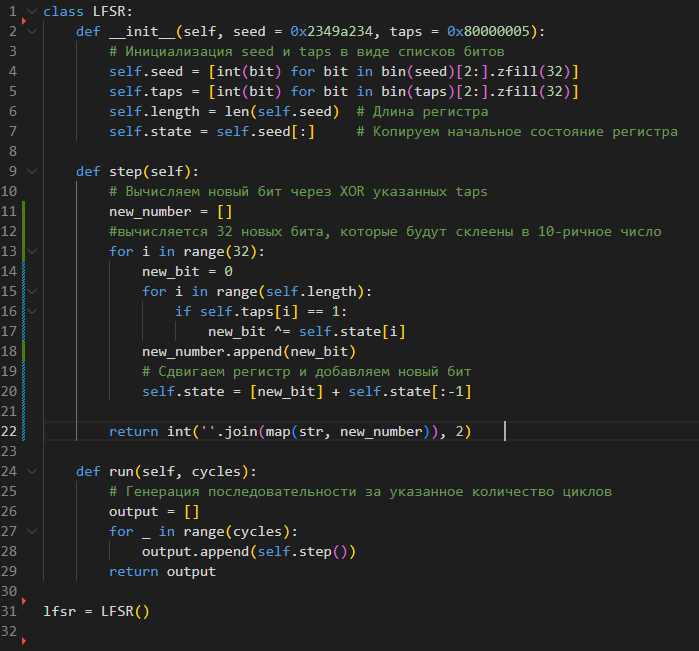
Генераторы псевдослучайных чисел и их проверка

Сутормин М. П. ИНБ-б-о-22-1

Было выбрано 4 генератора случайных чисел, описанных в книге Слеповичева. А именно fish, линейный конгруэнтный генератор, вихрь Мерсенна, регистр сдвига с линейной обратной связью. Алгоритмы были реализованы для десятичных чисел на языке python.

Генератор: регистр сдвига с линейной обратной связью изображен на рисунке 1:

В начале происходит Инициализация начального заполнения seed и порождающего полинома taps в виде списков битов. По алгоритму, описанному в книге Слеповичева, генерируется новый бит последовательности, который записывается в первую ячейку памяти и в список new\_number. Как только длина new\_number станет равна 32, оно выводится, как результат одной итерации. В методе run указывается необходимое количество чисел которые нужно сгенерировать.

Рисунок 1 – Генератор: регистр сдвига с линейной обратной связью

Генератор fish изображен на рисунке 2. В начале происходит заполнение двух листов по 55 и по 52 элемента с помощью стартовых элементов *A*0 и *B*0. По алгоритму, описанному в книге Слеповичева на i-ой операции высчитываются значения *A*i и *B*i по формулам

К этим последовательностям применяется процедура прореживания в зависимости от младшего значащего бита *Bi* : если значение равно 1, то пара используется, если 0 – игнорируется.

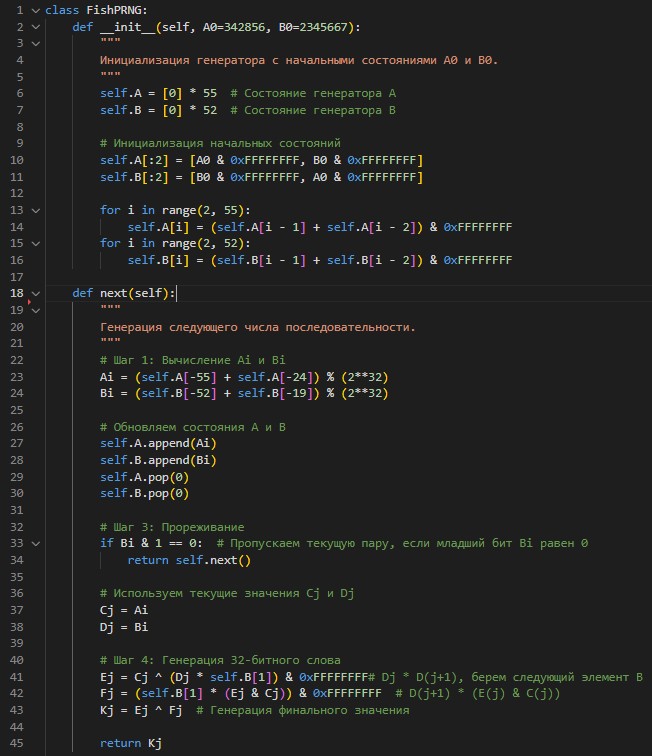
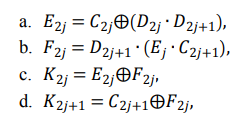


Рисунок 2 – Генератор: fish

Результатом работы генератора на j-м шаге является 32-битное слово, вычисленное по формуле:



где Cj – последовательность используемых слов Aj , а Dj – это последовательность используемых слов Bj. K2j и K2j+1 – пара, получаемая на выходе генератора. В классе FishPRNG также есть метод run, который на вход принимает количество чисел, а на выходе выдаёт лист случайных чисел заданной длинны.

Линейный конгруэнтный генератор изображён на рисунке 3.

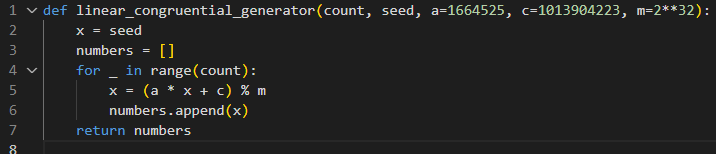


Рисунок 3 – линейный конгруэнтный генератор

В данном генераторе задаются константы a, c, m и начальное заполнение seed. i-е число считается по формуле:

4-ый генератор: вихрь Мерсенна. Он также описан в книге [1], реализован он во встроенном в python модуле random.

В итоге на вход каждому генератору подаётся количество чисел, которые будут сгенерированы по соответствующему алгоритму. Все десятичные числа были ограничены 32 битами.

Весь код разбит на несколько файлов и импортирован в 1 файл ipynb, где были также рассмотрены некоторые графики: “гистограмма частот каждого значения”, “график значений для первых 100 чисел”, “гистограмма частоты разностей соседних элементов”, “график зависимости i+1 от i элемента для первых 1000 чисел”. В начале генерируется по 100 000 чисел с помошью каждого генератора. По этим последовательностям будут сгенерированы некоторые графики, которые представлены на рисунках 4 – 8

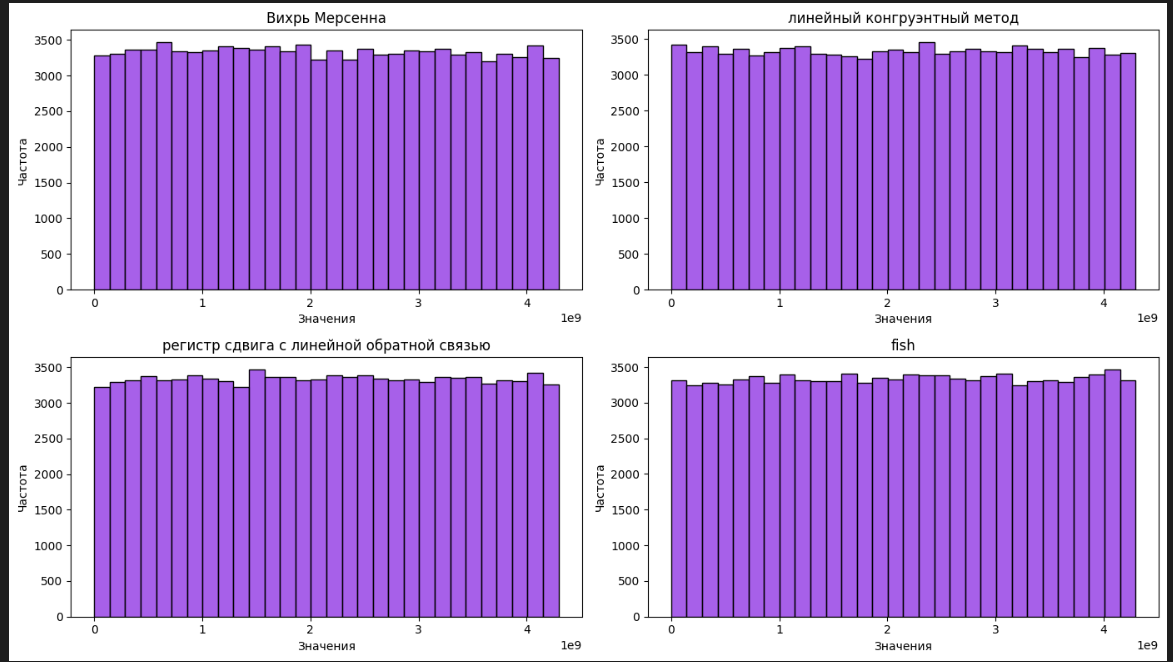


Рисунок 4 – гистограмма частот каждого значения

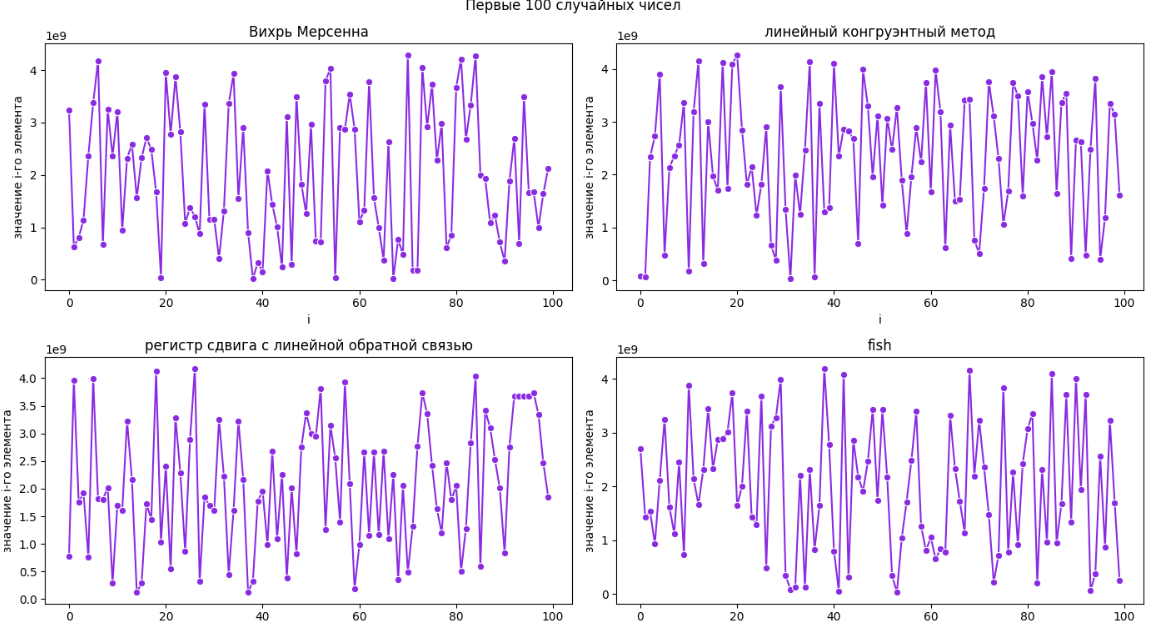


Рисунок 5 – график значений для первых 100 чисел

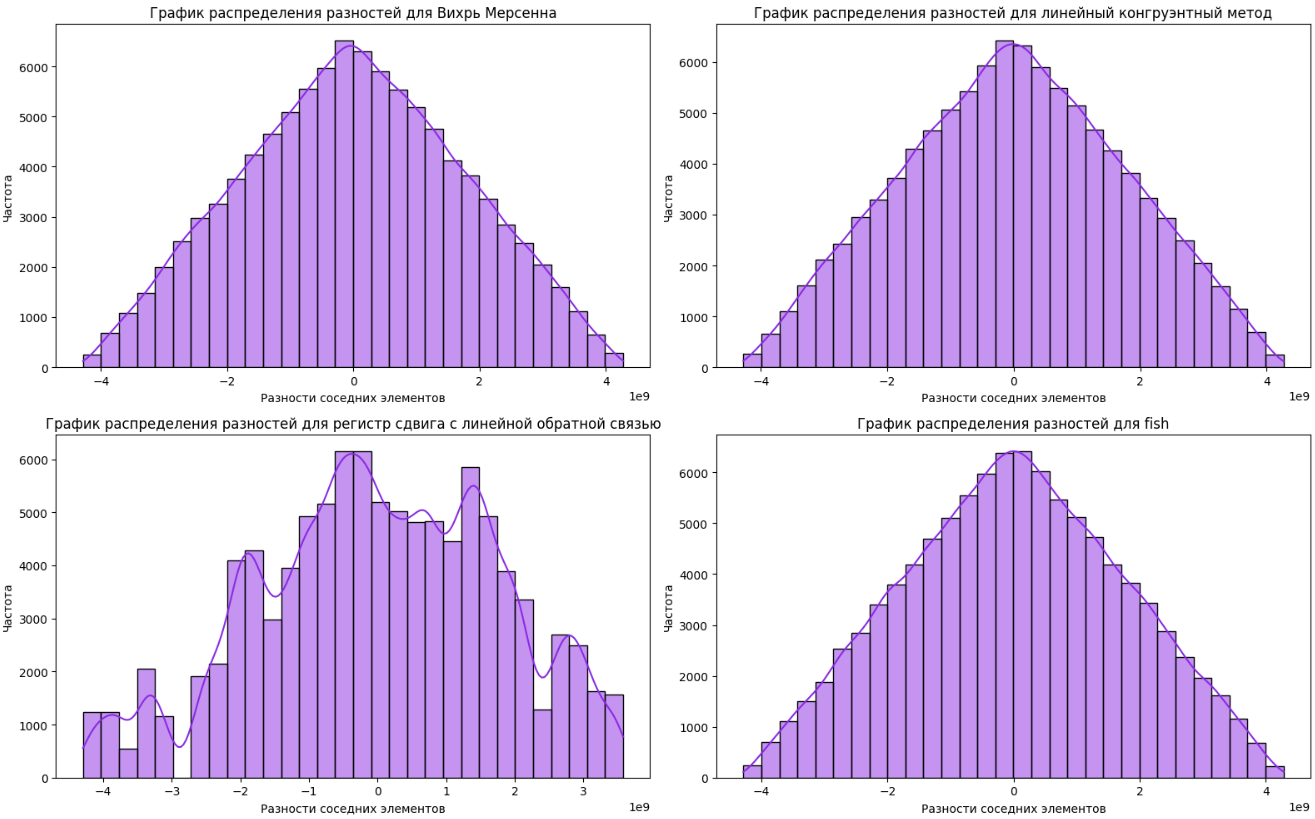


Рисунок 6 – гистограмма частоты разностей соседних элементов

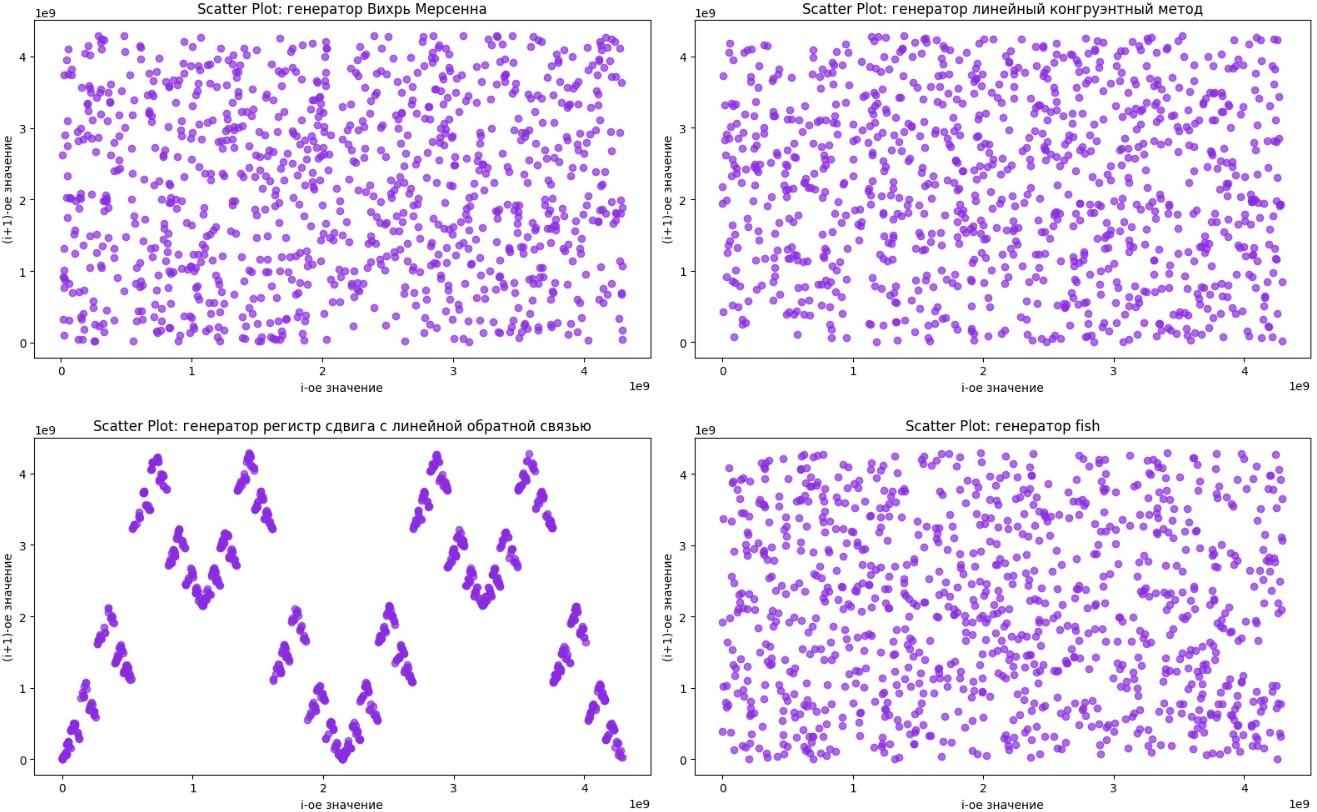


Рисунок 7 – график зависимости (*i*+1)-го значения от *i*-го значения

Были изучены некоторые статистические проверки по нормам NIST. Были выбраны тесты: “частотный побитовый тест”, “Частотный блочный тест”, “Тест на идущие подряд биты”, “Спектральный тест”. Их математическая реализация и расчёт p-уровня значимости были рассмотрены в статье [2]. Тесты были реализованы на языке Python.

частотный побитовый тест. Очевидно, что чем более случайна последовательность, тем ближе это соотношение к 1. Данный тест оценивает, насколько это соотношение близко к 1.

Принимаем каждую «1» за +1, а каждый «0» за -1 и считаем сумму по всей последовательности. Это можно записать так:

Sn = X1 + X2 +… + Xn, где Xi = 2xi — 1.

Возьмем такую последовательность: 1011010101, Тогда

S = 1 + (-1) + 1 + 1 + (-1) + 1 + (-1) + 1 + (-1) + 1 = 2

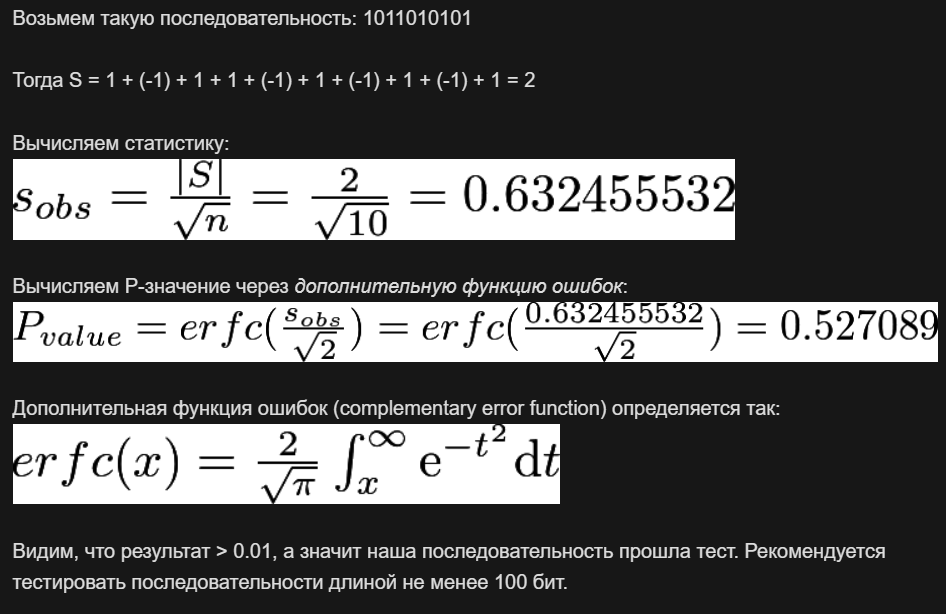


Рисунок 8 – математическая реализация частотного побитового теста

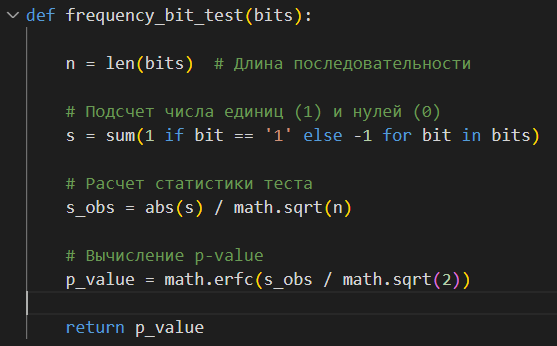


Рисунок 9 – программная реализация частотного побитового теста

Частотный блочный тест. Этот тест делается на основе предыдущего, только теперь значения пропорции «1»/«0» для каждого блока анализируются методом Хи-квадрат [3]. Ясно, что это соотношение должно быть приблизительно равным 1.

Например, пусть дана последовательность 0110011010. Разобъем ее на блоки по 3 бита (0 на конце отброшен):

011 001 101

Посчитаем пропорции πi для каждого блока: π1 = 2/3, π2 = 1/3, π3 = 1/3. Далее вычисляем статистику по методу Хи-квадрат c N степенями свободы (здесь N — количество блоков):

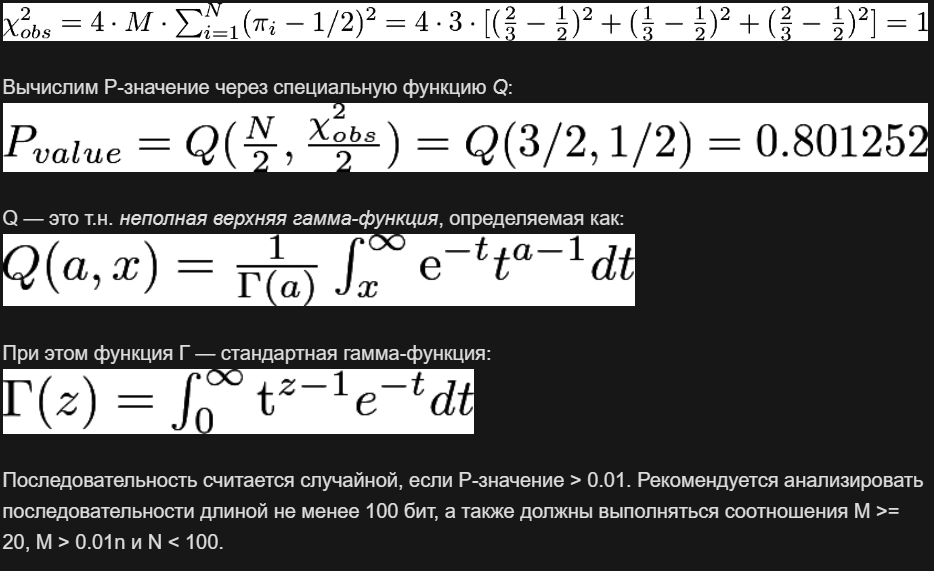


Рисунок 10 – математическая реализация частотного блочного теста

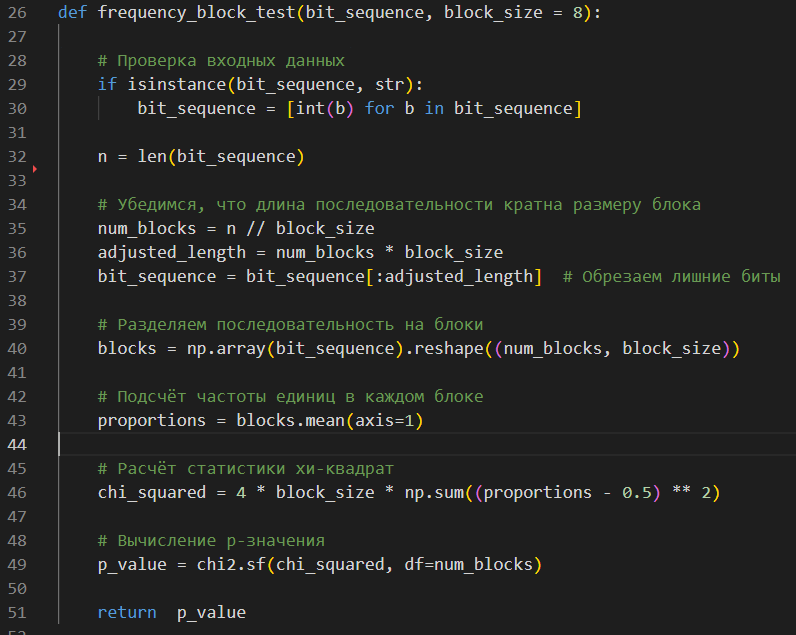


Рисунок 11 – программная реализация частотного блочного теста

Тест на одинаковые идущие подряд биты. В тесте ищутся все последовательности одинаковых битов, а затем анализируется, насколько количество и размеры этих последовательностей соответствуют количеству и размерам истинно случайной последовательности. Смысл в том, что если смена 0 на 1 (и обратно) происходит слишом редко, то такая последовательность «не тянет» на случайную.

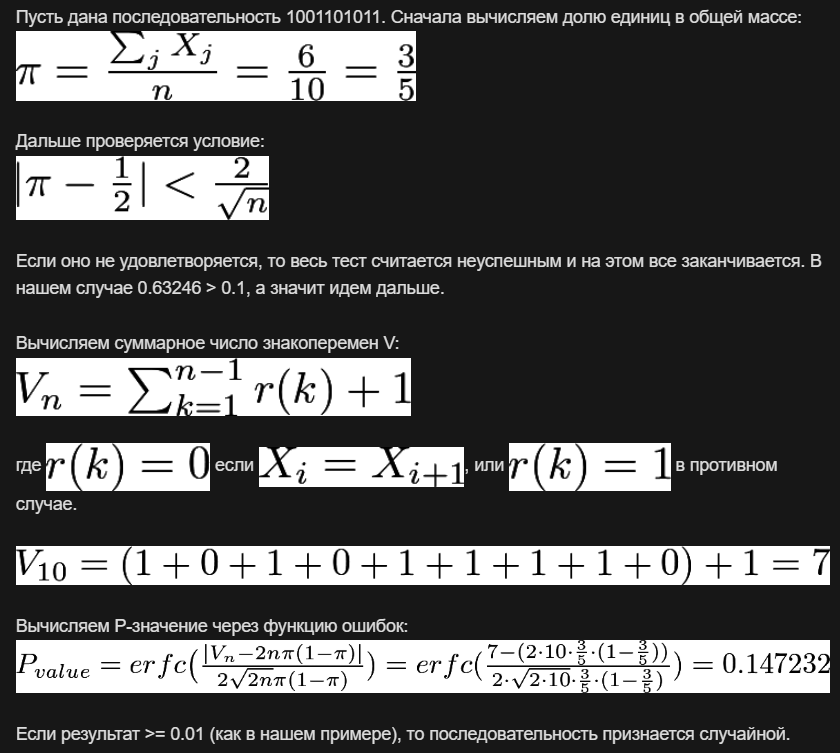


Рисунок 12 – математическая реализация теста на одинаковые идущие подряд биты

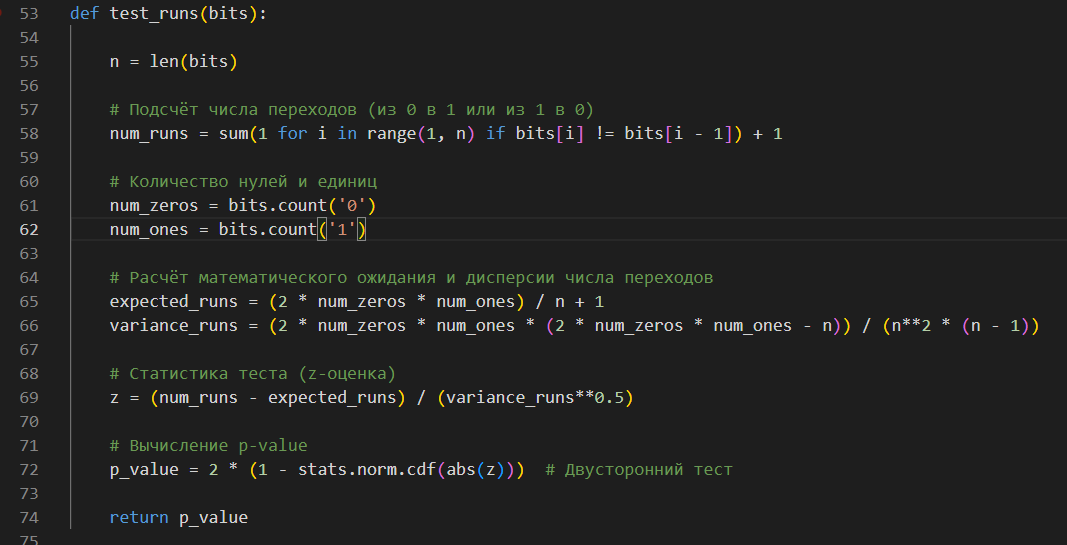
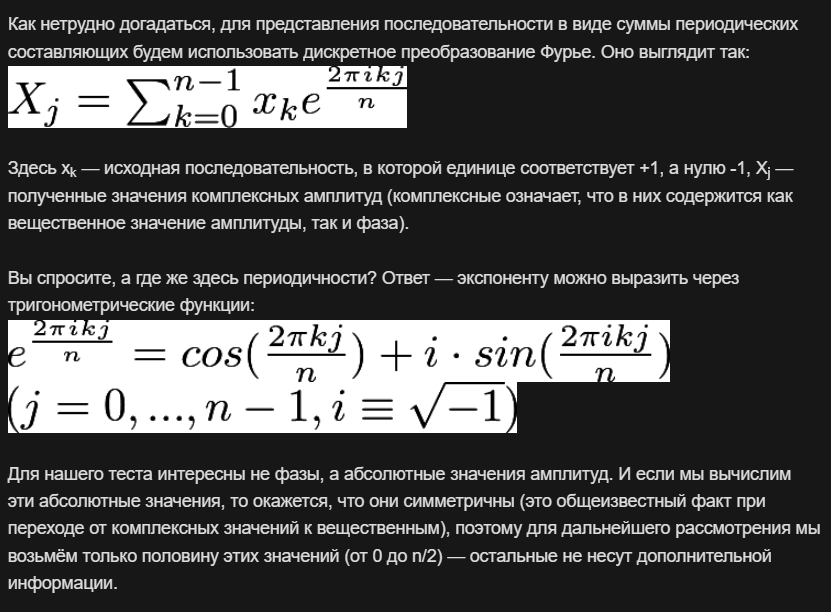


Рисунок 13 – программная реализация теста на одинаковые идущие подряд биты

Спектральный тест. Подопытная последовательность рассматривается как дискретный сигнал, для которого делается спектральное разложение с целью выявить частотные пики. Очевидно, что такие пики будут свидетельствовать о наличии периодических составляющих, что не есть гут. Если вкратце, то тест выявляет пики, превышающие 95%-й барьер, после чего проверяет, не превышает ли доля этих пиков 5%.



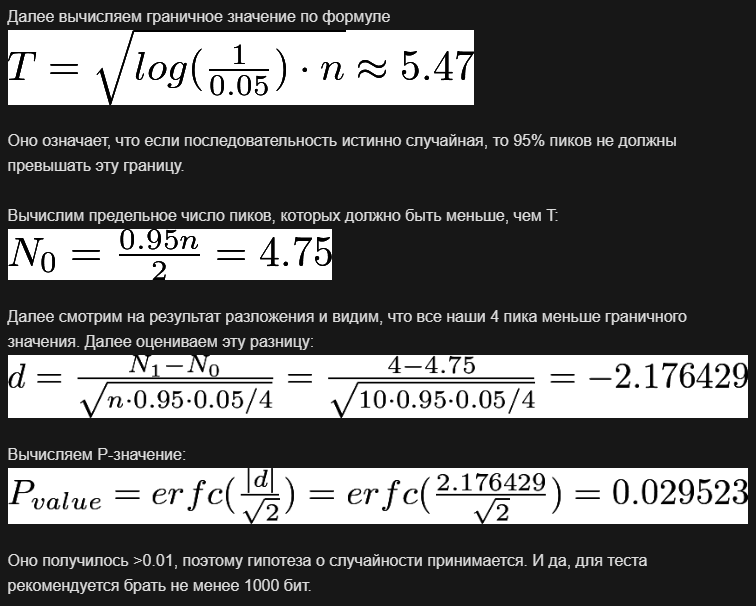


Рисунок 14 – математическая реализация спектрального теста

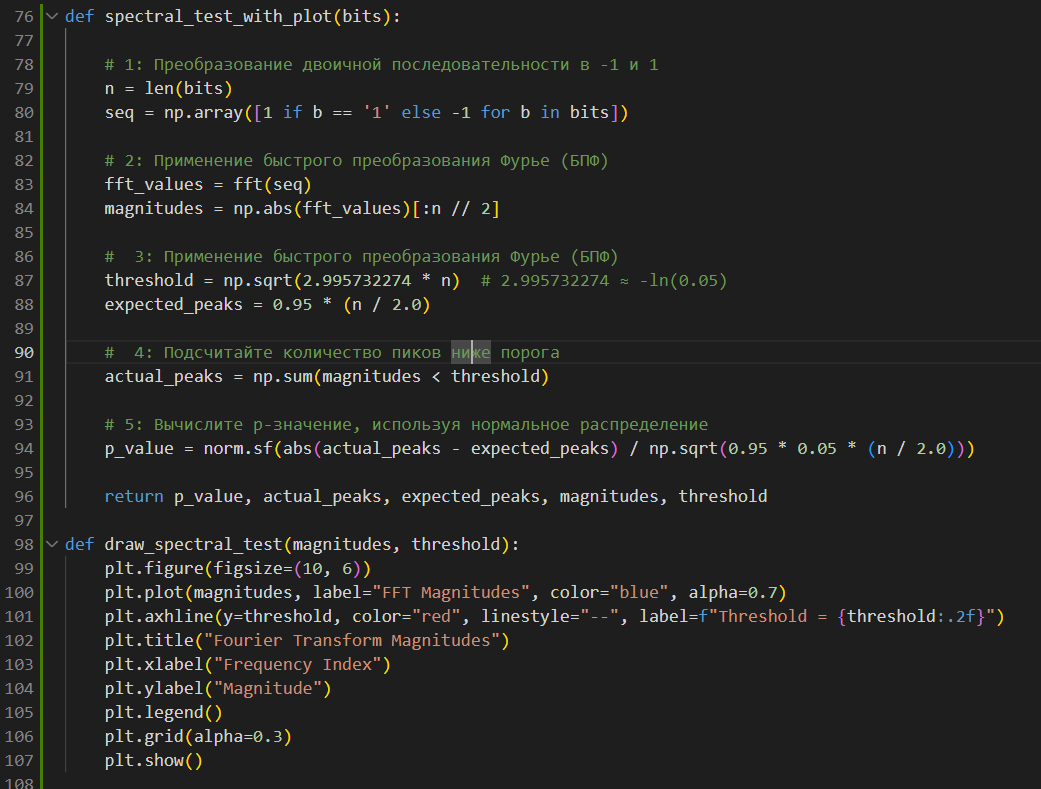


Рисунок 15 – программная реализация спектрального теста и код для создания графика преобразования Фурье

Все десятичные последовательности чисел были преобразованы в двочиные для прохождения тестов следующим образом: взять младший бит каждого числа и соединить их в одну последовательность.

После того как числа были преобразованы в двоичные последовательности. Провели выше описанные тесты и посчитан р уровень значимости. Уровень значимости – это вероятность того, что подопытный генератор произведет последовательность не хуже, чем гипотетический истинный. Если P­значение = 1, то наша последовательность идеально случайна, а если оно = 0, то последовательность полностью предсказуема. В дальнейшем P-значение сравнивается с α, и если она больше α, то нулевая гипотеза принимается и последовательность признается случайной. В противном случае – отбраковывается. В качестве порога α было выбрано значение 0,05. Результаты тестов приведены на рисунках 16 – 19.

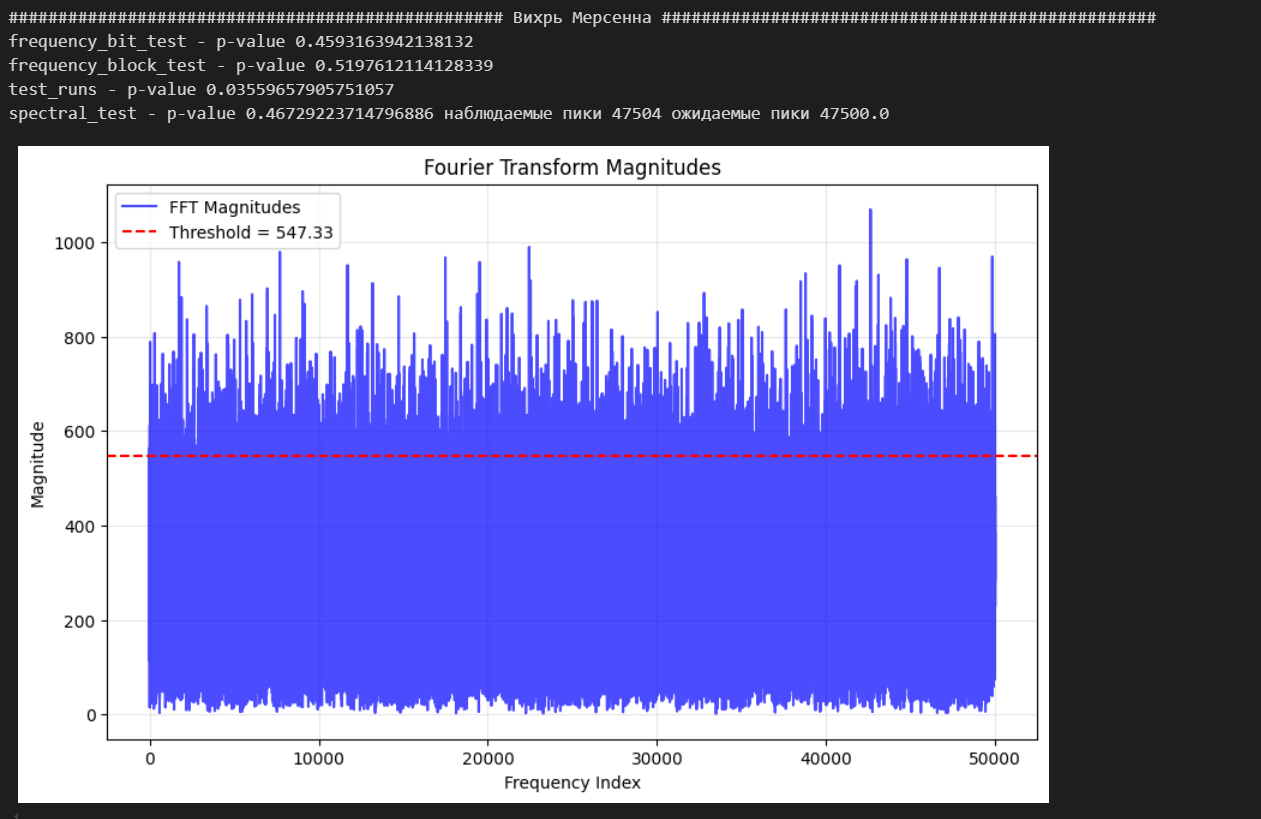


Рисунок 16

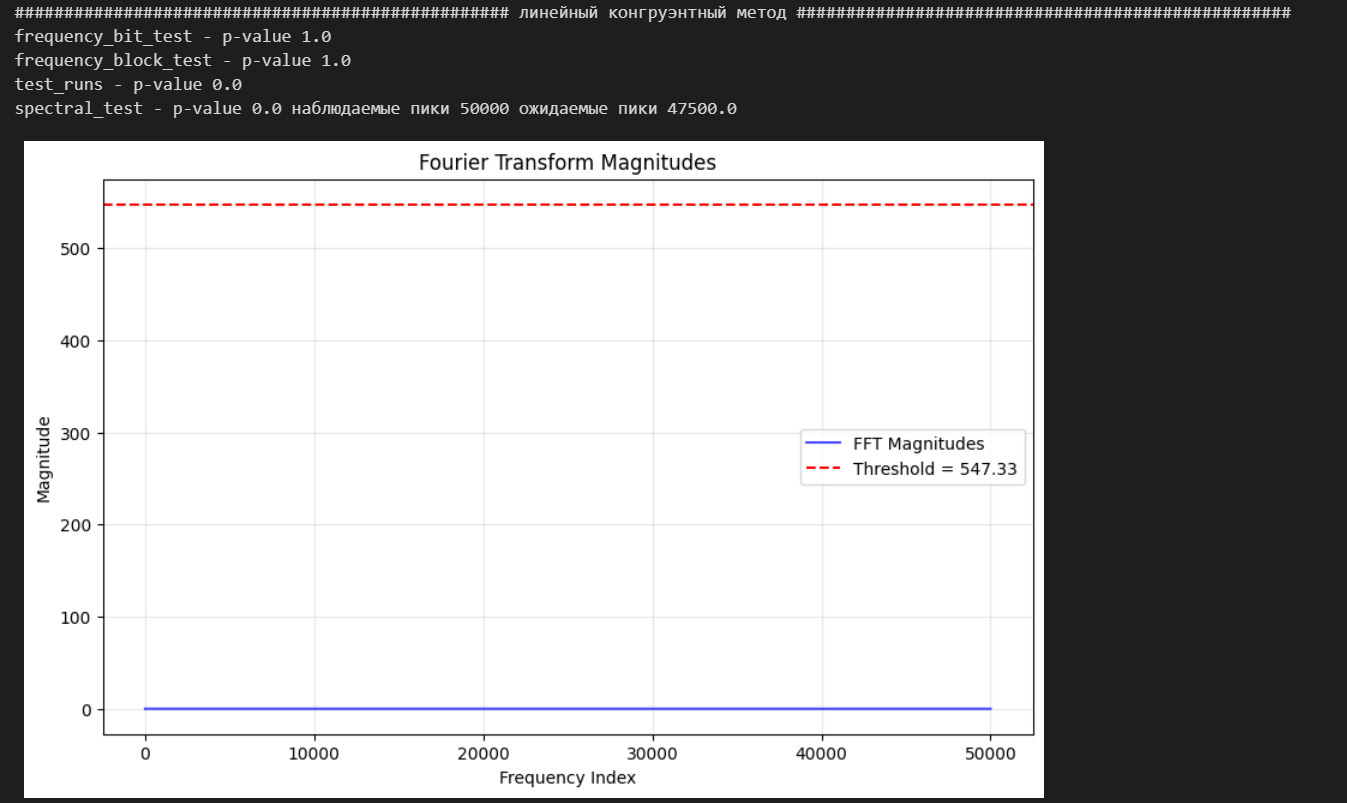


Рисунок 17

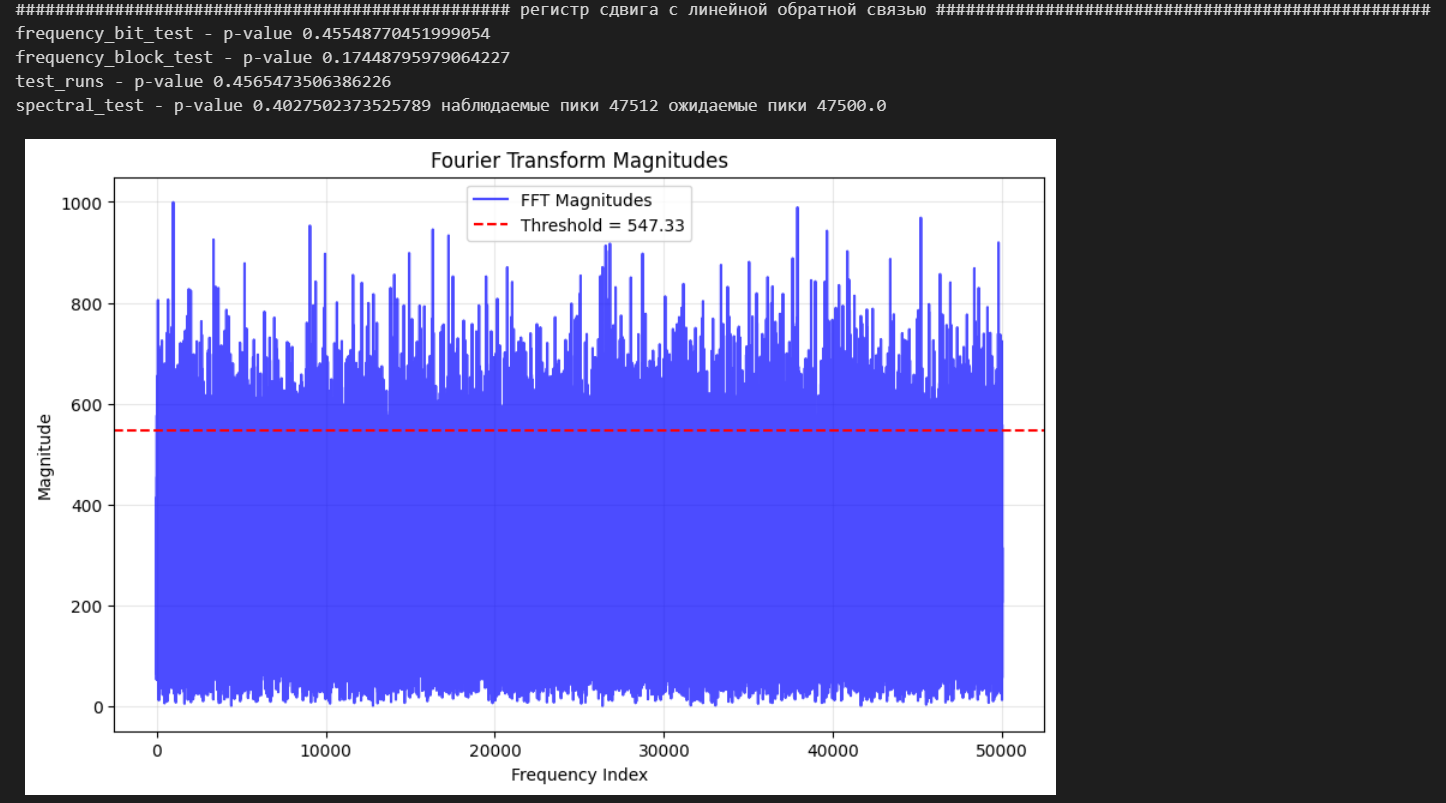


Рисунок 18

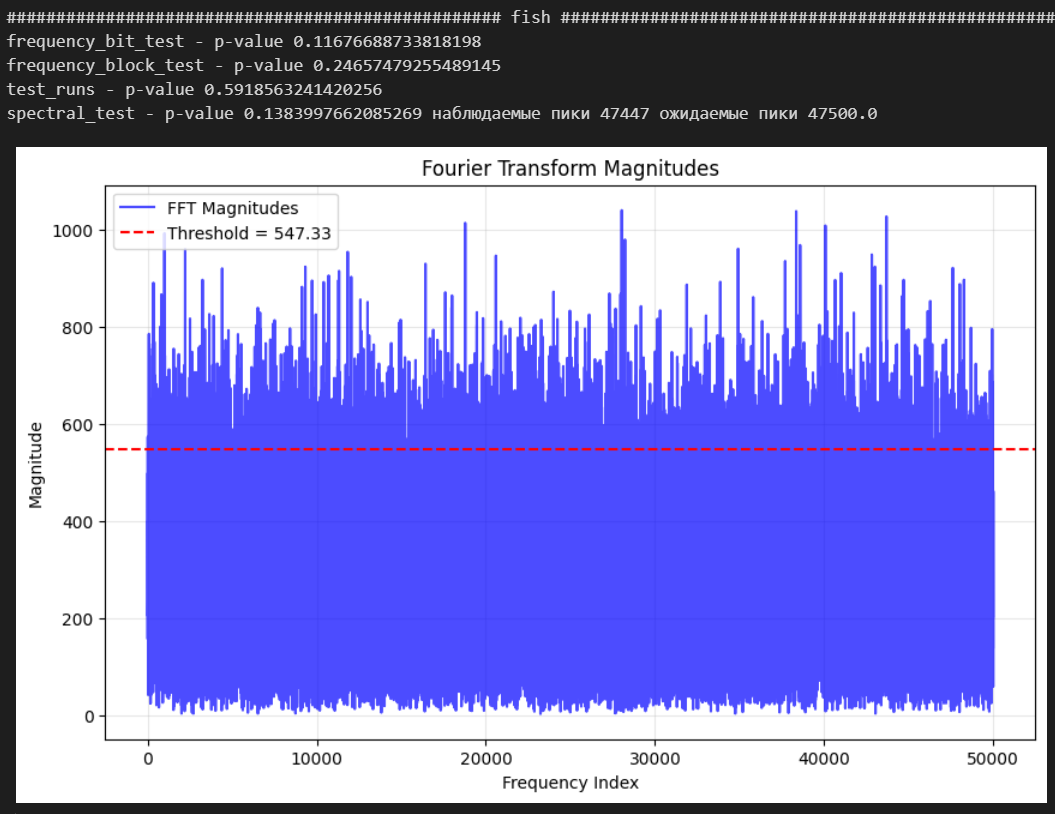


Рисунок 19

Выводы: Были рассмотрены некоторые генераторы псевдослучайных чисел, а также некоторые тесты из стандарта NIST. Почти все генераторы, кроме линейного конгруэнтного прошли написанные тесты, т. к. p-уровень значимости оказался больше 0.05, что означает близость к истинно-случайной последовательности по нормам NIST. На некоторых графиках, например на графике зависимости (*i*+1)-го значения от *i*-го значения у генератора на регистре сдвига были замечены скопления точек в кластерах. Гистограмма разностей соседних элементов тоже выглядит необычно в случае с генератором на регистре сдвига.

[1] Слеповичев, И. И. Генераторы псевдослучайных чисел /– Саратов: СГУ, 2017

[2] [Статистическая проверка случайности двоичных последовательностей методами NIST / Хабр](https://habr.com/ru/companies/securitycode/articles/237695/)

[3] Ковалевский А. П. статистические критерии / Учебное пособие Новосибирск; 2022