ЖУРНАЛ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ, 2020, том 60, № 11, с. 1807–1814

ОБЩИЕ ЧИСЛЕННЫЕ МЕТОДЫ

УДК 519.65

НЕНАДЕЖНОСТЬ ИЗВЕСТНЫХ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

© 2020 г. А. А. Белов[1,2,*], Н. Н. Калиткин[3,**], М. А. Тинтул[1]

[1] 119991 Москва, Ленинские горы, МГУ, физ. фак-т, Россия

[2] 115419 Москва, ул. Орджоникидзе, 3, Российский университет дружбы народов, Факультет физико-математических и естественных наук, Россия

[3] 125047 Москва, Миусская пл., 4, ИПМ, Россия

*e-mail: aa.belov@physics.msu.ru

**e-mail: kalitkin@imamod.ru

Поступила в редакцию 07.02.2020 г.
Переработанный вариант 07.02.2020 г.
Принята к публикации 07.07.2020 г.

Рассмотрена проблема построения последовательностей равномерно распределенных псевдослучайных чисел. Использован простой визуальный критерий для оценки случайности чисел последовательности. Этот тест показал, что наиболее распространенные современные генераторы случайных чисел, основанные на методе вихря Мерсена, линейной конгруэнтной последовательности и ряде других принципов, дают неудовлетворительные результаты. Поэтому проблема построения хороших генераторов остается нерешенной, а к результатам расчета случайных процессов (метод молекулярной динамики и др.) следует относиться с осторожностью. Библ. 13. Фиг. 15.

Ключевые слова: методы Монте-Карло, псевдослучайные числа, тестирование.

DOI: 10.31857/S0044466920110046

1. ПРОБЛЕМА

Методы Монте-Карло широко используются в различных областях прикладной математики: вычисление многомерных интегралов, разыгрывание столкновений в методах молекулярной динамики и задачах реакторной защиты, оценка отказов в сложных конструкциях и многие другие. Все эти методы используют некоторые последовательности случайных чисел с заданной функцией распределения. Чаще всего такие задачи сводятся к использованию последовательности случайных чисел γ_n, равномерно распределенных на отрезке [0, 1]. Такие последовательности имитируют наборами псевдослучайных чисел, получаемых с помощью некоторых математических алгоритмов. Все эти алгоритмы являются по сути эвристическими.

Любой предлагаемый алгоритм обычно тестируют – проверяют, насколько выполняются те или иные свойства, характерные для чисто случайных последовательностей. Например, близко ли выборочное среднее от γ_n к 1/2; существенна ли корреляции между γ_n и γ_{n+1}, равномерность распределения точек в кубах разной размерности и т.д. В [1] тестировалось 13 популярных генераторов псевдослучайных чисел. При этом использовалась стандартная библиотека diehard-тестов, включавшая проверку равномерности заполнения единичного куба с числом измерений до 15. Автор работы констатирует, что последовательности удовлетворяют тестам, однако, он считает, что это не гарантирует хорошего качества последовательностей. Поэтому систему формальных математических тестов нельзя считать достаточно надежной.

Математики дают путевку в жизнь только тем генераторам, которые проходят подобные тесты. Затем физики, которые не всегда хорошо знают математику, но очень часто ее обожествляют, доверчиво используют эти последовательности в своих расчетах. Рассмотрим критически эту проблему. Любой математический алгоритм является детерминированной процедурой, поэтому любая псевдослучайная последовательность не может быть истинно случайной и должна содержать какие-то закономерности. Вопрос лишь в одном – насколько они существенны. Мы про-

тестировали ряд широко употребительных алгоритмов и убедились в том, что их вряд ли можно считать надежными. Данная работа посвящена иллюстрации этих выводов.

2. ПРОСТЕЙШИЙ ВИЗУАЛЬНЫЙ ТЕСТ

Тесты псевдослучайных последовательностей можно условно разбить на две группы: статистические и визуальные. В статистических тестах проверяются важнейшие соотношения между числами последовательности. Например, можно вычислять выборочные моменты и корреляции исследуемой последовательности и сравнивать их с теоретически ожидаемыми для данного распределения. Однако для тщательного исследования требуется вычислять моменты и корреляции высоких порядков. Это слишком трудоемко. Обычно ограничиваются несколькими первыми моментами и парными корреляциями, что не обеспечивает надежность тестирования.

Визуальные тесты не являются строгими, зато они очень просты и наглядны. Например, очень популярен следующий тест. В единичном кубе строят псевдослучайные точки. Затем используют программу проекции куба на экран компьютера и вращают куб в пространстве. Как правило, при некоторых углах обзора удается наблюдать на экране неравномерность распределения точек, причем значительную. Обычно точки группируются вблизи некоторых параллельных плоскостей в трехмерном пространстве, а в промежутках между этими плоскостями плотность распределения точек существенно меньше. Эта процедура требует хорошей квалификации тестирующего, иначе можно пропустить соответствующий угол поворота.

Заметим, что в этом сложном способе мы наблюдаем не всю трехмерную картину, а лишь ее проекцию на плоскость. Поэтому гораздо проще с самого начала ограничиться плоским случаем – исследованием равномерности распределения точек в единичном квадрате. Получаемую картину легко оценить визуально. При этом отметим одну существенную техническую деталь. Размер маркеров на экране или при выводе на печать надо выбирать так, чтобы в местах максимального сгущения точек они почти перекрывались бы. Для этого суммарная площадь всех маркеров должна составлять 15–25% от площади единичного квадрата (таким образом, размер маркера зависит от числа выведенных точек).

Нередко визуальный контроль объявляют субъективным. Однако известно, что глаз зачастую лучше выявляет закономерности, чем формальные математические методы. Например, глаз легко оценивает выход последовательности точек на асимптоту. Формальные математические критерии требуют гораздо большего числа точек для надежного установления такого выхода. Это хорошо известно тем, кто разрабатывает методики автоматического контроля точности расчетов.

Рассмотренные генераторы. Мы исследовали 13 генераторов псевдослучайных чисел, широко распространенных в отечественной и зарубежной литературе, а также в коммерческих пакетах:

- вихрь Мерсенна (Mersenne twister) [2];
- быстрый вихрь Мерсенна (SIMD-oriented fast Mersenne twister) [3];
- мультипликативный конгруэнтный генератор (Multiplicative congruential generator) [4];
- мультипликативный генератор Фибоначчи с запаздыванием (64-bit multiplicative lagged Fibonacci generator) [5],
- комбинированный множественный рекурсивный генератор (combined multiple recursive generator) [6];
- генератор Philo 4×32 [7];
- генератор Threefry 4×64 [7];
- генератор Марсалья (Marsaglia's SHR3 shift-register generator) [8];
- модифицированный генератор Subtract-with-Borrow (modified Subtract-with-Borrow generator) [9];
- rand на языке C/C++;
- модифицированная последовательность Лемера [10];
- последовательность чисел из сборника [11];
- последовательности Соболя [12], [13].

Этот список частично совпадает с последовательностями, тестированными в [1]. Дадим краткие характеристики этих алгоритмов.

Вихрь Мерсенна – это весьма сложный алгоритм. В нем генерируются две различные независимые последовательности 32-битовых целых чисел, из которых составляется одно 64-битовое число. Поэтому окончательный результат соответствует точности double precision. Такая комби-

нация двух последовательностей обеспечивает период 2^{19937}-1, далеко превосходящий все ожидания современных вычислителей. Ясно, что при такой длине невозможно использовать целую замкнутую цепочку чисел. Этот алгоритм считается одним из наиболее совершенных.

Быстрый вихрь Мерсенна считается более быстрой реализацией предыдущего алгоритма. Однако проверить, приводят ли оба алгоритма к строго одинаковым результатам, непросто.

Мультипликативный конгруэнтный генератор – это алгоритм Лемера с некоторыми конкретными константами:

$$m_{n+1} = (Am_n + C) \bmod M, \quad A = 7^5, \quad C = 0, \quad M = 2^{32} - 1. \tag{1}$$

Он использует одну 32-битовую последовательность целых чисел, и его период не может превышать M. Это несравненно меньше, чем для вихря Мерсенна.

64-битовый мультипликативный генератор Фибоначчи с запаздыванием использует одну 64-битовую последовательность целых чисел. Его период составляет приблизительно 2^{124}. Это много больше, чем нужно для современных приложений. Константы запаздывания равны $l = 63$, $k = 31$.

Комбинированный множественный рекурсивный генератор использует две 32-битовые последовательности целых чисел. Рекурсивная процедура обеспечивает длину периода 2^{191}.

Philox 4 × 32 – генератор с использованием четырех независимых последовательностей 32-битовых чисел. По-видимому, этот генератор ориентирован на задачи криптографии. В каждой последовательности нахождение m_{n+1} через m_n осуществляется с помощью сети Фейстеля с некоторыми наборами ключей. Период составной последовательности равен 2^{193}.

Threefry 4 × 64 – это адаптация блочного криптографического алгоритма Threefish из хэш-функции Skein. Она использует четыре 64-битовых последовательности и обеспечивает период 2^{514}.

Генератор Марсалья использует линейный конгруэнтный алгоритм (1) с параметрами $A = 69069$, $M = 2^{32}$, $C = 1234567$. Вычисления производятся с 32-битовыми числами. Каждое полученное по формуле (1) число подвергается следующей процедуре. Сначала оно сдвигается влево на 5 регистров и суммируется с исходным числом. Затем полученное число сдвигается на 17 регистров вправо и оба этих числа суммируются. Новое число сдвигается на 13 регистров влево и также выполняется суммирование. Все указанные суммирования выполнялись с 64 разрядами. Период полученной последовательности оценивался как 2^{64}. Использование в качестве $M = 2^{32}$ степени числа 2 удешевляет вычисления, но такой выбор может ухудшить качество полученных чисел.

Модифицированный генератор Subtract-with-Borrow аналогичен генератору Фибоначчи с задержкой (константы задержки 27 и 12). Модификация, описанная в [10], позволила увеличить период до 2^{1492}.

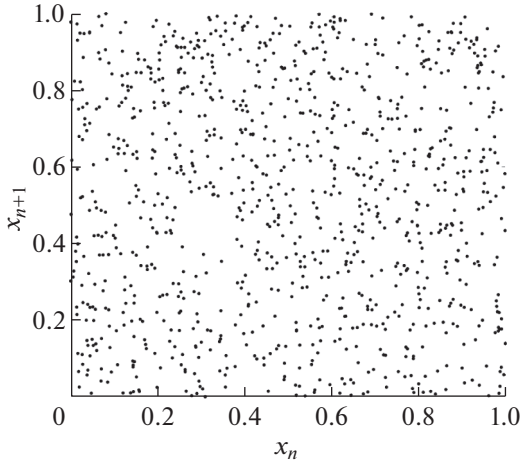
Генератор rand языка C/C++ использует алгоритм Лемера и 16-битовые целые числа. Период такой последовательности не может превышать 2^{16}, т.е. он очень невелик и годится лишь для учебных расчетов. Несмотря на это, генератор очень популярен (быть может, потому, что он является стандартным генератором в C/C++).

Модифицированный алгоритм Лемера использует три независимые последовательности 16-битовых целых чисел. Он был распространен около 1990 г., когда широко использовались 16-разрядные PC XT. Период такого генератора не может превышать 2^{48}.

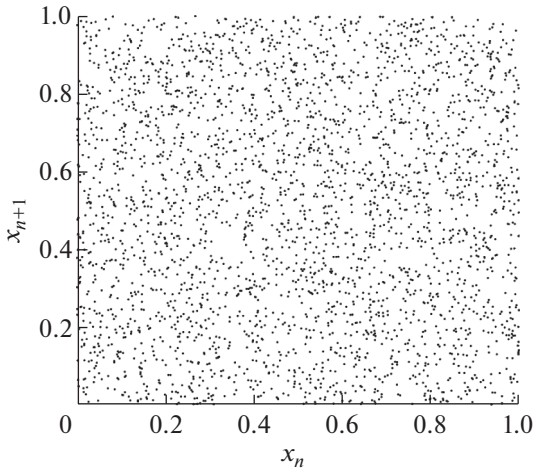
Во всех указанных программах начало последовательности либо детерминировано, либо случайно (и нередко привязывается к часам компьютера).

В 1955 г. американской корпорацией RAND была опубликована книга [11], содержащая миллион случайных цифр и сто тысяч стандартных нормальных отклонений. Эти случайные цифры были получены с помощью электронной рулетки, т.е. физического прибора, а не математического алгоритма.

Особняком стоят последовательности Соболя [12], [13]. Это название широко распространено в зарубежной литературе, а сам автор назвал их $ЛП_\tau$-последовательностями. Эти числа следует называть не псевдослучайными, а квазислучайными. Они строятся так, что при магических числах точек $N = 2^k$ проекции всех точек на любую ось единичного куба различаются и образуют равномерную сетку с шагом $2^{-k} = 1/N$. При промежуточных значениях N часть узлов этой сетки



Фиг. 1. Вихрь Мерсенна, $N = 1000$.



Фиг. 2. Вихрь Мерсенна, $N = 3000$.

остается незанятой. Период такой последовательности неограничен. В [13] приведены таблицы для построения этих последовательностей при размерности пространства $p \le 13$; но в настоящее время такие таблицы разработаны для $p \sim 4000$.

3. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

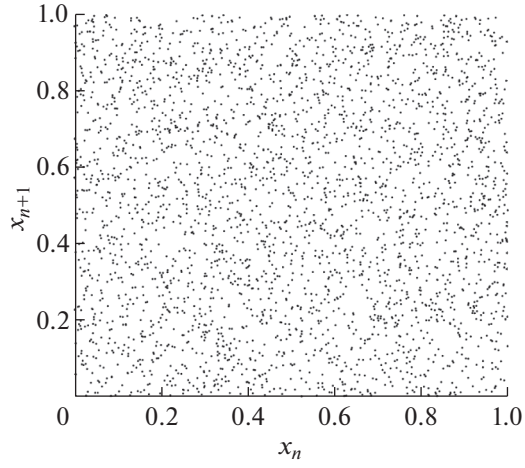
Зависимость от N. Естественно ожидать, что при увеличении N распределение точек будет все более равномерным. Подробные расчеты были проведены для генератора Мерсенна для $N = 100, 300, 1000, 3000, 10\,000$; при дальнейшем увеличении N точки сливались, и визуальное исследование становилось проблематичным. Начало последовательности было одинаковым во всех этих расчетах. Типичные примеры для $N = 1000$ и $N = 3000$ приведены на фиг. 1 и 2 соответственно. На фиг. 1 диаметр точек в $\sqrt{3}$ раз больше, чем на фиг. 2, чтобы суммарная площадь точек была одинаковой.

На обоих рисунках хорошо видны области сильного сгущения и сильного разрежения точек. Важно то, что фиг. 2 строится по той же последовательности с тем же началом, т.е. он получен наложением на фиг. 1 следующего отрезка последовательности длиной 2000. Тем не менее области сгущения и разрежения точек оказываются в других участках квадрата. При дальнейшем увеличении числа точек места сгущения и разрежения также смещаются. Но даже при очень больших $N = 10\,000$ неравномерность заполнения остается значительной. Это свидетельствует о существенной неравномерности заполнения квадрата и о не слишком высоком качестве генератора.

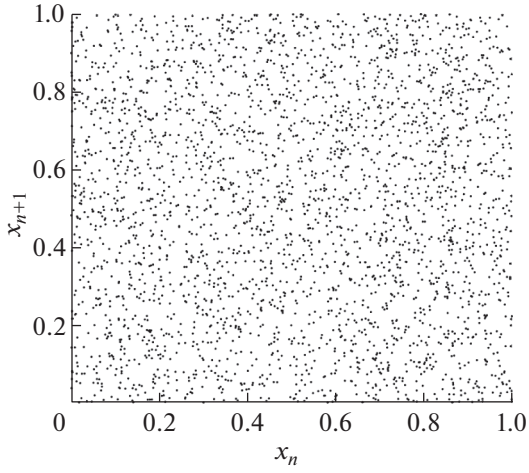
Начало последовательности. На фиг. 3 приведен расчет для генератора Мерсенна с $N = 3000$, но со случайно выбранным началом последовательности. Сравнение с фиг. 2 показывает, что неравномерность заполнения, по-прежнему, достаточно велика, а области наибольшей и наименьшей плотности занимают другие места. Аналогичная картина была при других начальных точках последовательности. Таким образом, все участки последовательности характеризуются неравномерностью распределения точек.

Другие генераторы. На фиг. 4–14 приведены результаты тестирования других 11 генераторов псевдослучайных чисел. Для всех выбрано $N = 3000$. Видно, что картина оказывается такой же, как для генератора Мерсенна. При этом отметим, что одна из этих последовательностей RAND (см. фиг. 14) использует не математические псевдослучайные числа, а электронную рулетку – электронный источник шумов, от которого следовало бы ожидать истинной случайности. Тем не менее конструктивные особенности датчика внесли, по-видимому, неслучайность в результаты.

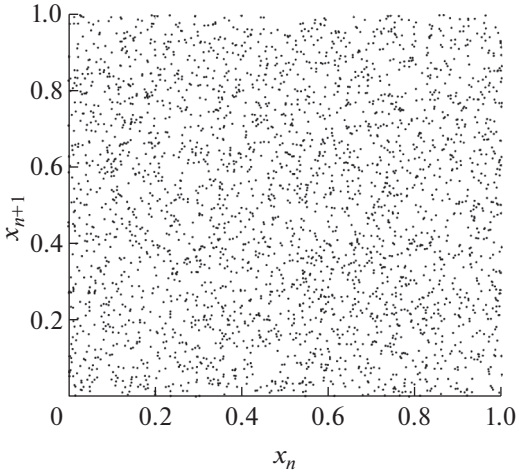
Эти иллюстрации показывают, что различные генераторы псевдослучайных чисел дают качественно сходные результаты. Отчетливо наблюдается неравномерность заполнения квадрата, а места сгущения и разрежения точек зависят от числа точек и выбранного участка последовательности. Нами исследована только часть опубликованных генераторов. Постоянно продолжают



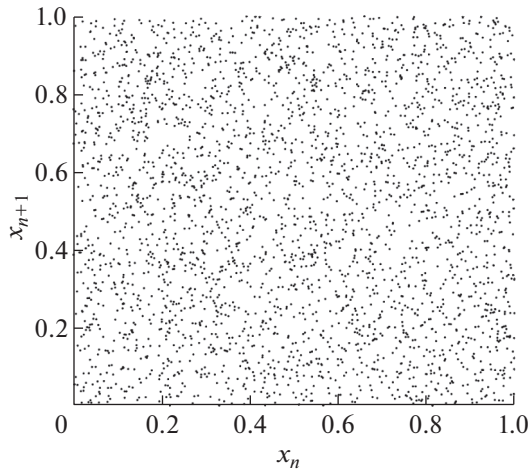
Фиг. 3. Вихрь Мерсенна, $N = 3000$, случайное начало последовательности.



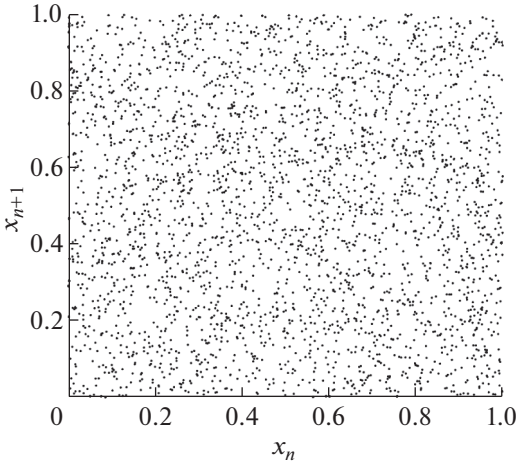
Фиг. 4. Быстрый вихрь Мерсенна, $N = 3000$.



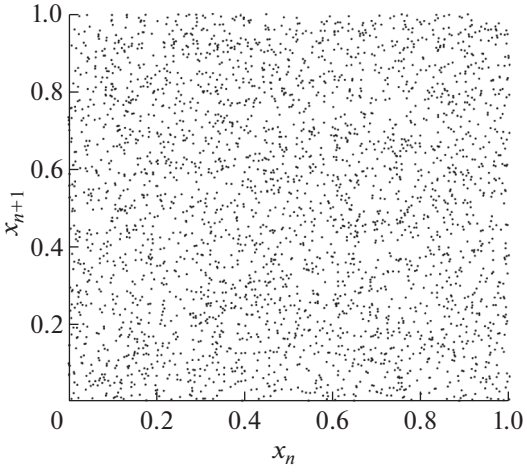
Фиг. 5. Мультипликативный конгруэнтный генератор, $N = 3000$.



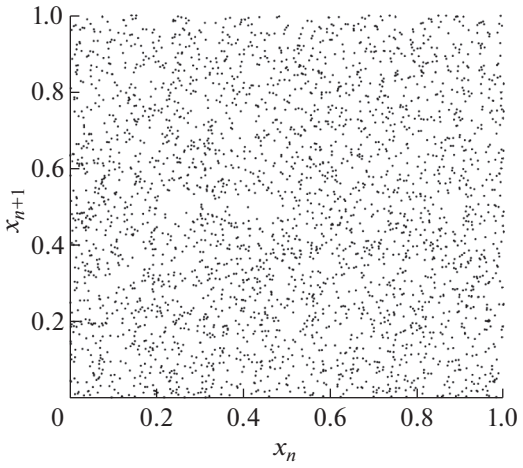
Фиг. 6. Мультипликативный генератор Фибоначчи с запаздыванием, $N = 3000$.



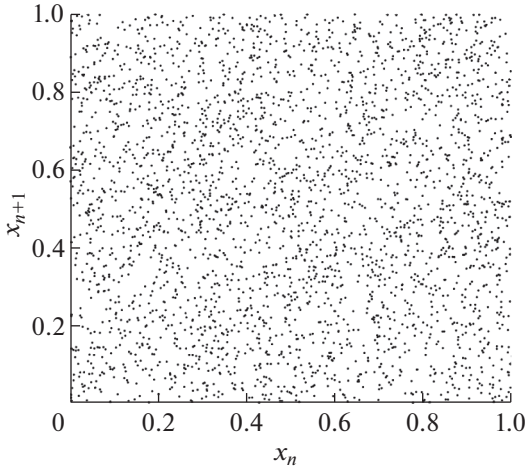
Фиг. 7. Комбинированный множественный рекурсивный генератор, $N = 3000$.



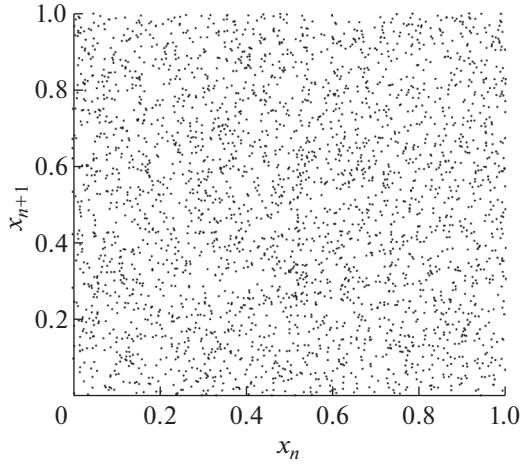
Фиг. 8. Генератор Philox 4 × 32, $N = 3000$.



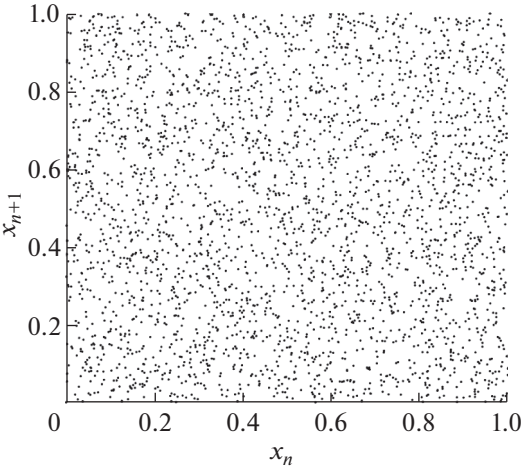
Фиг. 9. Генератор Threefry 4×64, $N = 3000$.



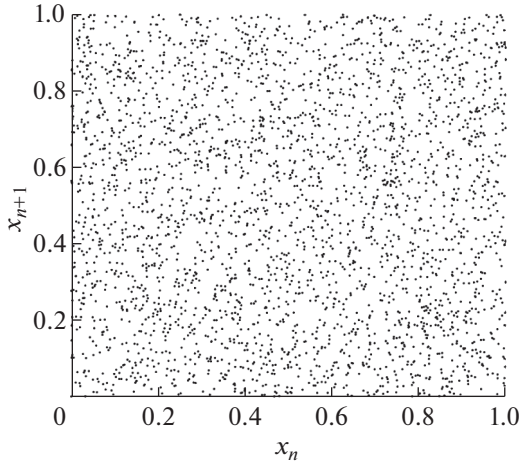
Фиг. 10. Генератор Марсалья, $N = 3000$.



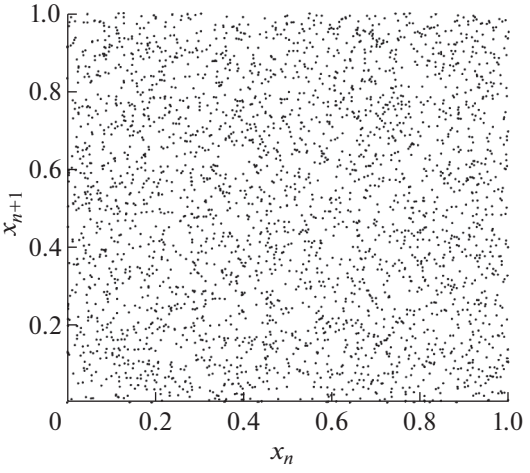
Фиг. 11. Модифицированный генератор Subtract-with-Borrow, $N = 3000$.



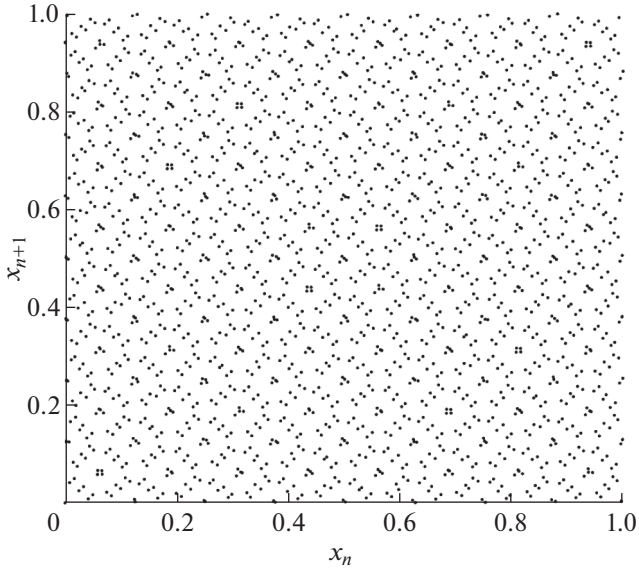
Фиг. 12. Генератор rand языка C/C++, $N = 3000$.



Фиг. 13. Модифицированная последовательность Лемера, $N = 3000$.



Фиг. 14. Последовательность чисел из сборника [11], $N = 3000$.



Фиг. 15. Последовательность Соболя, $N = 2048$.

строиться новые генераторы псевдослучайных чисел. Тем не менее принципиальных прорывов пока что не видно.

Последовательность Соболя. Результаты для нее при магическом числе $N = 2048$ представлены на фиг. 15. Наблюдаемая картина качественно отличается от фиг. 1–14. Видно, что заполнение имеет характер регулярного кружева. Точки в среднем распределены в квадрате гораздо равномернее, чем для псевдослучайных последовательностей, хотя на малых участках квадрата размерами $\sim N^{-1/2}$ их распределение неравномерно.

Поэтому такую последовательность называют не псевдослучайной, а квазислучайной. Интуитивно ясно, что такая последовательность должна дать гораздо лучшие результаты для многомерного численного интегрирования. Однако для других приложений, где требуется существенная случайность (например, криптография), такая последовательность может оказаться неподходящей.

Сетка точек при $N \neq 2^k$ получается из магической сетки выбрасыванием части точек. Интуитивно следует ожидать, что результаты вычисления многомерного интеграла при немагическом числе точек окажутся хуже, чем при магическом.

4. ЗАКЛЮЧЕНИЕ

В настоящее время вряд ли существует какой-то алгоритм построения псевдослучайных чисел, который можно с полной уверенностью использовать для расчета любых случайных процессов. В литературе встречались высказывания о том, что хорошие случайные свойства проявляются только на отрезках очень большой длины $N \sim 10^9$. Вероятно, при меньших N известные алгоритмы можно применять для тех или иных конкретных задач, но такие ситуации нужно дополнительно обосновывать. Например, для многомерных кубатур можно уверенно рекомендовать последовательности Соболя с магическими числами точек.

Поэтому проблема построения хороших генераторов остается нерешенной, а к результатам расчета случайных процессов (метод молекулярной динамики и др.) следует относиться с осторожностью.

Авторы искренне благодарны И.М. Соболю, И.А. Козлитину, Д.Д. Соколову за полезные обсуждения.

СПИСОК ЛИТЕРАТУРЫ

1. *Цветков Е.А.* Эмпирическое исследование статистических свойств некоторых генераторов псевдослучайных чисел // Матем. моделирование. 2011. Т. 23. № 5. С. 81–94.
2. *Matsumoto M., Nishimura T.* Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator // ACM Transactions on Modeling and Comput. Simulat. 1998. V. 8. № 1. P. 3–30.
3. *Matsumoto M., Saito M.* A PRNG Specialized in Double Precision Floating Point Numbers Using an Affine Transition // Monte Carlo and Quasi-Monte Carlo Methods. 2008. https://doi.org/10.1007/978-3-642-04107-5_38
4. *Park S.K., Miller K.W.* Random Number Generators: Good Ones Are Hard to Find // Communicat. of the ACM. 1998. V. 31. № 10. P. 1192–1201.
5. *Mascagni M., Srinivasan A.* Parameterizing Parallel Multiplicative Lagged-Fibonacci Generators // Parallel Computing. 2004. V. 30. P. 899–916.
6. *L'Ecuyer P.* Good Parameter Sets for Combined Multiple Recursive Random Number Generators // Operations Research. 1999. V. 47. № 1. P. 159–164.
7. *Salmon J.K., Moraes M.A., Dror R.O., Shaw D.E.* Parallel Random Numbers: As Easy as 1, 2, 3 // Proc. of the Internat. Conference for High Performance Comput., Networking, Storage and Analys. (SC11). New York: ACM, 2011.
8. *Marsaglia G., Tsang W.W.* The ziggurat method for generating random variables // Journal of Statistical Software. 2000. V. 5. P. 1–7.
9. *Marsaglia G., Zaman A.* A new class of random number generators // Annals of App. Probability. 1991. V. 1. № 3. P. 462–480.
10. *Wichmann B.A., Hill I.D.* An efficient and portable pseudo-random number generator // Appl. Statistics. 1982. V. 31. № 2. P. 188–190.
11. RAND Corporation. A million random digits with 100000 normal deviates. The Free Press, 1955.
12. *Соболь И.М.* О распределении точек в кубе и сетках интегрирования // Успехи матем. наук. 1966. Т. 21. № 5. С. 271–272.
13. *Соболь И.М.* Численные методы Монте-Карло. М.: Наука, 1973.