МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Исследование и реализация алгоритмов статистического моделирования»

Обучающийся Т.А. Зубов _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.ф.-м. н., доцент, Г.А Тырыгина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант Е.В. Косс
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Тема выпускной квалификационной работы: Исследование и реализация алгоритмов статистического моделирования.

Работа была выполнена студентом Тольяттинского Государственного Университета, института математики, физики и информационных технологий, группы ПМИб-1802б, Зубовым Тимофеем Александровичем.

Выпускная квалификационная работа посвящена исследованию методов генерации случайных последовательностей чисел.

Выпускная работа разделена на несколько логически связанных частей: изучение методов построения последовательностей случайных чисел, описание критериев для оценки качества реализуемых генераторов на основе методов, а также реализация и тестирование этих генераторов.

Задачи работы:

- изучить алгоритмы получения псевдослучайных чисел;
- реализовать генераторы с помощью методов;
- проверить качество генераторов;
- сравнить полученные результаты и сделать вывод.

В заключении хотелось бы подчеркнуть, что данная работа актуальна для решения задач. Задачи относятся к статистическому моделированию, защите информации в ЭВМ и сетях, а также для моделирования физической и игровой реальности, для придания действиям игрового искусственного интеллекта элемента спонтанности. Для решения этих задач необходимо вырабатывать огромные количества случайных чисел с самыми разнообразными свойствами

Abstract

The title of the graduation work is «Research and implementation of statistical modeling algorithms».

The work is performed by Timofey Zubov, a student of Togliatti State University, Institute of Mathematics, Physics and Information Technology, group PMIb-1802b.

The graduation work is devoted to the research of methods for generating random sequences of numbers.

The graduation work is divided into several logically connected parts which are: research of methods for constructing sequences of random numbers, description of criteria for evaluating the quality of implemented generators based on methods, as well as the implementation and testing of these generators.

The goals of the work are as follows:

- to explore algorithms for obtaining pseudorandom numbers;
- to implement generators by using methods;
- to check the quality of generators;
- to compare the results and make a conclusion.

In conclusion we'd like to stress that this work is relevant to solving the tasks. The tasks relate to statistical modeling, information protection in computers and networks, as well as to modeling physical and game reality, to give the actions of game's artificial intelligence an element of spontaneity. To solve these tasks, it is needed to generate huge amounts of random numbers with a wide variety of properties.

Содержание

Введение

Статистическое моделирование – область математической науки, а также аналитические методология, для которой кардинальным является понятие случайной величины. Двумя наиболее важными свойствами таких величин являются случайность и непредсказуемость, о которых следовало бы рассудить по отдельности.

Первое – случайность – обретается при помощи задания некоторых статистических параметров, которые суть два критерия оценки величин в качестве случайных: однородность, то есть презумпция некоторой условно равной частоты появления каждого значения; второе – независимость – указывает на невозможность коррелирования значений последовательности между собой.

«С известной долей условности можно сказать, что не существует такого теста, что позволит нам рассудить о числах последовательности как о независимых; с другой стороны, мы имеем возможность проводить статистическое тестирование, притом формализованное в качестве некоторой стратегии, которое позволит "ухватить" взаимную зависимость.»[14]

Второе свойство — непредсказуемость. При условно "верной" трактовке случайной последовательности каждое число статистически независимо от другого. Следующий элемент последовательности, таким образом, никак не может быть предсказан исходя из значения предыдущего. Настоящая случайная последовательность — невоспроизводима. Эксперимент генерации таких последовательностей абсолютно всегда будет выдавать принципиально иной результат.

На практике, однако, источников подобных последовательностей обнаружено не очень много. Одним из наиболее "очевидных" источников могут стать, к примеру, детекторы ионизирующей радиации, на основании которых можно "ухватить" квантовое состояние частицы, чей спин и прочие параметры будут истинно случайными. Конечно, можно предположить, что

каталогизация псевдослучайных величин может быть источником того же рода, однако эта интуиция порочна: мы, как операторы каталогов, можем с некоторой долей вероятности предположить последовательность каталогов.

Математический анализ при всем объеме своего инструментария не дает возможности до предела точно описать и составить последовательность "идеально" случайных чисел, поскольку значения последовательности могут быть неявным способом коррелированы. Однако подобный ригоризм не кажется необходимым: Генератор псевдослучайных чисел (последовательность которых велика до той меры, что коррелированность на сколь-нибудь исследованном участке не может быть обнаружена) достаточен для, к примеру, генерации практически непредсказуемых событий в видеоиграх.

ГПСЧ – алгоритм, работа которого возможна при наличии двух условий: семени и алгоритма. Первоначально в качестве алгоритмов ГПСЧ применялись линейные конгруэнтные генераторы. Это рекурсивные алгоритмы, которые генерируют следующую величину на основе предыдущей. Следующим логическим шагом развития линейного генератора стал так называемый "вихрь Мерсенна". Хотя псевдослучайные числа на самом деле не являются случайными, существует множество причин их использования. Во-первых, они быстры для воспроизводства. Кроме того, поскольку ГПСЧ - это просто алгоритмы, их можно проверить и всегда быть уверенным, что они работают правильно.

Эта простота – причина, по которой большинство языков использует ГПСЧ. С другой стороны, этот способ не лишён недостатков. Важнейший из них – коррелированность на предельно крупной последовательности. Эта уязвимость, к слову говоря, используется игроки, которых интересует скоростное прохождение видеоигр. Просчитав вероятность наступления конкретного события, нечестные пользователи контента могут использовать информацию для своих нужд.

Кроме того, помимо безобидных манипуляций с игровым контентом,

подобные злоумышленники имеют возможность создать ключи RSA в сертификатах TLS, и в этом случае необходим более надёжный способ получения случайных чисел.

«Но бывают случаи, когда предсказание случайных чисел имеет гораздо большее значение. Например, создание ключей безопасности.

Если злоумышленник выяснит исходное значение, используемое для создания ключей RSA в сертификатах TLS, он потенциально может расшифровать сетевой трафик. Это означает, что он сможет получать пароли и другую личную информацию, передаваемую через интернет. В таких ситуациях нужен более безопасный способ получения случайных чисел. Истинно случайные числа непредсказуемы и не следуют шаблону.»[1]

«Но как компьютер может достичь этого? Как было указано ранее, компьютерам требуется внешний источник случайности. В случае ГПСЧ, это было семя. В случае ГИСЧ, это энтропия. Теория хаоса и термодинамика выходят далеко за рамки данной статьи. Достаточно сказать, что энтропия — это чистый нефильтрованный хаос. И лучший источник этого хаоса—сам компьютер.»[21]

Компьютер не может функционировать случайным образом, но составляющие его части—могут. Компьютер—это сложная система со множеством движущихся частей и изменчивостью. Тепловой шум, фотоэффект и другие квантовые явления происходят постоянно.

Но хаос был бы бесполезен, если бы его нельзя было обуздать. К счастью, инженеры-аппаратчики придумали, как это сделать. Посредством сложной схемы аппаратных микросхем и компонентов, компьютеры могут преобразовывать этот физический шум в цифровые нули и единицы.

Один из наиболее очевидных вариантов применения ГИСЧ—цифровые азартные игры. Бросание костей, перетасовка карт и вращение колеса рулетки —все это зависит от того, чтобы быть неопределимым. Хотя он часто используется при опросах общественного мнения, призыве на военную

службу и отборе присяжных, а также там, где случайность используется как метод справедливости.

В действительности сферы применения ГИСЧ довольно ограничены, поскольку у них есть свой набор недостатков. Во-первых, они медлительны. Хотя это варьируется, ГИСЧ может выдавать только небольшое количество бит в секунду.

«Кроме того, ГИСЧ не всегда надежны. Компьютерам требуется достаточное количество энтропии для производства истинно случайных чисел. Но суть случайности в том, что она возникает случайным образом. Простаивающий или новый сервер не сможет генерировать числа настолько же качественно, как активный.»[18]

1 Алгоритмы получения псевдослучайных чисел

1.1 История появления первых генераторов случайных чисел и их последовательностей

Генераторы случайных чисел были изобретены до того, как появились символы для записи чисел, и задолго до механических и электронных компьютеров. Все основные цивилизации на протяжении веков испытывали потребность в случайном отборе по разным причинам. Сегодня генераторы случайных чисел, особенно на компьютерах, являются важным (хотя и часто скрытым) компонентом человеческой деятельности. В этой статье мы приводим исторический отчет о разработке, внедрении и тестировании генераторов однородных случайных чисел, используемых для моделирования.

У римлян уже был простой метод генерации (приблизительно) независимых случайных битов. Подбрасывание монеты для выбора между двумя исходами было тогда известно, как “navia aut caput”, что означает “лодка или голова” (на их монетах с одной стороны был корабль, а с другой - голова императора). Игральные кости были изобретены гораздо раньше: в Ираке и Иране были найдены кости возрастом около 5000 лет. Они также были популярны в Индии и Китае по крайней мере 4000 лет назад, а затем вокруг Средиземного моря, в частности в Египте (Карр 2017, Глимне 2017).

На этих кубиках было от 1 до 6 точек с каждой стороны, как и сегодня. У них не было письменных чисел, потому что письменные числа еще не были изобретены. То есть люди изобрели генераторы случайных чисел раньше, чем символы для представления чисел! Другие типы устройств для получения случайных результатов были обнаружены практически в каждой части мира.

Всем нужны случайные числа. Интересно, что в большинстве древних цивилизаций люди не верили в случайность; они верили, что игральные кости (или другое устройство) мы сообщали им решение бога. В некотором смысле они “поняли”, что их случайные числа не были на самом деле случайными.

«Около 127 лет назад Гальтон (1890) разработал метод выборки

(приблизительно) из заданного распределения вероятностей с использованием игральных костей. Он использовал кубические кости (с шестью гранями), но после броска кости он брал каждый кубик вручную и размещал его выровненным перед собой, закрыв глаза, и рассматривал ориентацию верхней грани. Это дает 24 возможных результата на кубик (почти 4,6 случайных бита).»[24]

В наши дни бросать кости и вводить результат на компьютере, очевидно, было бы слишком медленно. Первая идея состоит в том, чтобы переработать цифры. «До появления компьютеров статистики, которым нужны были случайные числа для случайной выборки и вероятностных экспериментов, решили составить и опубликовать большие таблицы так называемых чисел случайной выборки.»[26] Это были дни, когда можно было опубликовать научную статью или даже книгу, заполненную только случайными цифрами! Типпетт (1927) опубликовал первую таблицу с 41 600 “случайные” цифры взяты из отчета о переписи населения 1925 года, очевидно, по предложению Карла Пирсона. За ним последовали и другие, в основном в конце 1930-х и в 1940-х годах. Например, Фишер и Йейтс (1938) опубликовали таблицу, построенную путем выбора цифр из большой таблицы логарифмов с точностью до 20 знаков после запятой.

Статистическое тестирование вскоре выявило очевидные проблемы со многими из этих таблиц; в частности, некоторые цифры появлялись слишком часто, а другие - недостаточно. В своем знаменательном вкладе Кендалл и Бабингтон-Смит (1938) изучили философские и практические проблемы, возникающие при построении таблиц чисел случайной выборки, и различие, которое должно провести различие между критериями качества для этих конечных таблиц и понятием независимых случайных цифр в теории вероятностей. В рамках теории вероятностей каждая возможная таблица (для данного размера) имеет одинаковую вероятность появления, поэтому можно утверждать, что ни одна таблица не лучше любой другой! Чтобы справиться с этим очевидным парадокс, Кендалл и Бабингтон-Смит (1938) ввели понятие

локально случайной конечной последовательности, свободно определяемое как означающее, что каждая достаточно длинная подпоследовательность должна выглядеть случайной и проходить определенный набор простых статистических тестов на случайность. Обратите внимание, что действительно случайные бесконечные последовательности не удовлетворяют этому критерию; они содержат произвольные длинные (редкие) подпоследовательности, которые повторяют только одну и ту же цифру, например.

Очевидно, что можно выбрать только ограниченное количество тестов из астрономического числа из возможных тестов, которые могут быть применены к конечной последовательности заданной длины. Кендалл и Бабингтон-Смит (1938) предложили четыре типа статистических тестов для цифр; первый проверяет только однородность цифр, а три других также проверяют независимость: частотный тест, в котором наблюдаемая частота каждой цифры сравнивается с ее ожидаемой (теоретической) частотой; последовательный тест, который измеряет частоты для пар последовательных цифр; покерный тест, который подсчитывает частоты определенных блоков из 5 цифр; и тест на разрыв, который отслеживает количество позиций между последовательными появлениями любой заданной цифры и подсчитывает частоту каждого размера разрыва. В каждом случае наблюдаемые частоты сравниваются с ожидаемыми (теоретическими) с помощью статистики хи-квадрат.

1.2 Устройство образования последовательностей случайных чисел

Генераторы псевдослучайных последовательностей бывают физическими которые базируются на использовании физических явлений (например, шумов электронных устройств, радиоактивного излучения и др.). Во время применения физических генераторов случайный электронный сигнал преобразуется в двоичный код, который вводится в компьютер с

помощью специальных аналого-цифровых преобразователей.

Также генераторы могут быть табличными, примером которых может являться опубликованная корпорацией “Ренд” в 1955 году таблица, которая содержала миллион значений. Для заполнения этих таблиц применялись аппаратные методы. Данные этих таблиц можно использовать при моделировании систем с помощью метода статистических испытаний.

И наиболее популярный на практике способ получения таких чисел является программный вид генераторов.

Каждый из возможных генераторов, построенных на основе разных алгоритмических методов построения случайных, а также псевдослучайных последовательностей имеет свои преимущества и недостатки.

«К примеру, если физические генераторы создают последовательности истинных случайных чисел, то программные как правило создают последовательности псевдослучайных чисел.»[28]

Природа предлагает некоторые источники случайности, но их использование очень дорого.

В макроскопических средах существует несколько источников, таких как изменения климата или космический микроволновый фон. Такого рода генераторы очень дороги: представьте, что вы покупаете систему, которая соединяет вас со спутником, который измеряет колебания СМВ и отправляет результат обратно через несколько миллисекунд. Стоимость составляет около 400-500 миллионов долларов США.

В микроскопическом мире вся окружающая среда управляется хаосом: насколько мы знаем, это вероятностный мир; но опять же, управление такого рода источниками обходится очень дорого. Идеально сбалансировать квантовое состояние частицы - непростая задача.

По этим причинам мы всегда и удобно встраиваем генератор в наши машины (компьютеры, смартфоны, телевизоры и т.д.). Кроме того, всегда полезно иметь более компактный способ вычисления случайной строки: если ваша система извлекает последовательность из локальной температуры в

МКК, любой может воспроизвести ту же последовательность, расположив датчик рядом с вашим; или даже любой может манипулировать обнаружениями и контролировать ваши последовательности.

Компьютер использует центральный процессор для выполнения инструкций, и центральный процессор основанный на детерминированном механизме. Как мы можем создать случайность из среды, в которой нет никакого источника случайности?

Прежде чем мы начнём разбираться в этом вопросе, давайте определим генератор псевдослучайных чисел (ГСПЧ). Отсюда я буду рассматривать ГСПЧ, которые работают с битом (0 и 1), но очень легко проверить его свойства для других случаев, поскольку можно закодировать двоичную последовательность в число.

В этой работе будут рассмотрены следующие методы: линейный конгруэнтный метод, квадратичный конгруэнтный метод, метод сдвигового регистра и самоуправляемый 2-линейный регистр сдвига, метод серединных квадратов.

1.3 Линейный конгруэнтный метод

$$X_{n+1} = (aX_n + c) mod m \quad (1)$$

«Где m – модуль, m >0;

a – множитель, $0 \leq a < m$;

c – приращение, $0 \leq c < m$;

X_0 – начальное значение, $0 \leq X_0 < m$.

Последовательность, полученная по формуле (1) называется линейной конгруэнтной последовательностью. Конгруэнтная последовательность всегда образует циклы. Это свойство является общим для всех последовательностей вида $X_K = f(X_k - 1)$, где f преобразует конечное множество само в себя. Повторяющиеся циклы называются периодами. Это

похоже на рулетку, где крупье вращает колесо с 37 карманами в одном направлении, затем вращает шар в противоположном направлении по наклонной круговой дорожке, проходящей по окружности колеса. Мяч в конце концов замедляется и приземляется в одном из m = 37 карманов. 0 ≤ a < m называется множителем, а 0 ≤ c < m - приращением.

Формулу (1) можно обобщить следующим образом, где $(n+k)$ −ый член последовательности выражается непосредственно через n -ый:

$$X_{n+k} = \left(a^k X_n + (a^k - 1)\frac{c}{b}\right) mod m, k \geq 0, n \geq 0 \qquad (2)$$

Выбор модуля, множителя и приращения.

$$b = a - 1$$

При выборе модуля m необходимо учитывать 2 фактора:

- число m должно быть достаточно большим, так как период последовательности не может быть больше, чем m;
- нужно подобрать значение m так, чтобы $(aX_n + c) mod m$ вычислялось быстро.»[2]

Теорема 1.

«Линейная конгруэнтная последовательность, определенная числами m, $a, c,$ и $X_0,$ имеет период максимальной длины m, тогда и только тогда, когда число c и m взаимно простые.

$b = a - 1$ кратно p для каждого простого p, являющегося делителем m.

b кратно 4, если m кратно 4.

Чтобы доказать теорему рассмотрим вспомогательные теоретико-числовые результаты.

Лемма 1.

Пусть p - простое число, а e - положительное целое число, такое что, $p^e > 2$. Если

$$x \equiv 1(mod\ p^e), x \neq 1(modp^{e+1}) \quad (3)$$

то

$$x^p \equiv 1(mod\ p^{e+1}), x^p \neq 1(mod\ p^{e+2} \quad (4)$$

Лемма 2.

Пусть число m допускает разложение на простые множители в виде

$$m = p_1^{e_1} \dots p_t^{e_t} \quad (5)$$

Длина λ линейной конгруэнтной последовательности, определенной параметрами (X_0, a, c, m), является наименьшим общим кратным длин λ_j периодов линейных конгруэнтных последовательностей

$$\left(X_0 mod p_j^{e_j}, a\ mod\ p_j^{e_j}, c\ mod\ p_j^{e_j},\ p_j^{e_j}\right), 1 \leq j \leq t \quad (6)$$

Теперь мы можем доказать теорему 1. Из леммы 2 следует, что теорему достаточно доказать для случая, когда m является степенью простого числа, поскольку

$$p_1^{e_1} \dots p_1^{e_t} = \lambda = HOK(\lambda_1, \dots, \lambda_t) \leq \lambda_1 \dots \lambda_t \leq p_1^{e_1} \dots p_t^{e_t} \quad (7)$$

выполняется тогда и только тогда, когда $\lambda j\ = {p_j}^{e_j}$ для $1 \leq j \leq t$.

Предположим, поэтому, что $m = p^e$, где p - простое число, а e - целое положительное число. Так как утверждение теоремы 1 очевидно при $a\ =\ 1$, то можно считать, что $a\ >\ 1$. Период может иметь длину m тогда и только тогда, когда каждое целое число x , такое что $0 < x < m$, встречается в этом

периоде. Действительно, никакое значение x не может встретиться в периоде более одного раза. Таким образом, период последовательности имеет длину m тогда и только тогда, когда период последовательности с начальным значением $X_0 = 0$ имеет период длинной m. Поэтому достаточно доказать теорему, когда $X_0 = 0$. Справедлива следующая формула:

$$X_n = \frac{(a^n-1)}{(a-1)} mod m \tag{8}$$

Если c и m не взаимно простые числа, то значение X_n никогда не может быть равно 1. Следовательно, условие 1 теоремы необходимо. Период имеет длину m тогда и только тогда, когда наименьшее положительное значение n, для которого $X_n = X_n = 0$, равняется $n = m$. Из (4) и условия 1 теоремы следует, что для доказательства теоремы необходимо доказательства следующего утверждения.

Лемма 3.

Предположим, что $1 < a < p^e$, где p - простое число. Если λ-наименьшее целое положительное число, для которого $(a^\lambda - 1)/(a-1) \equiv 0(mod\ p^e)$, то

$\lambda = p^e$ тогда и только тогда, когда $\begin{cases} a \equiv 1(\text{mod } p), \text{где} & p > 2, \\ a \equiv 1(\text{mod } 4), \text{где} & p = 2. \end{cases}$

Доказательство:

Предположим, что $\lambda = p^e$. Если $a \neq 1(mod\ p)$, то $(a^n - 1)/(a-1) \equiv 0(mod\ p^e)$ тогда и только тогда, когда $(a^n - 1) \equiv 0(mod\ p^e)$. Значит условие $a^{p^e} - 1 \equiv 0(mod\ p^e)$ влечет равенство $a^{p^e} \equiv 1\ (mod\ p)$, но $a^{p^e} = a(mod\ p)$. Таким образом, предположение, что $a \neq 1(mod\ p)$, приводит к противоречию. Если $p = 2, a = 3\ (mod\ 4)$, то

$$(a^{2^{e-1}} - 1)/(a-1) \equiv 0(mod\ 2^e) \tag{9}.$$

Эти рассуждения показывают, что в большинстве случаев необходимо,

чтобы $a = 1 + qp^f$, где $p^f > 2$ и q не кратны p, когда $\lambda = p^e$.

Докажем достаточность. По лемме 1, для всех $g > 0$:

$$a^{p^g} \equiv 1(mod\ p^{f+g}), a^{p^g} \neq 1(mod\ p^{f+g+1}) \quad (10)$$

следовательно

$$(a^{p^g} - 1)/(a-1) \equiv 0(mod\ p^g) \quad (11)$$

$$(a^{p^g} - 1)/(a-1) \neq 0\ (mod\ p^{g+1}) \quad (12)$$

В частности, $(a^{p^g} - 1)/(a-1) \equiv 0(mod\ p^e)$. Тогда для конгруэнтной последовательности, определяемой параметрами $(0, a, 1, p^e)$ X_n, справедливо $X_n = (a^n - 1)/(a-1) \equiv 0(mod\ p^e)$. Значит, ее период равен λ, то есть $X_n = 0$ тогда и только тогда, когда n кратно λ . Следовательно, p^e кратно λ. Это возможно только, если $\lambda = p^g$ для некоторый g и соотношения (5) означают, что $\lambda = p^e$.»[25]

1.4 Квадратичный конгруэнтный метод

«Этот алгоритм генерации псевдослучайной последовательности $x_t \in A = \{0.1, \dots, N-1\}$ определяется квадратичным рекуррентным соотношением:

$$x_{t+1} = (dx_t^2 + ax_t + c) mod N, t = 0,1 \quad (13)$$

Где $x_0, a, c, d \in A$ - параметры генератора. Выбор этих параметров осуществляется на основе следующих двух свойств последовательности (11).

Свойство С1. Квадратичная конгруэнтная последовательность (11)

имеет наибольший период $T_{max} = N$ тогда и только тогда, когда выполнены следующие условия:

1) c, N - взаимно простые числа;

2) $d, a - 1$ - кратны p, где p - любой нечетный простой делитель N.

3) d - четное число, причем

$$d = \begin{cases} (a-1)mod4, \text{если } N \text{ кратно } 4, \\ (a-1)mod2, \text{если } N \text{ кратно } 2 \end{cases}$$

4) если N кратно 9, то$d\ mod9 = 0$, либо $d\ mod9 = 1$ и $cd\ mod9 = 6$

Свойство С2. Если $N = 2^q,\ q \geq 2$, то наибольший период $T_{max} = 2^q$ тогда и только тогда, когда c - нечетно, d - четно, a - нечетное число, удовлетворяющее соотношению $a = (d+1)mod\ 4$»[9]

1.5 Линейный сдвиговый регистр с обратной связью

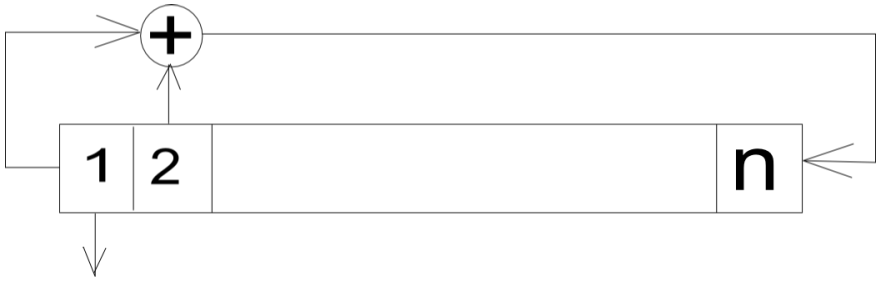

Рисунок 1 – вид сдвигового регистра

«Регистры сдвига или сдвиговые регистры (англ. shift register) выглядят как последовательно соединенная цепочка триггеров как на рисунке 1. Основной режим их работы - это сдвиг разрядов кода, записанного в эти триггеры. То есть по тактовому сигналу содержимое каждого предыдущего триггера переписывается в следующий по порядку в цепочке триггеров. Код, хранящийся в регистре, с каждым тактом сдвигается на один разряд в сторону старших разрядов или в сторону младших разрядов.»[3]

Сдвиговый регистр с обратной связью состоит из двух частей:

сдвигового регистра и функции обратной связи. Длина сдвигового регистра - количество битов. Когда нужно извлечь бит, все биты сдвигового регистра сдвигаются вправо или влево на одну позицию. Новый крайний слева бит определяется функцией остальных битов регистра. На выходе сдвигового регистра оказывается один, обычно младший, значащий бит. Период сдвигового регистра - длина получаемой последовательности до начала ее повторения. Для LFSR функция обратной связи представляет собой сумму по модулю 2 (xor) некоторых битов регистра (эти биты называются отводной последовательностью). LFSR может находиться в $2^n - 1$ внутренних состояниях, где n - длина сдвигового регистра. Если сдвиговый регистр заполнен нулями, то такое состояние будет порождать на выходе только нули (так как в качестве функции обратной связи используется xor), поэтому такое состояние бесполезно. Теоретически LFSR может генерировать последовательность с длиной бит $2^n - 1$, так как длина последовательности совпадает с количеством внутренних состояний. LFSR будет проходить все внутренние состояния (иметь максимальный период) только при определённых отводных последовательностях - если многочлен, образованный из отводной последовательности и константы 1, является примитивным по модулю 2, степень многочлена - длина сдвигового регистра. Примитивный многочлен степени n - это неприводимый многочлен, который является делителем $x^{2^n-1} + 1$, но не является делителем $x^d + 1$ для всех d, делящих $2^n - 1$.

«Преимущества генератора ПСП, основанного на сдвиговом регистре: эффективная аппаратная реализация; быстродействие.

Недостатки: генераторы на основе регистров образуют только циклические последовательности чисел. Для получения нециклических последовательностей необходимо использовать дополнительный комбинационный преобразователь кодов, включаемый на выходе генератора. Но при этом основные параметры генератора - быстродействие, мощность, площадь кристалла - ухудшаются.»[11]

1.6 Самоуправляемый 2-линейный регистр сдвига

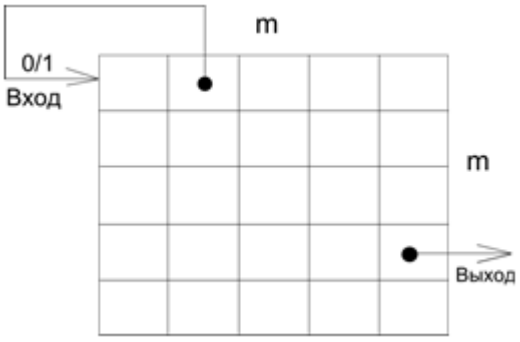

Рисунок 2 – вид матрицы

«Возьмем 2 неприводимых многочлена $F_1(x)$ и $F_2(x)$,причем $degF_1(x) = degF_2(x) = m$, и составим соответствующую матрицу размером m*m как на рисунке 2.

На вход генератора подается либо «0», либо «1».

Если на вход генератора подается «0», то все строки генератора преобразуются в соответствии с заданным законом рекурсии $F_1(x)$.

Если на вход генератора подается «1», то все столбцы матрицы преобразуются в соответствии с законом рекурсии $F_2(x)$.

Начальное заполнение генератора – любое, кроме нулевого.»[22]

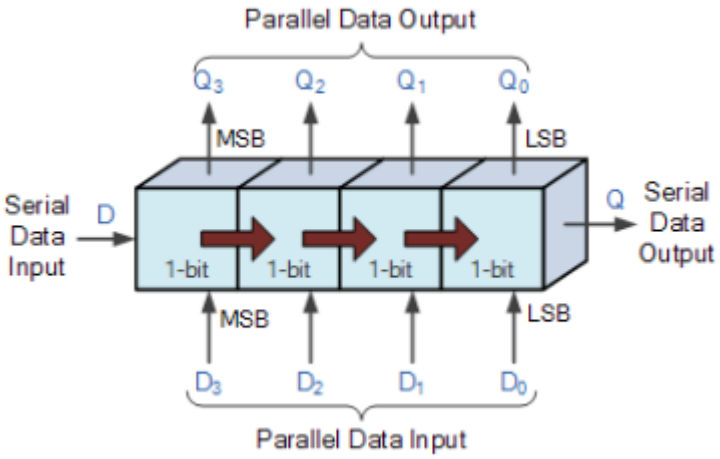

Рисунок 3 – схема работы алгоритма

Это последовательное устройство загружает данные, присутствующие на его входах, а затем перемещает или “сдвигает” их на свой выход один раз за каждый такт, отсюда и название Сдвиговый регистр.

Регистр сдвига как показано на рисунке 3 в основном состоит из нескольких однобитовых “защелок данных D-типа”, по одной для каждого бита данных, либо логического “0”, либо “1”, соединенных вместе в последовательной схеме последовательного типа, так что выход из одной защелки данных становится входом следующей защелки и так далее.

Биты данных могут вводиться или выводиться из сдвигового регистра последовательно, то есть один за другим либо с левого, либо с правого направления, либо все вместе одновременно в параллельной конфигурации.

Количество отдельных защелок данных, необходимых для создания одного устройства сдвигового регистра, обычно определяется количеством битов, которые должны быть сохранены, причем наиболее распространенными являются 8-битные (один байт), построенные из восьми отдельных защелок данных.

Регистры сдвига используются для хранения данных или для перемещения данных и поэтому обычно используются в калькуляторах или компьютерах для хранения данных, таких как два двоичных числа, прежде чем

они будут сложены вместе, или для преобразования данных из последовательного в параллельный или параллельно последовательный формат. Все отдельные защелки данных, составляющие единый регистр сдвига, управляются общим тактовым сигналом (Clk), что делает их синхронными устройствами.

1.7 Метод серединных квадратов

«Один из первых алгоритмических методов получения равномерно распределенных псевдослучайных чисел получил название "метод середины квадрата". Он был предложен Джоном фон Нейманом в 1946 году и заключается в следующем:

1. Выбрать начальное случайное число X_0имеющее n-разрядное представление (возможно полученное от внешнего источника энтропии).

2. Возвести это число X_iв квадрат, в результате чего, мы получим 2n-разрядное число Y_i.

3. Следующее число X_{i+1}получим, составив его n-разрядное представление, выбрав средние n разрядов из числа Y_i.

Например, если начальное число X_0=3485, то $Y_1=3485^2$= 12145225, X_1= 1452, а X_2=1083, и т.д.

В качестве начального числа для этого алгоритма часто берут рациональное число в десятичной записи. Например, X_0 = 0,3485, X_1 = 0,1452, X_2=0,1083 и т.д.»[17]

Очевидно, что данный метод может быть реализован с помощью операций деления нацело и взятия остатка от деления предыдущего члена последовательности.

Недостаток этого метода — наличие корреляции между числами последовательности, а в ряде случаев случайность вообще может отсутствовать. Например, если X_0 = 0,4500X_1= 0,2500, X_2 = 0,2500, X_3 = 0,2500 и т.д. Кроме того, данный «метод обладает малым периодом и сейчас

представляет интерес лишь в историческом аспекте.»[29]

– Берём N-значное число 5772156649

– Возводим число в квадрат 33317**7923805949**09201

– Из результата (2N-значного числа) берём N средних цифр 7923805949

– Возвращаемся к 1-му шагу.

Таким образом в данной главе были изучены различные методы получения псевдослучайных последовательностей, также разобрано устройство образования этих последовательностей, были построены схематические изображения некоторых методов. С помощью нескольких методов были сформированы примеры последовательностей.

2 Оценка качества

«Качество случайных чисел может быть измерено различными способами. Одним из распространенных методов является вычисление плотности информации, или энтропии, в серии чисел. Чем выше энтропия в ряду чисел, тем сложнее предсказать данное число на основе предыдущих чисел в ряду. Последовательность хороших случайных чисел будет иметь высокий уровень энтропии, хотя высокий уровень энтропии не гарантирует случайности» [6].

Существует два основных критерия, которые учитываются при выборе генератора псевдослучайных чисел. Первый - это период сгенерированной последовательности. В идеале это должно быть так долго, чтобы никогда не повторяться в течение жизненного цикла приложения. Современные генераторы, которые здесь затрагиваются, обычно имеют очень длинные периоды, и при работе на текущих компьютерных тактовых частотах время повторения сопоставимо со временем жизни известной вселенной. В этом смысле период не часто является прямой проблемой.

Несколько отклонений, сгенерированных для того, чтобы сделать поведение игровой программы “интересным” для игрока, не требуют генератора с вызывающе большой длиной повтора. Однако вычисления методом Монте-Карло, которые могут занять недели или месяцы ресурсов суперкомпьютера, должны иметь генераторы с очень длинными периодами. В последние 20-30 лет неуклонно растущей производительности суперкомпьютеров сохраняется интерес ко все более длительным алгоритмам генерации периодов.

Это часто связано с необходимостью большего количества битов, используемых в генераторе. «16-битные генераторы на основе целых чисел позднего 1970-е годы были вытеснены 24-битными (с плавающей запятой) алгоритмами, такими как алгоритм Марсальи с запаздыванием по Фибоначчи, с помощью 64-битного твистора Мерсенна-Твистора на основе целых чисел, а

в совсем недавнее время с помощью 128-битных алгоритмов и даже дольше для криптографически надежной генерации случайных чисел. Однако второй критерий более тонкий и определенно вызывает известную озабоченность у некоторых алгоритмов. Этот вопрос касается того, насколько на самом деле случайны или некоррелированы отклонения в последовательности – в контексте потребностей приложения для вождения. Есть несколько широко распространенных использовались статистические тесты, которые в настоящее время широко распространены и которые представляют лучшие знания исследовательских сообществ о том, что является “достаточно случайным»[7].Приложениям обычно требуются либо случайные целые числа с плоской равномерной вероятностью получения всех значений в пределах заданного диапазона, либо равномерные отклонения с плавающей запятой в диапазоне [0, 1), опять же с равномерным распределением вероятностей по всему диапазону. Как правило, если у кого-то есть генератор, который производит в любом из них можно построить отклонения от более сложных распределений с помощью подходящих алгоритмов преобразования.

Устройство для реализации генераторов псевдослучайных чисел обычно приводит к необработанным отклонениям в одной из этих двух форм - однородных целых чисел или однородных чисел с плавающей запятой, и можно найти способы преобразования одного в другое. В случае отклонений с плавающей запятой можно просто умножить на N, чтобы получить целые числа в диапазоне [0, N), а в случае целых чисел, которые, как известно, находятся в этом диапазоне, можно разделить на N. Различное оборудование обработки будет выполнять эти операции выполняются с разной скоростью в зависимости от тактовой частоты и стандартов с плавающей запятой. Если у кого-то есть случайный источник некоррелированных b-битов , можно легко получить (беззнаковые) целые числа . в подходящем диапазоне [0, 2 b) или [0, 2 в − 1]. Затем можно соответствующим образом разделить, чтобы получить равномерные отклонения с плавающей запятой. Однако обратная операция не так проста. Большинство процессоров используют стандартное битовое

представление IEEE с плавающей запятой для 32-разрядной или 64-разрядной точности. Они определяют знаковый бит, показатель степени и мантиссу, из которых нетривиально получить равномерно несмещенный случайные биты без какой-либо арифметики, которые обязательно должны тратить часть исходных 32 или 64. Это порождает еще один важный критерий для генераторов случайных чисел - в идеале они должны быть «хорошо спроектированы с точки зрения наличия подключаемых программных интерфейсов.»[27]

«Это означает, что код может быть протестирован и реализован с использованием любого количества различных алгоритмов генератора Эффективность статистического моделирования систем на ЭВМ и достоверность получаемых результатов существенным образом зависят от качества исходных (базовых) последовательностей псевдослучайных чисел, которые являются основой для получения стохастических воздействий на элементы моделируемой системы. Поэтому, прежде чем приступать к реализации моделирующих алгоритмов на ЭВМ, необходимо убедиться в том, что исходная последовательность псевдослучайных чисел удовлетворяет предъявляемым к ней требованиям, так как в противном случае даже при наличии абсолютно правильного алгоритма моделирования процесса функционирования моделируемой системы по результатам моделирования нельзя будет достоверно судить о характеристиках системы.

Поэтому все применяемые генераторы случайных чисел должны перед моделированием системы пройти тщательное предварительное тестирование, которое представляет собой комплекс проверок по различным статистическим критериям, включая в качестве основных проверки (тесты) на равномерность, стохастичность и независимость. Кроме того, очень важными показателями качества базовой последовательности являются длина периода и длина отрезка периодичности.» [8] Рассмотрим возможные методы проведения таких проверок.

2.1 Проверка равномерности

«Проверка равномерности последовательностей псевдослучайных квазиравномерно распределенных чисел $\{x_r\}$ может быть выполнена по гистограмме или с использованием косвенных признаков.

Суть проверки по гистограмме сводится к следующему. Выдвигается гипотеза о равномерности распределения чиел в интервале (0 1). Затем интервал (0 1) разбивается на m равных частей. При генерации последовательности РРСЧ подсчитывается количество попаданий N_k в каждый из m подинтервалов. Вычисляется относительная частота попадания случайных чисел последовательности $\{x_r\}$ в каждый из подинтервалов

$C_K = N_K/N,$

где N =$\sum_{k=1}^{m} N_k$− общее количество чисел в последовательности {$x_г$}.

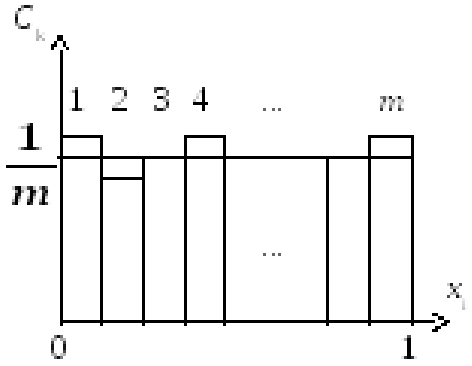

Рисунок 4 - Проверка равномерности по гистограмме

После просмотра рисунка 4 очевидно, что при равномерности последовательности чисел, частоты должны быть близкими при достаточно больших N к теоретической вероятности попадания в подинтервалы, равной $1/m$.»[13]

«Суть проверки равномерности по косвенным признакам сводится к следующему. Вся последовательность $\{x_r\}$ разбивается на пары чисел:

$(x_1, x_2), (x_3, x_4), \ldots, (x_{2i-1}, x_{2i}), \ldots, (x_{N-1}, x_N)$

Затем подсчитывают число пар K, для которых выполняется условие:

$x_{2i-1}^2 + x_{2i}^2 < 1$

Геометрически это означает, что точка с координатами (x_{2i-1}^2, x_{2i}) расположена внутри четверти круга радиуса $R = 1$, вписанного в единичный квадрат.

В общем случае точка (x_{2i-1}, x_{2i}) всегда попадет внутрь единичного квадрата. Тогда теоретическая вероятность попадания этой точки в четверть круга равна отношению площади четверти круга к площади единичного квадрата:

$$P = S_{\frac{1}{4}} \text{круга} / S\text{квадрата} = (\pi R 2/4)/(1 \cdot 1) = \pi/4. \qquad (14)$$

Если числа последовательности $\{x_r\}$ равномерны, то в силу закона больших чисел теории вероятностей при больших N относительная частота попадания точки в единичный квадрат, равная отношению числа K пар (x_{2i-1}, x_{2i}), для которых проверочное условие выполнилось к общему числу $N/2$ пар последовательности должна сходиться к P:»[18]

$$\frac{2K}{N} \to \frac{\pi}{4}$$

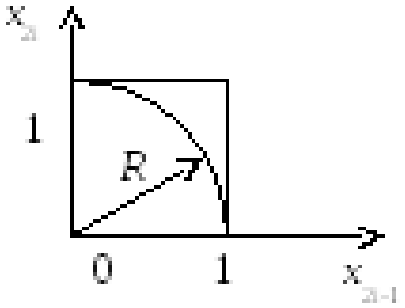

Рисунок 5 - Проверка равномерности по косвенным признакам

«Проверка равномерности распределения элементов последовательности с помощью критерия Колмогорова-Смирнова Критерий Колмогорова-Смирнова как на рисунке 5 используется в случаях, когда объем выборки находится примерно в пределах от 10 до 100 элементов. ».[1]

2.2 Проверка стохастичности

«Это исследование последовательностей псевдослучайных чисел $\{x_r\}$ наиболее часто проводится методами комбинаций и серий. Рассмотрим метод комбинаций.

Сущность метода комбинаций сводится к определению закона распределения закона распределения (появления) числа единиц (нулей) в n-разрядном двоичном числе x_r. На практике длину последовательности N берут достаточно большой и проверяют все п разрядов или только l старших разрядов числа x_r .

Теоретически закон появления j единиц в l разрядах двоичного числа описывается исходя из независимости отдельных разрядов биномиальным законом распределения:

$$P(j,l) = C_l^j P(1)[1-P(1)]^{l-j} = C_l^j P^l(1), \qquad (15)$$

где $P(j,l)$ – вероятность появления j единицв l разрядах числа x_r; $P(1) = P(0) = 0{,}5$ – вероятность появления единицы (нуля) в любом разряде числа x_r $C_l^j = l!/[j!\,(l-j)!]$

Тогда при фиксированной длине выборки N теоретически ожидаемое число появления случайных чисел с j единицами в проверяемых l разрядах будет равно

$$n_j = NC_I^j P' \qquad (16)$$

После нахождения теоретических и экспериментальных вероятностей $P(j,l)$ или чисел n_j при различных значениях $l \leq n$ гипотеза о стохастичности проверяется с использованием критериев согласия.»[19]

2.3 Проверка независимости

«Случайные величины ξ и η называются независимыми, если закон распределения каждой из них не зависит от того, какое значение приняла другая.

Проверка независимости проводится на основе вычисления корреляционного момента. В общем случае корреляционный момент случайных величин ξ и η с возможными значениями x_i и y_i определяется по формуле

$$K_{\xi\eta} = \sum_i \sum_j (x_i - M[\xi])(y_i - M[\eta])P_{ij} \quad (17)$$

где P_{ij} – вероятность того, что (ξ, η) принимает значение (x_i, y_i), а $M[\xi]$, $M[\eta]$ – математические ожидания случайных величин.

Если случайные числа независимы, то $x_{\xi\eta} = 0$.

Независимость элементов последовательности $\{x_r\}$ может быть проверена путем введения в рассмотрение последовательности $\{y_r\}$ такой, что $\{y_r\} = \{x_{r+\tau}\}$, где τ – величина сдвига последовательностей.»[5]

Иногда вместо корреляционного момента удобней использовать коэффициент корреляции

$$p_\xi = \frac{K_{\xi n}}{Q_\xi Q_n}$$

где σ_ξ и σ_η – среднеквадратические отклонения величин ξ и η.

Возможные значения коэфициента корреляции лежат в пределах от 0 (полная независимость) до 1 (жесткая функциональная связь).

При любом $\tau \neq 0$ для достаточно больших N с доверительной вероятностью β справедливо соотношение

$$|p_\xi| \leq \beta\sqrt{\frac{1}{N}} \tag{18}$$

Если вычисленное экспериментальным путём ρ лежит в этих пределах, то с вероятностью β можно утверждать, что последовательность корреляционно независима.

При проведении оценок коэффициента корреляции на ЭВМ удобно для вычисления использовать следующее выражение:

$$p_{\xi\eta} = \frac{(\frac{1}{N-\tau}\sum_{i=1}^{N-\tau} x_{i+\tau} - \frac{1}{(N-\tau)N-\tau^2}\sum_{i=1}^{N-\tau} x_i \sum_{i=1}^{N-\tau} x_{i+\tau})}{\sqrt{D[x_i]D[x_{i+\tau}]}} \tag{19}$$

Где

$$D[x_i] = \left[\frac{1}{N-\tau}\right]\sum_{i=1}^{N-\tau}(x_i^2 - \frac{1}{(N-\tau)^2}(\sum_{i=1}^{N-\tau} x_i)^2) \tag{20},$$

$$D[x_{i+\tau}] = \frac{1}{N-\tau}\sum_{i+\tau}^{N-\tau}(x_{i+\tau}^2 - \frac{1}{(N-\tau)^2}(\sum_{i=1}^{N-\tau} x_{i+\tau})^2) \tag{21}.$$

2.4 Определение длины периода и отрезка апериодичности

«При статистическом моделировании с использованием программных генераторов псевдослучайных квазиравномерных последовательностей важными характеристиками качества генератора является длина периода P и длина отрезка апериодичности L. Длина отрезка апериодичности L псевдослучайной последовательности $\{x_r\}$, заданной уравнением

$$X_{i+1} = (AX_i + C) mod\ M, x_i = X_i/M, \quad (22)$$

есть наибольшее целое число, такое, что при $0 \leq k < i \leq L$ событие $P(x_i = x_j)$ не имеет места. Это означает, что все числа x_i в пределах отрезка апериодичности не повторяются.

Очевидно, что использование при моделировании систем последовательности чисел $\{x_r\}$, длина которой больше отрезка апериодичности L, может привести к повторению испытаний в тех же условиях, что и раньше, т. е. увеличение числа реализаций не дает новых статистических результатов.

Способ экспериментального определения длины периода P и длины отрезка апериодичности L сводится к следующему:

– Запускается программа генерации последовательности чисел $\{x_r\}$ с начальным значением x_0 на V значений, фиксируется x_y;

– Запуск программы генерации с x_0 и фиксируется i_1 и i_2, такие, что в первый и во второй раз выполняется условие $x_{i_1} = x_y$ и $x_{i_2} = x_y$. Вычисляется длина периода последовательности $P = i_1 - i_2$;

– Запускается программа генерации с начальными значениями x_0 и x_p и фиксируется минимальный номер i_3, для которого справедливо $x_{i_3} = x_{i_{3+p}}$. Вычисляется длина отрезка апериодичности $L = i_{3+p}$;

Теоретически при использовании мультипликативного метода длина периода не может быть больше чем 2^n, где n−разрядность ЭВМ. Для увеличения длины периода прибегают к специальным приемам.» [13]

2.5 Критерий серий

«Критерий серий позволяет убедиться в том, что пары последовательных чисел равномерно распределены независимым образом. Проверка критерия проводится по аналогии с предыдущим случаем, однако, считать будем

количество совпадений

$$(y_{2j}, y_{2j+1}) = (q, r), \quad 0 \le j \le n, 0 \le q, r \le d$$

Хи-квадрат критерий применяем к полученному набору с параметрами

$$k = d^2, p_j = \frac{1}{d^2}$$

Для достоверности результатов, полученных критерием хи-квадрат, величина проверяемой выборки должна быть значительно больше, чем количество возможных интервалов (значений). Учитывая это, рекомендуется выбирать длину выборки, по крайней мере,$n \ge 5d^2$ или уменьшать значение d. Критерий можно обобщить на тройки, четверки и т.д. случайных величин. Однако ширина проверяемого диапазона увеличивается пропорционально степени, равной количеству величин в кортеже. Поэтому на практике, при рассмотрении больших серий чисел, используются менее точные критерии.» [16]

2.6 Критерий хі-квадрат (x^2 -критерий)

«Критерий хи-квадрат широко применяется в статистических приложениях для проверки гипотезы о том, что некоторая выборка чисел $X^n = (x_1, x_2, \dots, x_n)$ подчиняется некоторому закону распределения $F(x)$. Критерий имеет другое название – критерий согласия Пирсона. Он является базовым для многих других критериев проверки статистических гипотез.

Тест на произвольные отклонения (The Random Excursions Test). Суть данного теста заключается в подсчете числа циклов, имеющих строго k посещений при произвольном обходе кумулятивной суммы. «Произвольный обход кумулятивной суммы начинается с частичных сумм после последовательности (0,1), переведенной в соответствующую последовательность от -1 до -2.» [15] «Цикл произвольного обхода состоит из серии шагов единичной длины, совершаемых в произвольном порядке.» [12] Кроме того, такой обход начинается и заканчивается на одном и том же элементе. Цель данного теста — определить отличается ли число посещений

определенного состояния внутри цикла от аналогичного числа в случае абсолютно случайной входной последовательности. Фактически данный тест есть набор, состоящий из восьми тестов, проводимых для каждого из восьми состояний цикла: −4, −3, −2, −1 и +1, +2, +3, +4.» [8]

«Тест приблизительной энтропии (The Approximate Entropy Test) Как и в тесте на периодичность в данном тесте акцент делается на подсчете частоты всех возможных перекрытий шаблонов длины m бит на протяжении исходной последовательности битов.»[10] «Цель теста — сравнить частоты перекрывания двух последовательных блоков исходной последовательности с длинами m и m+1 с частотами перекрывания аналогичных блоков в абсолютно случайной последовательности показывает число заранее заданных битовых строк (шаблонов) в последовательности. Поиск шаблонов осуществляется методом бегущего окна, с подсчетом найденных совпадений. В этом тесте, при найденном совпадении окно перескакивает на следующий за обнаруженным шаблоном бит.» [8]

«Тест на перекрывающиеся периодические шаблоны (Overlapping (Periodic) Template Matching Test). Цель теста – проверка линейной зависимости между подстроками фиксированного размера – матрицами 32x32 бита. Для этого осуществляется расчет рангов непересекающихся подматриц, построенных из исходной двоичной последовательности. Длина последовательности – не менее 38 912 бит, или 38 матриц» [4]

«Тест на произвольные отклонения (The Random Excursions Test) Суть данного теста заключается в подсчете числа циклов, имеющих строго k посещений при произвольном обходе кумулятивной суммы. Произвольный обход кумулятивной суммы начинается с частичных сумм после последовательности (0,1), переведенной в соответствующую последовательность (-1, +1). Цикл произвольного обхода состоит из серии шагов единичной длины, совершаемых в произвольном порядке. Кроме того, такой обход начинается и заканчивается на одном и том же элементе. Цель данного теста - определить отличается ли число посещений определенного

состояния внутри цикла от аналогичного числа в случае абсолютно случайной входной последовательности. Фактически данный тест есть набор, состоящий из восьми тестов, проводимых для каждого из восьми состояний цикла: −4, −3, −2, −1 и +1, +2, +3, +4.» [20]

«Тест на периодичность определяет, действительно ли количество появлений 2-ух перекрывающихся шаблонов длиной m бит приблизительно такое же, как в случае абсолютно случайной входной последовательности бит. Тест заключается в подсчете частоты всех возможных перекрываний шаблонов длины m бит на протяжении исходной последовательности битов. Предполагается, что в абсолютно случайной последовательности каждый шаблон длиной m бит появляется в последовательности с одинаковой вероятностью.» [23]

Вывод по главе

Во второй главе рассмотрены разные виды проверки генераторов псевдослучайных последовательностей.

Описаны разные критерии для оценки алгоритмов и для проверки равномерности распределения последовательностей случайных чисел, а также выведено описание качественного генератора.

3 Реализация и тестирование

3.1 Реализация мультипликативного конгруэнтного генератора

На языке C++ представлена реализация мультипликативного конгруэнтного генератора:

```
#include <stdio.h>
unsigned long int seed = 1;
unsigned int lcg() {
	int a = 16807;
	unsigned int m = 0x7fffffff;

	seed = seed * a % m;
	return seed;
}
int main(){

	unsigned int r = lcg();
	unsigned int n = 1;
	while(lcg() != 16807) {
		n++;
	}

	printf("%d\n" , n);
	return 0;

}
```

На выходе программа выводит глубину последовательности в виде числа.

3.2 Реализация квадратичного конгруэнтного генератора

Здесь представлена реализация квадратичного конгруэнтного генератора

```
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <math.h>

using namespace std;

int main(){
   int k;
   int a0 = 258;
   int a = 3, b = 22, c = 7, d = 5,count,a_next, a_prev;
   int m;
   cout<<"Input sequence length: ";
   cin >> k;
   m=32;
   a_prev = a0;
   for(count=1; count<k+1; count++) {
      a_next=a*pow(a_prev,3);
      a_next=a_next+b*pow(a_prev,2);
      a_next=a_next+c*a_prev;
      a_next=a_next+d;
      a_next =(a_next%m);
      a_prev = a_next;
      cout<<"new generated integer is "<<a_next<<endl;
   }
return 0;
```

3.3 Реализация линейного сдвигового регистра с обратной связью

Тут представлена конфигурация Галуа в виде кода для регистра сдвига длины 32 бит на языке Си:

```
int LFSR_Galois (void)
{
   // for polynomial 0x80000057, reversed 0xea000001
   static unsigned long S = 0x00000001;
   if (S & 0x00000001) {
      S = ((S ^ 0x80000057) >> 1) | 0x80000000;
      return 1;}
   else {
      S >>= 1;
      return 0;}
}
```

3.4 Реализация метода серединных квадратов

На рисунке 6 представлена 1 часть реализации метода серединных квадратов.

```
from datetime import datetime
def random_generator(initial_number=None):

   # только в случае, когда входное значение отсутствует, сами генерируем
первое случайное число
   if initial_number == None:
      initial_number = _get_initial_number()

   if not isinstance(initial_number, int):
```

Рисунок 6 – Первая часть реализации

Далее на 7 рисунке можно увидеть продолжение реализации:

```
raise ValueError("Входное значение не является числом!")

    while True:
        square_str = str(initial_number ** 2)
        start_index = len(square_str) // 4
        finish_index = start_index + 1 if len(square_str) % 2 else start_index

        initial_number = int(square_str[start_index:-finish_index])
        yield initial_number


def _get_initial_number():
    """
        Возвращает количество миллисекунд в текущей дате для построения
случайных чисел
    """
    now = datetime.now()
    return now.microsecond
|
# если происходит запуск модуля, то выведем первые 5 случайных чисел
if __name__ == "__main__":
    generator = random_generator(3485)
    for index, number in (zip(range(5), generator)):
        print("{0}: {1}".format(index, number))
```

Рисунок 7 – Вторая часть реализации

Заключение

В нынешнее время современная информатика не может обойтись от использования псевдослучайных последовательностей. Очень важно использовать качественные генераторы для получения достойных результатов.

Взглянув на результаты можно заметить, что многие широко используемые ГСЧ имеют серьезные недостатки, в том числе генераторы по умолчанию нескольких очень популярных программных продуктов. Поэтому перед проведением важных экспериментов всегда следует проверять, какой ГСЧ используется по умолчанию, и быть готовым заменить его, если это необходимо. Перед внедрением следует убедиться, что ГСЧ имеет надежную теоретическую поддержку, что он достаточно быстр и удобен.

Среди прочего. Никакой ГСЧ не может давать гарантий от всех возможных дефектов, но следует по крайней мере, избегать тех, которые с треском проваливают простые статистические тесты, и выбирать более надежный, например, в котором после многих лет использования и тестирования не было обнаружено никаких серьезных проблем.

В работе были изучены и реализованы алгоритмы генерации псевдослучайных последовательностей. Разработан блок статистических тестов, позволяющий проверить соответствие полученной последовательности равномерному распределению, а также ее случайность и независимость. Так же была изучена и проанализирована история первых появившихся алгоритмов и генераторов, построенных на них.

Хотелось бы подчеркнуть, что данная работа актуальна для решения нестандартных задач из разных сфер деятельности. В этих сферах используются последовательности разной длинны и для разных нужд. Для этого необходимо производить эти огромные наборы чисел с разными свойствами с помощью различных генераторов.

Список используемой литературы

1. А.В. Духанов О.Н. Медведева имитационное моделирование сложных систем. 2010. 118с. URL:http://simulation.su/uploads/files/default/2010-duhanov-medvedeva-1.pdf

2. В. С. Гончарук, Ю. С. Атаманов, С.Н. Гордеев. Методы генерации случайных чисел. //Молодой учёный. 2017. С. 20-23

3. В.А. Песошин, В.М. Кузнецов. Генераторы псевдослучайных чисел на регистрах сдвига. 2007. 297с.

4. Гришкин С.Г., Столов Е.Л. К проблеме идентификации начального состояния в генераторах псевдослучайных последовательностей. 1994. 17с. URL: https://new-disser.ru/_avtoreferats/01000212203.pdf

5. Дональд Э. Кнут. Искусство программирования. 2000. 800с. URL:https://bookree.org/reader?file=536014

6. Ивченко Г.И., Медведев Ю.И. Математическая статистика. 2010. 600с. URL: https://cs.petrsu.ru/~musen/appstat/readings/Ivchenko_Medvedev_-_Statistika.pdf

7. Левитан Ю.Л., Соболь И.М. О датчике псевдослучайных чисел для персональных компьютеров. 1990. 12с. URL:http://www.mathnet.ru/links/b27521bedb7b6e77461b2bce4ce63473/mm2431.pdf

8. М. А. Иванов, И. В. Чугунков. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. 2003. 122с. URL: https://bookree.org/reader?file=468181

9. М.Б. Будько, М.Ю. Будько, А.В. Гирик, В.А. Грозов. Методы генерации и тестирования случайных последовательностей. 2019. 73с.

10. Марченко М.А. Дистрибутивы программ и материалы по генератору псевдослучайных чисел и системе MONC. 2003. 12с.

11. Нечаев А.А. Мультирегистры сдвига и сложность мультипоследовательностей, 2003.

URL:https://www.bibliofond.ru/view.aspx?id=525770

12. Поляк Ю.Г, Филимонов В.А, Статистическое машинное моделирование средств связи. 1988. 175с. URL: https://www.studmed.ru/pollyak-yug-filimonov-va-statisticheskoe-mashinnoe-modelirovanie-sredstv-svyazi_dc0db8a163c.html

13. Слеповичев И.И. Генераторы псевдослучайных чисел. 2017. 118с.

14. Советов Б.Я., Яковлев С.А. Моделирование систем. 1985. 270с. URL:http://library.samdu.uz/files/717eb4889886817aeb071a743f030e0f_Моделирование%20систем..pdf

15. Страуструп Б. Язык программирования C++. 1999. 368с. URL: http://lib.ru/CPPHB/cpptut.txt

16. Шнайер Б. Прикладная криптография. 2003. 610с.

17. Ю.А. Кораблёв. Имитационное моделирование. 2017. 145с.

18. Artemiev S.S., Averina T.A. Numerical analysis of systems of ordinary and stochastic differential equations. 1997. 176с. URL: https://www.degruyter.com/document/doi/10.1515/9783110944662/html

19. Bratley P., Fox B. L., Niederreiter H. Implementation and tests of low-discrepancy sequences. 1992. 18с. URL: https://dl.acm.org/doi/pdf/10.1145/146382.146385

20. Intel Math Kernel Library. Reference Manual. 2003. 1461с. URL:https://www.bu.edu/tech/files/2010/02/mklman61.pdf

21. International Standard ISO/IEC DTR 19768 // Draft Tech. Rep. on C++ Library Extensions. 2007. 175с. URL: https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1836.pdf

22. Kirkpatrick S., Stoll E. A very fast shift-register sequence random number generator. 1981. 24с. URL: https://www.researchgate.net/publication/1872589_Four-tap_shift-register-sequence_random-number_generators

23. L’Ecuyer P. Good parameter sets for combined multiple recursive random number generators. 1999. 5с. URL:

https://www.researchgate.net/publication/261858547_Good_Parameters_and_Implementations_for_Combined_Multiple_Recursive_Random_Number_Generators

24. L’Ecuyer P. History of uniform random number generation. 2017. 29c.

25. L’Ecuyer P. Tables of linear congruential generators of different sizes and good lattice structure. 1999. 12c. URL: https://www.researchgate.net/publication/220577404_Tables_of_linear_congruential_generators_of_different_sizes_and_good_lattice_structure

26. Marsaglia G., Zaman A. A new class of random number generators. 1991.18c.URL:https://www.researchgate.net/publication/38363004_A_New_Class_of_Random_Number_Generators

27. Matsumoto M., Nishimura T. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. 1998. 28c. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.315.6296&rep=rep1&type=pdf

28. Mikhailov G.A., Marchenko M.A. Parallel realization of statistical simulation and random number generators. 2002. 180c.

29. Helmut G. Katzgraber. Random Numbers in Scientific Computing. 2010. 20c. URL: https://arxiv.org/pdf/1005.4117.pdf