Разработка генератора псевдослучайных чисел на основе логистического отображения

А. С. Шеханов, И. Н. Полякова

***Аннотация* — Разработка генераторов псевдослучайных чисел (ГПСЧ) является актуальной задачей на сегодняшний день, поскольку ГПСЧ применяются в широком спектре областей — от научных исследований, моделирования и статистического анализа до криптографии, финансовых систем и игровой индустрии. Качество работы таких генераторов напрямую влияет на надежность вычислений, безопасность информационных систем и достоверность результатов моделирования. Основной целью данной работы является разработка генератора псевдослучайных чисел, обеспечивающего высокую степень случайности генерируемых последовательностей. В рамках исследования предлагается метод построения ГПСЧ на основе логистического отображения, представляющего собой нелинейную динамическую систему с хаотическим поведением. Для оценки качества получаемых последовательностей разработан метод тестирования на основе энтропийного анализа с использованием скользящего окна, позволяющий оценивать степень случайности битовой последовательности. Проведено сравнение предложенного генератора с линейным конгруэнтным генератором и вихрем Мерсенна. Тестирование выполняется с использованием энтропийного теста и анализа гистограмм распределения. Полученные результаты визуализируются в виде графиков и диаграмм, что позволяет провести сравнительный анализ характеристик исследуемых генераторов.**

***Ключевые слова* — генераторы псевдослучайных чисел, логистическое отображение, энтропия, динамический хаос.**

I. Введение

В наше время компьютерные системы используются в различных сферах жизни, от научных исследований до финансов и бизнеса. Для решения многих задач в компьютерных системах требуется генерация данных, таких как номера транзакций, пароли и других, которые должны быть случайными и не поддающимися предсказанию. Для генерации таких данных используются генераторы псевдослучайных чисел.[1]

Генераторы псевдослучайных чисел (ГПСЧ) – это алгоритмы, которые производят последовательность чисел, которые выглядят случайными, но на самом деле являются детерминированными и могут быть воспроизведены. В отличие от истинно случайных чисел, которые могут быть получены, например, при помощи физических процессов, таких как радиоактивный распад, ГПСЧ используют математические формулы для создания последовательности чисел.

Генерация псевдослучайных чисел является важной задачей в области криптографии, статистики, компьютерных симуляций, игр и других областях, где требуется случайность, и для каждой области существуют свои критерии оценки того, насколько генератор подходит под конкретную ситуацию. Например, в криптографии ГПСЧ используются для создания криптографических ключей, которые должны быть безопасными и не поддающимися взлому. Если криптографический ключ, основанный на псевдослучайных числах, становится известен злоумышленнику, то это может привести к нарушению безопасности системы. Поэтому важно обеспечивать криптографическую стойкость генераторов псевдослучайных чисел, если они используются в области информационной безопасности. Если же ГПСЧ используется для моделирования случайных событий в науке, то криптографическая стойкость не так важна, а важна достаточная случайность генерируемых чисел [2].

Существует множество различных алгоритмов генерации псевдослучайных чисел, каждый из которых имеет свои сильные и слабые стороны.

Несмотря на то, что генераторы псевдослучайных чисел являются неотъемлемой частью многих компьютерных систем и программ, их качество и безопасность не всегда получают достаточное внимание. Недостаточно качественные генераторы могут привести к серьезным проблемам, таким как возможность подбора ключей в криптографии, или неэффективность при моделировании случайных событий в науке и инженерии [3].

Поэтому важно не только уметь выбирать наиболее подходящий ГПСЧ для конкретной задачи, но и тщательно проверять его качество и безопасность при помощи соответствующих статистических тестов и анализа.

II. Существующие рассматриваемые генераторы псевдослучайных чисел

Линейный конгруэнтный генератор (ЛКГ). Простой алгоритм генерации псевдослучайных чисел, основанный на линейной рекуррентной формуле. Он получает на вход начальное значение и генерирует последовательность псевдослучайных чисел, определяемых по формуле:

Статья получена 27 мая 2026.
А.С. Шеханов, магистрант, Московский государственный университет имени М.В. Ломоносова (email: alexeysh2002@yandex.ru)

И. Н. Полякова, к. ф.-м. н., доцент, Московский государственный университет имени М.В. Ломоносова (email: polyakova@cs.msu.ru)

$$x_{n+1} = (a * x_n + c)\ mod\ m,$$

- x_n – текущее значение;
- x_{n+1} – следующее значение в последовательности;
- $m > 0$ – модуль, период генератора будет меньше либо равен, чем это значение;
- $0 \le x_0 \le m$ – начальное значение;
- $0 \le a < m$ – множитель;
- $0 \le c < m$ – приращение (инкремент);

Основное преимущество линейного конгруэнтного генератора – это его простота и быстрота. Однако, также существуют и недостатки. Например, период этого генератора ограничен и не может превышать m. Кроме того, если выбрать плохие значения для параметров a, c и m, то можно получить последовательность с низким качеством генерации псевдослучайных чисел [4].

Для обеспечения хорошего качества генерации псевдослучайных чисел и максимального периода последовательности важно тщательно подбирать значения параметров a, c и m. Например, рекомендации изложил Дональда Кнута [5].

Вихрь Мерсенна. Является генератором псевдослучайных чисел, встроенным во многие языки программирования: Python, R, C++, PHP, Ruby и т.д. Существуют по меньшей мере два общих варианта алгоритма, различающихся только величиной используемого простого числа Мерсенна, то есть числа вида 2^p – 1, где p – простое число, наиболее распространённым из которых является алгоритм MT19937, период которого составляет 2^{19937} – 1 [6]. Во встроенном модуле языка Python, который называется random, используется данный вариант алгоритма [7].

III. Существующие методы тестирования генераторов псевдослучайных чисел

Для того, чтобы убедится, что выбранный генератор псевдослучайных чисел подходит для конкретной задачи нужно проводить тестирование. Тестирование ГПСЧ важно для обеспечения надежности и качества случайных данных в различных приложениях и системах. Это позволяет убедиться в том, что генерируемые числа обладают нужной степенью случайности, равномерности распределения и непредсказуемости. Тестирование ГПСЧ помогает обеспечить безопасность в криптографических системах, точность в научных и инженерных расчетах, надежность в программном обеспечении и честность в играх и развлекательных приложениях. Кроме того, тестирование ГПСЧ важно для проверки соответствия генераторов стандартам и требованиям качества в соответствующих областях применения [8].

Тесты для генераторов псевдослучайных чисел можно разделить на две большие группы — это статические и графические тесты.

Графические тесты представляют собой метод оценки качества генерируемой последовательности случайных чисел путем визуализации. Эти тесты могут помочь выявить недостатки в равномерности распределения, периодичности и других характеристиках случайной последовательности. Например, можно построить график точек и оценить наличие корреляций между числами. Можно также построить гистограмму и оценить, насколько равномерно распределены числа в выбранном диапазоне. Графические тесты являются важным инструментом при разработке и оценке генераторов псевдослучайных чисел, поскольку они позволяют не только количественно оценить статистические характеристики, но и визуально идентифицировать аномалии или неравномерности в генерируемой последовательности [9].

Статистические тесты, в свою очередь, основаны на математическом анализе последовательности псевдослучайных чисел. В отличие от графических тестов, статистические тесты дают количественную оценку последовательности и позволяют определить, был ли тест пройден. Они также могут использовать различные статистические критерии для проверки гипотез о распределении чисел в последовательности [10].

IV. Предлагаемый алгоритм построения генератора псевдослучайных чисел. Логистическое отображение

Логистическое отображение, также известное как квадратичное отображение, представляет собой пример модели динамического хаоса, которая демонстрирует сложное поведение, проистекающее из простого нелинейного уравнения. Это дискретный аналог непрерывного логистического уравнения Ферхюльста, которое описывает изменение популяции в дискретные моменты времени. Математическая формула для этого отображения выглядит следующим образом [11]:

$$x_{n+1} = 4 * r * x_n * (1 - x_n),$$

- $x_n \in (0, 1)$ – текущее значение последовательности;
- x_0 – начальное значение, также называемое seed;
- r – положительный параметр.

Поведение уравнения при различных значениях параметра r показано на рис. 1.

В работе проведено исследование, сравнение и тестирование уравнения при параметре $r = 1$, так как только при этом значении, мощность множества значений наибольшая, а так же, как видно по диаграмме, при этом значении нет “окна”, а x_n не стремится к конкретному значению. Поведение уравнения при других значениях параметра r также было исследовано, но в результате экспериментов при разных значениях параметра и разных начальных значениях было получено, что наилучшие результаты по статистическим тестам достигаются при r = 1.

V. Предлагаемый алгоритм тестирования на основе энтропии

Данный алгоритм тестирования можно применять к любому генератору псевдослучайных чисел. Единственное, что нужно сделать, это перевести сгенерированную последовательность чисел в последовательность бит, то есть нулей и единиц. Так как все рассматриваемые ГПСЧ создают последовательность чисел из интервала $(0; 1)$, то предлагается делать это довольно простым способом: если число больше $\frac{1}{2}$, то переводим его в 1, а если меньше $\frac{1}{2}$, то в 0.

После данного преобразования получаем последовательность нулей и единиц, с которой далее

будем производить все вычисления.

Пусть последовательность чисел имеет длину n.

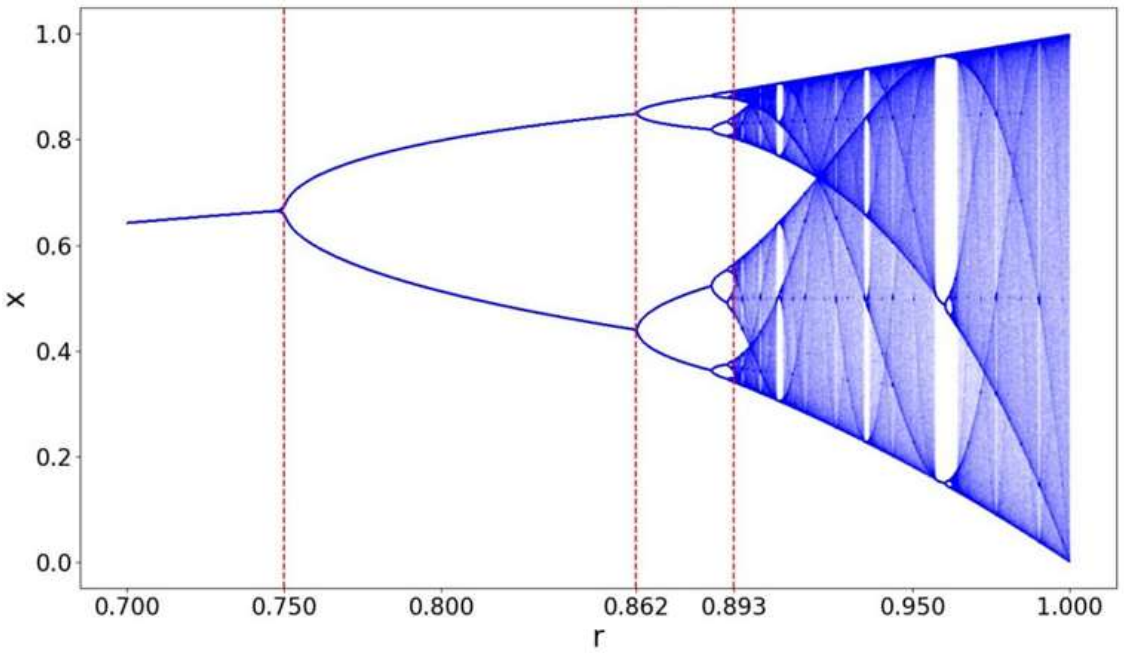

Рис. 1 - Бифуркационная диаграмма логистического отображения.

Определяем ширину скользящего окна, далее постепенно увеличиваем размер скользящего окна на 1. Зафиксируем его размер, который будет равен k. Далее двигаем окно от начала до конца последовательности с шагом в один символ. Внутри окна каждый раз будет наблюдаться новое число в двоичной системе длины k. Так как размер окна k, то количество комбинаций, которое может встретиться при движении окна, равно 2^k. Последовательность бит внутри окна рассматриваем как двоичное число. Все разнообразие таких чисел, очевидно, исчерпывается сегментом $[0, 2^k - 1]$. Введем счетчик p_i, где $i \in [0, 2^k - 1]$, и посчитаем, сколько раз встретилось каждое число. Количество раз, когда встретилось i равно числу p_i. Далее введем c_i, как $c_i = \frac{p_i}{n-k+1}$, где знаменатель дроби — это количество положений окна размера k, при длине последовательности n. То есть будем интерпретировать c_i как вероятность встретить число i при движении окна. Основываясь на формуле энтропии для дискретного распределения, считаем энтропию для каждого из размеров скользящего окна:

$$H(k) = -\sum_{i=0}^{m-1} c_i \log_m c_i, \qquad (1)$$

- k – текущий размер окна.
- m – количество комбинаций 0 и 1 при текущем размере окна, то есть 2^k.

Границы размера скользящего окна будем определять следующим образом: левая равна 1, а правая зависит от длины тестируемой последовательности. Обозначим правую границу за k_{right}, тогда количество положений окна будет равно $n - k_{right} + 1$, а количество комбинаций, которое может встретиться при движении окна это $2^{k_{right}}$. При большой длине последовательности, $n \gg k_{right} - 1$, значит можно пренебречь слагаемым $(-k_{right} + 1)$ и считать, что количество положений окна приблизительно равно n.

Тогда, если окажется, что количество положений окна будет меньше, чем общее количество комбинаций, то физически невозможно будет наблюдать все комбинации. Максимальное количество комбинаций, которое можно увидеть, при условии, что комбинации встретятся по одному разу, будет равняться $2^{k_{right}}$. То есть функция $H(k)$ не сможет держать единицу при таком условии. А если еще больше увеличивать размер окна, то значение функции энтропии будет стремиться к нулю. Значит должно выполняться соотношение:

$$n \geq 2^{k_{right}};$$

$$\log_2 n \geq k_{right};$$

$$k_{right} = \lfloor \log_2 n \rfloor. \qquad (2)$$

Правая граница k_{right} будет являться максимальным размером скользящего окна, при котором функция энтропии должна “держать” единицу. Для наших тестов будем брать правую границу чуть больше, чем получилось k_{right}, чтобы посмотреть, насколько быстро функция энтропии стремиться к нулю.

Значение энтропии лежит в полуинтервале $(0; 1]$, так как было выбрано основание логарифма, равному количеству комбинаций. Максимум энтропии, в нашем случае 1, будет достигаться для равномерного распределения. Очевидно, для достижения энтропией 1, нужно чтобы количество положений скользящего окна нацело делилось на количество комбинаций нулей и единиц, которые могут встретиться при его движении, при этом частота появления каждой комбинации должна быть одинаковой. То есть чем ближе к единице получилась энтропия, тем труднее статистически отличить последовательность, генерируемых тестируемым ГПСЧ от равномерно распределенной

последовательности. Соответственно, минимум энтропии, в нашем случае 0, будет достигаться, если все числа последовательности одинаковые, либо нули, либо единицы [12].

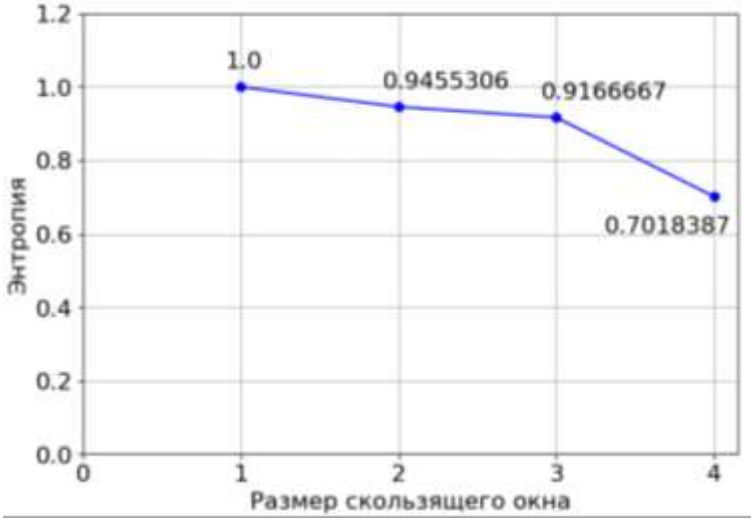

Рис. 2 – Модельный пример. Зависимость энтропии от размера скользящего окна

Модельный пример. Рассмотрим модельный пример. Пусть у нас есть битовая последовательность: «0100101110». Так как длина последовательности 10 символов, то правую границу считаем по формуле (1):

$$k_{right} = \lfloor \log_2 10 \rfloor;$$

$$k_{right} = \lfloor 3{,}3219 \rfloor;$$

$$k_{right} = 3.$$

Возьмем правую границу скользящего окна чуть больше, чтобы убедиться, что, когда правая граница станет больше 3, энтропия снизится.

Так как у нас одинаковое число нулей и единиц, то при размере скользящего окна равного 1, получим энтропию равную 1. При размере окна равного 2, у нас есть 4 варианта комбинаций нулей и единиц. При прохождении скользящим окном по последовательности наблюдаются следующие комбинации: «01», «10», «00», «01», «10», «01», «11», «11», «10».

Тогда получим количество различных комбинаций:

- «00» - 1;
- «01» - 3;
- «10» - 3;
- «11» - 2.

Теперь можно считать энтропию по формуле (1):

$$-\frac{1}{9}\log_4\frac{1}{9} - \frac{3}{9}\log_4\frac{3}{9} - \frac{3}{9}\log_4\frac{3}{9} - \frac{2}{9}\log_4\frac{2}{9} \approx 0{,}9455306\,.$$

Аналогично считаем энтропию для всех значений скользящего окна. Результаты показаны на рис. 2.

Таким образом будем считать энтропию и строить графики для всех предложенных в этой работе, а также для некоторых существующих генераторов псевдослучайных чисел для сравнения.

Если генератор псевдослучайных чисел показал себя хорошо на данном тесте, нельзя с полной уверенностью сказать, что он будет хорошо себя показывать в практических задачах, однако если результаты данного теста далеки от идеальных, то такой ГПСЧ не стоит использовать на практике.

VI. Результаты тестирования

Будем проводить тестирование генераторов псевдослучайных чисел с помощью алгоритма тестирования на основе энтропии и гистограммы распределения элементов с округлением до 2 знаков после запятой. Для каждого теста генерируем 1’000’000 чисел.

При размере скользящего окна большем, чем 19, энтропия будет сильно уменьшаться, потому что количество положений скользящего окна размера 20 на последовательности длиной 1’000’000 – это 1’000’000 - 20 + 1 = 999981 < 2^{20} = 1’048’576 - количество различных двоичных чисел длиной 20. Если увеличивать размер скользящего окна, то, количество комбинаций будет увеличиваться, а количество его положений уменьшаться, а значит, и энтропия будет уменьшаться.

Все функции являются дискретными, но они будут изображаться непрерывными для наглядности.

Несмотря на то, что гистограмма распределения у ГПСЧ на основе логистического отображения сильно отличается от равномерного распределения (рис. 3), все 3 рассматриваемых генератора псевдослучайных чисел показали примерно одинаковые результаты на тесте на основе энтропии (рис. 4, рис. 5, рис. 6), трудно выделить какой-то конкретный, при разных значениях скользящего окна, наибольшую энтропию показывают разные ГПСЧ. Энтропия начинает сильно отклоняться от 1, как и предполагалось, при значении размера плавающего окна равному 20.

VII. Заключение

Работа посвящена интересной и востребованной тематике генераторов псевдослучайных чисел (ГПСЧ). Рассмотрены как некоторые существующие генераторы, так и генераторы, основанные на предложенных авторами подходах к созданию ГПСЧ.

Проведенное масштабное (исследованы выборки размером в 1 000 000) экспериментальное исследование ГПСЧ позволило сформулировать рекомендации по их

применению. Достаточно интересным оказался следующий результат: — генератор на основе

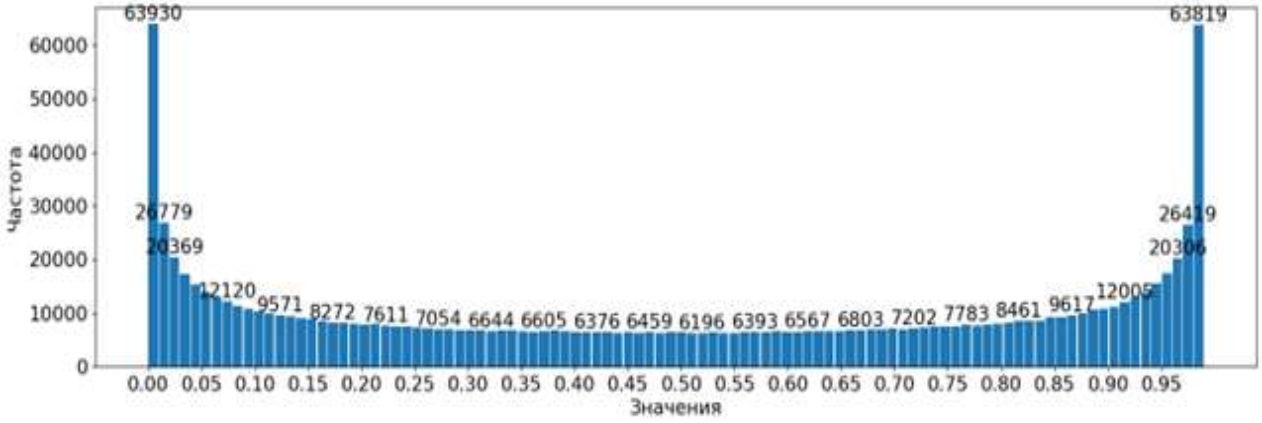

Рис. 3 - Логистическое отображение с параметрами r = 1, x_0= 0.32326721016063265. Гистограмма распределения.

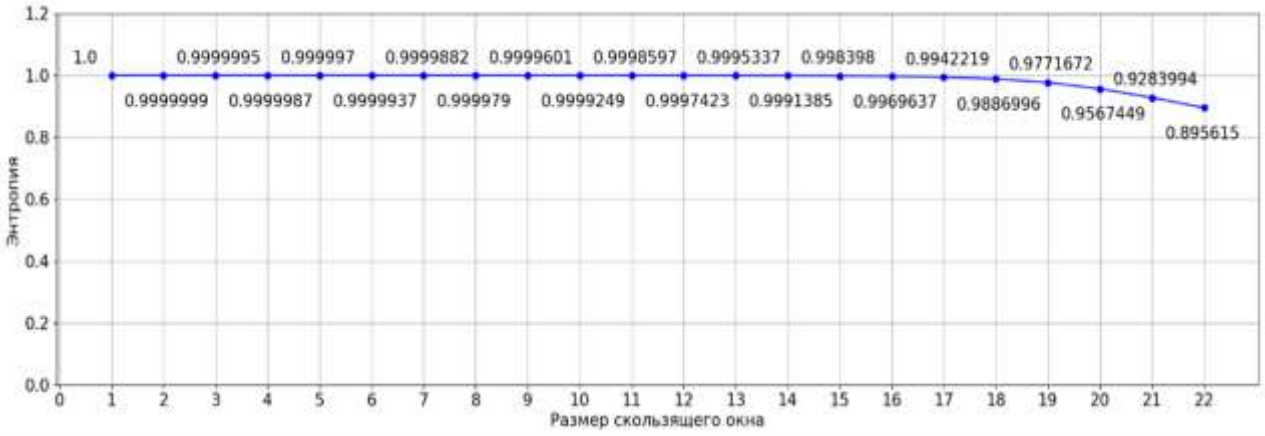

Рис. 4 - Логистическое отображение с параметрами r = 1, x_0 = 0.32326721016063265. Зависимость энтропии от размера скользящего окна.

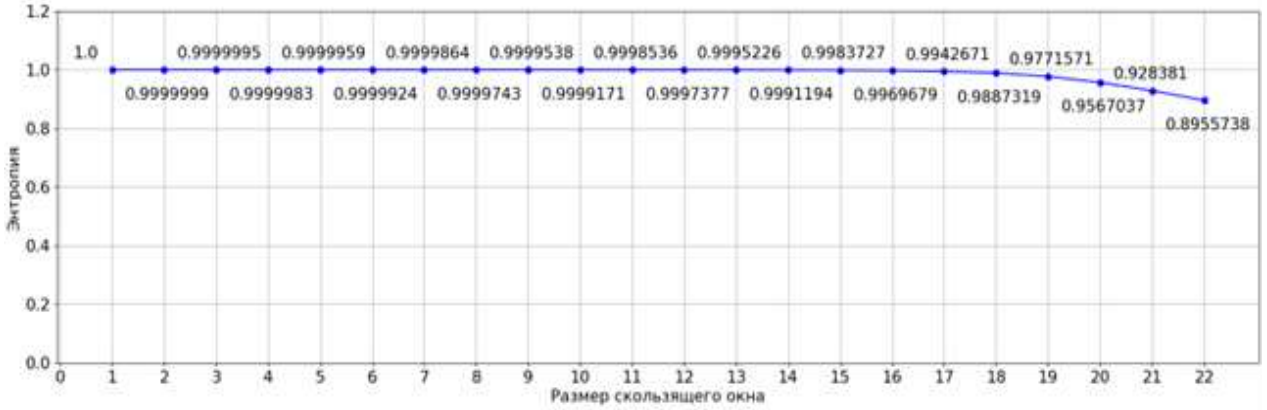

Рис. 5 - Вихрь Мерсенна. Зависимость энтропии от размера скользящего окна.

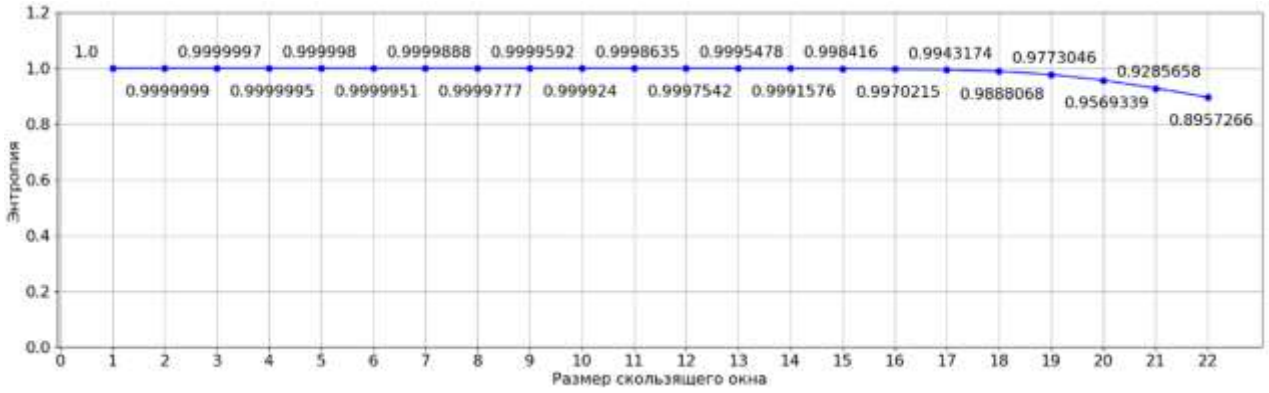

Рис. 6 - Линейный конгруэнтный генератор. Зависимость энтропии от размера скользящего окна.

логистического отображения показывает хорошие результаты как генератор псевдослучайных бит, но не может быть использован как генератор псевдослучайных на (0, 1) чисел из-за симметричной, но не равномерной гистограммы. Исследование причин такого поведения вполне может стать темой дальнейших научных исследований.

Благодарности

Работа выполнена в рамках НИР, проводимой на кафедре алгоритмических языков факультета вычислительной математики и кибернетики Московского государственного университета имени М.В.Ломоносова.

Авторы выражают благодарность профессору Ульянову Михаилу Васильевичу за постановку задачи и помощь в получении результатов.

Библиография

[1] N.G. Bardis, A.P. Markovskyi, N. Doukas, N. V. Karadimas. True Random Number Generation Based on Environmental Noise Measurements for Military Applications // Proceedings of the 8th WSEAS International Conference on SIGNAL PROCESSING, ROBOTICS and AUTOMATION. — 2009. — С. 68—73.

[2] Pierre L'Ecuyer Random Number Generation // Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. - 2007. - p. 93-137.

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery NUMERICAL RECIPES The Art of Scientific Computing. - 3 ed. - Cambridge University Press, 2007. - 1262 p.

[4] О.Р. Лапонина Основы сетевой безопасности. Часть 2. Технологии туннелирования. - М.: Национальный Открытый Университет «ИНТУИТ», 2014. - 474 с.

[5] Дональд Кнут. Получисленные методы // Искусство программирования. Т. 2, Изд. 3 — Москва: Издательский дом "Вильямс", 2018. — 832 С., С. 21–37.

[6] MATSUMOTO M., NISHIMURA T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator // ACM Transactions on Modeling and Computer Simulation - 1998. № 1. - Vol. 8. - P. 3-30.

[7] random — Generate pseudo-random numbers [Электронный ресурс]. — URL: https://docs.python.org/3/library/random.html (Дата обращения: 25.03.26).

[8] Григорьев А. Ю. Методы тестирования генераторов случайных и псевдослучайных последовательностей // Ученые записки УлГУ. Сер. Математика и информационные технологии. УлГУ. Электрон. журн. - 2017. - №1. - С. 22-28.

[9] Gould H., Tobochnik J., Wolfgang C. An Introduction to Computer Simulation Methods: Applications to Physical Systems — 1988. 695 P. — P. 142–182.

[10] Сметанин Ю.Г., Ульянов М.В., Пестова А.С. Энтропийный подход к построению меры символьного разнообразия слов и его применение к кластеризации геномов растений // Математическая биология и биоинформатика — 2016, № 1 — Т. 11 — С. 114-126.

[11] Будько М.Б., Будько М.Ю., Гирик А.В., Грозов В.А. Методы генерации и тестирования случайных последовательностей – СПб: Университет ИТМО, 2019. – 70 с.

[12] Пикуза М.О., Михневич С.Ю. Тестирование аппаратного генератора случайных чисел при помощи набора статистических тестов NIST. Доклады БГУИР. 2021; 19(4): 37-42

Development of a Pseudo-Random Number Generator Using the Logistic Map

Alexey Shekhanov, Irina Polyakova

***Abstract* — The development of pseudo-random number generators (PRNGs) is a relevant problem today, as PRNGs are widely used in various fields, ranging from scientific research, simulation, and statistical analysis to cryptography, financial systems, and the gaming industry. The quality of such generators directly affects the reliability of computations, the security of information systems, and the validity of simulation results. The main objective of this work is to develop a pseudo-random number generator that provides a high degree of randomness in the generated sequences. This study proposes a method for constructing a PRNG based on the logistic map, which is a nonlinear dynamical system exhibiting chaotic behavior. To evaluate the quality of the generated sequences, an entropy-based testing method using a sliding window is developed, allowing the assessment of randomness in binary sequences. The proposed generator is compared with a linear congruential generator and the Mersenne Twister. Testing is performed using entropy-based analysis and histogram distribution evaluation. The obtained results are visualized using graphs and diagrams, enabling a comparative analysis of the characteristics of the studied generators.**

***Keywords* — Pseudo-random number generators, logistic map, entropy, dynamical chaos.**

REFERENCES

[1] N.G. Bardis, A.P. Markovskyi, N. Doukas, N. V. Karadimas. True Random Number Generation Based on Environmental Noise Measurements for Military Applications // Proceedings of the 8th WSEAS International Conference on SIGNAL PROCESSING, ROBOTICS and AUTOMATION. — 2009. — C. 68—73.

[2] Pierre L'Ecuyer Random Number Generation // Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. - 2007. - p. 93-137.

[3] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery NUMERICAL RECIPES The Art of Scientific Computing. - 3 ed. - Cambridge University Press, 2007. - 1262 p.

[4] O.R. Laponina, *Osnovy setevoi bezopasnosti. Chast' 2. Tekhnologii tunnelirovaniya*. Moscow: Natsional'nyi Otkrytyi Universitet "INTUIT", 2014, 474 p.

[5] D. Knuth, *Poluchislennye metody, Iskusstvo programmirovaniya*, vol. 2, 3rd ed. Moscow: Izdatel'skii dom "Vil'yams", 2018, pp. 21–37.

[6] MATSUMOTO M., NISHIMURA T. Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator // ACM Transactions on Modeling and Computer Simulation - 1998. № 1. - Vol. 8. - P. 3-30.

[7] *random — Generate pseudo-random numbers* [Online]. Available: https://docs.python.org/3/library/random.html (accessed March 25, 2026)

[8] A.Yu. Grigor'ev, "Metody testirovaniya generatorov sluchainykh i psevdosluchainykh posledovatel'nostei," Uchenye zapiski UlGU. Ser. Matematika i informatsionnye tekhnologii, no. 1, pp. 22–28, 2017.

[9] Gould H., Tobochnik J., Wolfgang C. An Introduction to Computer Simulation Methods: Applications to Physical Systems — 1988. 695 P. — P. 142–182.

[10] Yu.G. Smetanin, M.V. Ul'yanov, and A.S. Pestova, "Entropiinyi podkhod k postroeniyu mery simvol'nogo raznoobraziya slov i ego primenenie k klasterizatsii genomov rastenii," Matematicheskaya biologiya i bioinformatika, vol. 11, no. 1, pp. 114–126, 2016.

[11] M.B. Bud'ko, M.Yu. Bud'ko, A.V. Girik, and V.A. Grozov, Metody generatsii i testirovaniya sluchainykh posledovatel'nostei. Saint Petersburg: Universitet ITMO, 2019, 70 p.

[12] Pikuza M.O., Mikhnevich S.Yu. Testing a hardware random number generator using NIST statistical test suite. Doklady BGUIR. 2021; 19(4): 37-42.