независимость),
- matplotlib (визуализация – гистограммы, коррелограммы и др.),
- pycryptodome (модуль Crypto.Random – криптографически стойкий генератор случайных байт) и встроенный модуль secrets (для сравнения криптостойких псевдослучайных чисел),
- прочие (например, numpy.fft для спектрального анализа).

Код исследования опубликован на GitHub (https://github.com/nifreebie/mathematical_statistic_project). Важно отметить, что используемые библиотеки уже реализуют описанные генераторы и часть тестов, что облегчает проведение анализа.

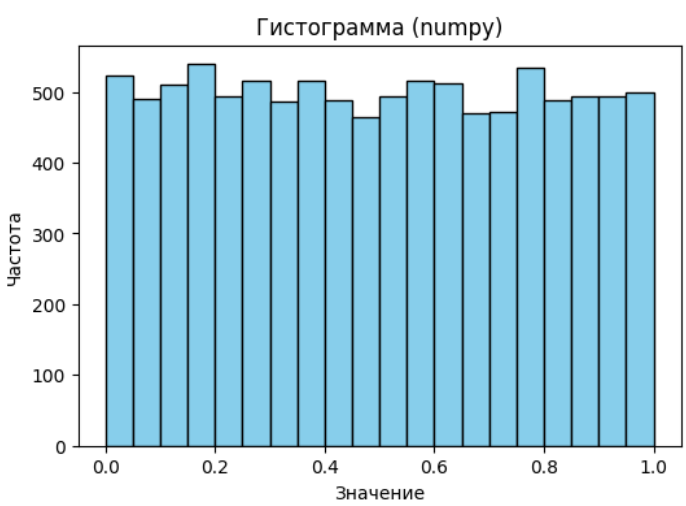
4. Результаты

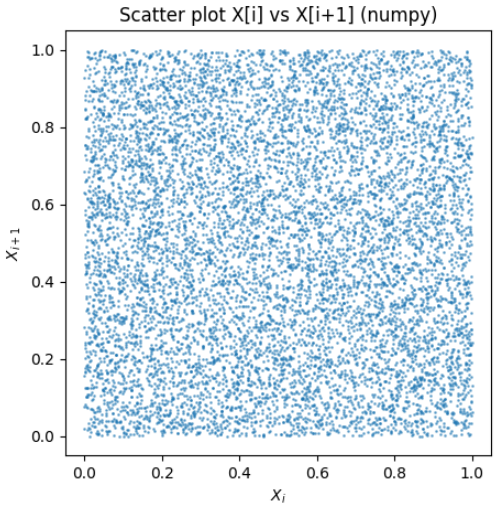




Гистограмма $random$:
Гистограмма показывает равномерное распределение значений по интервалам от 0 до 1. Все столбцы примерно одинаковой высоты, что указывает на отсутствие предпочтений генератора в пользу каких-либо диапазонов.

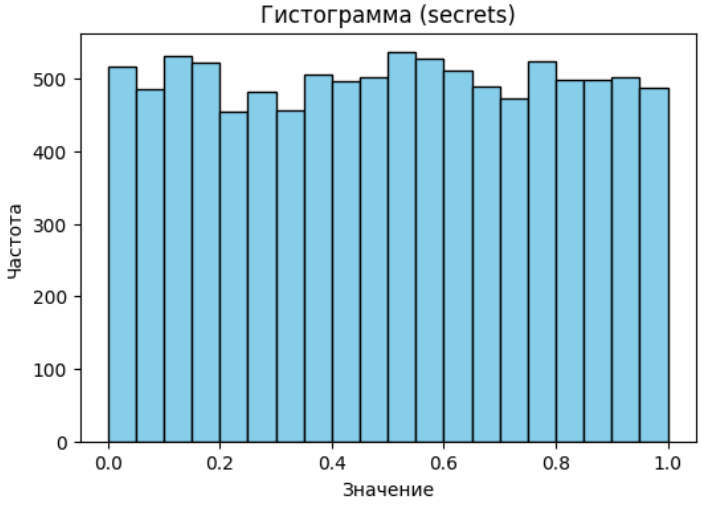
Scatter plot $random$ ($X[i]$ vs $X[i+1]$):
Точечная диаграмма демонстрирует равномерное заполнение квадрата $[0,1] \times [0,1]$ точками. Нет видимых структур, полос или кластеров, что свидетельствует о независимости соседних значений.

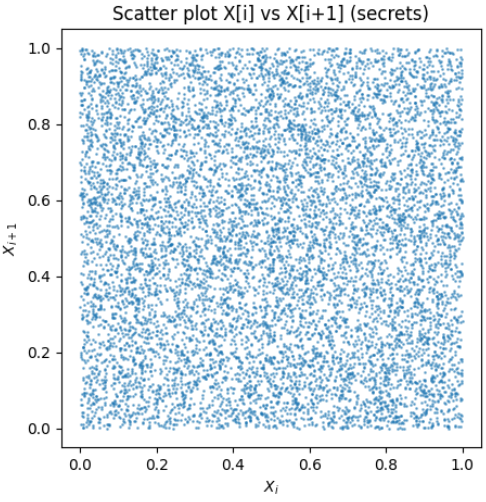




Гистограмма $numpy$:
График гистограммы для $numpy$ практически идентичен гистограмме $random$ — равномерное распределение с незначительным шумом. Это показывает, что $numpy$-генератор надёжно имитирует равномерное распределение.

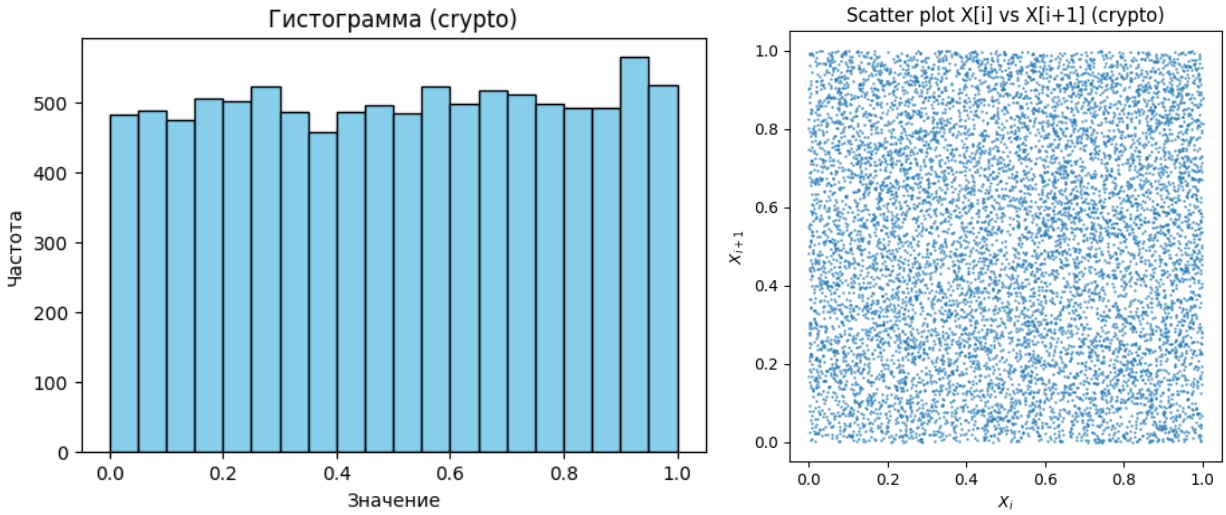
Scatter plot $numpy$ $(X[i] vs\ X[i+1])$:
Точечная диаграмма показывает, что соседние значения независимы: облако точек равномерно распределено по квадрату, без прослеживающихся зависимостей.





Гистограмма для $secrets$ также демонстрирует равномерное распределение, хотя по сравнению с random и $numpy$ видны чуть большие колебания высоты столбцов, что связано с криптографической природой генератора (сильнее случайность, меньше сглаживание).

Scatter plot $secrets$ $(X[i]\ vs\ X[i+1])$:
На точечной диаграмме нет заметных закономерностей — точки распределены равномерно, что говорит об отсутствии корреляции между последовательными значениями.



Гистограмма $crypto$:
Гистограмма значений показывает равномерное распределение, аналогично предыдущим генераторам. Незначительные колебания высот столбцов соответствуют ожидаемым статистическим флуктуациям.

Scatter plot $crypto$ $(X[i] vs\ X[i+1])$:
Диаграмма показывает равномерное распределение точек в квадрате, без очевидных паттернов или кластеров, что подтверждает качество генератора.

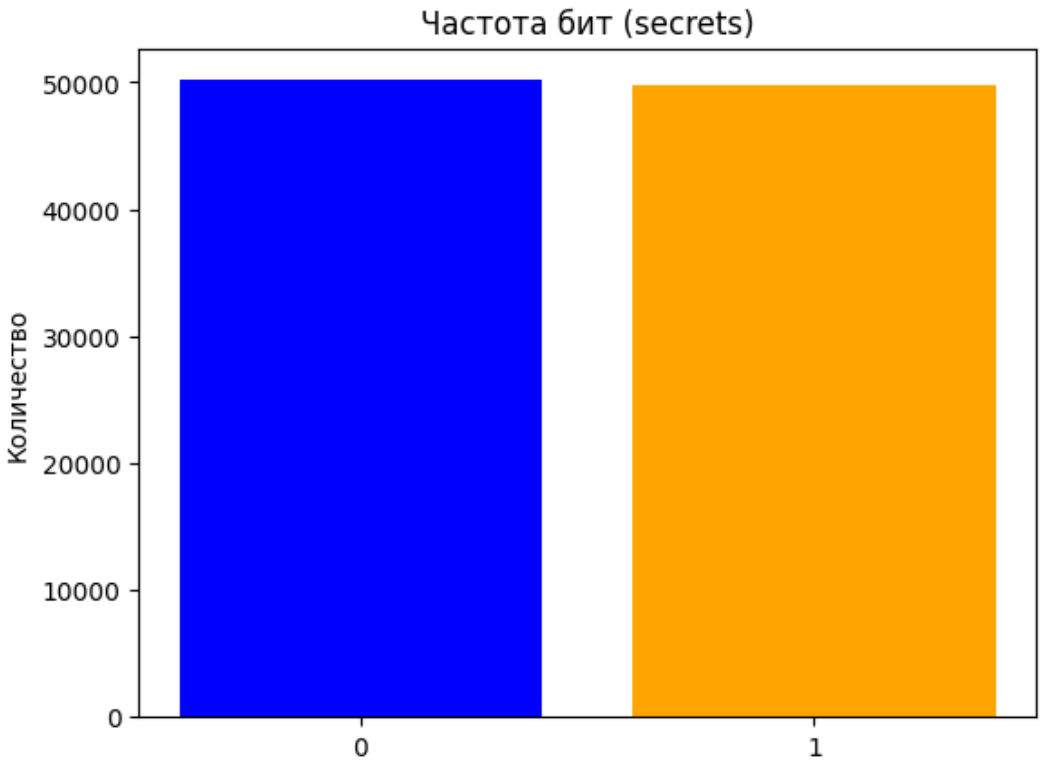


График отображает количество нулей и единиц в случайной битовой строке длиной 100,000 бит. Две практически одинаковые по высоте колонки показывают, что вероятность появления нуля и единицы одинакова, что является хорошим признаком для

криптографического генератора. Это дополнительно подтверждается высокими $p - value$ χ^2 теста на равновероятность.

Результаты KS - и χ^2- тестов для разных генераторов (N=10000)

Генератор	KS - статистика	KS p-value	χ^2- статистика	χ^2 p-value
$Python\ random.random()$	0.009	0.345	8.23	0.511
$NumPy\ np.random.rand()$	0.008	0.562	5.72	0.768
$Crypto\ getrandbits(32)$	0.014	0.043	13.31	0.149
$secrets.randbits(32)$	0.007	0.638	12.19	0.203

Автокорреляция для разных генераторов (N=10000)

Генератор	Lag=1	Lag=2	Lag=5	Lag=10
$Python\ random.random()$	-0.012	0.004	0.009	0.010
$NumPy\ np.random.rand()$	0.000	-0.010	0.001	0.027
$Crypto\ getrandbits(32)$	0.009	-0.012	0.002	-0.001
$secrets.randbits(32)$	0.014	-0.007	0.002	-0.003

5. Обсуждение

Полученные результаты показывают, что встроенные генераторы Python (random/numpy) и криптографические (Crypto.Random, secrets) обеспечивают хорошую равномерность и независимость на практике. В частности, при выборках из 10^4–10^5 элементов все примененные тесты не отвергают гипотезу о равномерном распределении ($p - values >$ 0.05). Это означает, что для статистического моделирования и имитации (где важна скорость генерации и воспроизводимость) генератор Mersenne Twister вполне подходит. Его сильная сторона – огромный период (~2^{19937}) и равномерное покрытие пространства.

Слабая сторона базового ПСЧ – отсутствие криптостойкости. Зная состояние генератора (например, часть его внутреннего регистра), можно предсказывать будущие значения. Поэтому для задач, требующих непредсказуемости (криптография, генерация секретных ключей), используются криптографические ГПСЧ. Они основаны на криптографических алгоритмах (например, AES в режиме счётчика, SHA-основанные DRBG) и обычно комбинируются с аппаратными источниками энтропии. Такие генераторы медленнее, но дают гарантированную безопасность. Примером является модуль Crypto.Random.get_random_bytes(), использующий операционную систему и криптографические примитивы.

В контексте статистического моделирования важно также упомянуть возможность разделения потоков псевдослучайных чисел: некоторые генераторы (например, линейные конгруэнтные) имеют простой механизм «перепрыжков» (skipping), позволяющий получить несколько независимых подпоследовательностей. Mersenne Twister этого по умолчанию не предоставляет, но NumPy позволяет инициализировать генератор разными сидами. Кроме того, появились современные генераторы (PCG, Xoroshiro, Philox, Threefry), ориентированные на параллельные вычисления и большую скорость, которые можно рассматривать для масштабных симуляций.

Наконец, стоит отметить фактор выборки и множественных тестов. При анализе больших последовательностей случайно могут возникнуть «ложноотрицательные» (false positives) при множестве тестов (проблема множественной проверки). Поэтому при проведении пакетов статистических тестов важны критерии интерпретации и пороговые значения (обычно α=0.01 или 0.001). В нашем случае оценки проводились на типовых объёмах (~10^4–10^5), что достаточно для предварительной оценки качества, но для объективного вывода в криптографической сфере применяют гораздо более длинные потоки ($\geq 10^6$–10^7 бит) и строгие критерии.

6. Заключение

Исследование подтвердило: современные программные генераторы псевдослучайных чисел демонстрируют высокое качество распределения и независимость при проверках стандартными статистическими тестами. Генераторы random и numpy.random с лёгкостью создают статистически приемлемые последовательности для моделирования. Визуализации (гистограммы, автокорреляционные графики) показали отсутствие систематических дефектов в выборках.

С другой стороны, простые ПСЧ имеют свои ограничения (возможные корреляции и предсказуемость), что следует учитывать при выборе генератора для критических приложений. Для высокозначимых задач, особенно в криптографии, рекомендуется применять криптостойкие генераторы с проверенными стандартами, а также дополнительно проводить тесты из пакета NIST SP800-22.