УДК 519.67, 519.23/.25

П.Ю. Войлов

СРАВНЕНИЕ ЭФФЕКТИВНОСТИ НЕКОТОРЫХ АЛГОРИТМОВ ГЕНЕРАЦИИ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Актуальность проблемы

В практике методов Монте-Карло для решения задачи имитации равномерно распределенных случайных величин используются табличный метод, метод физических датчиков и метод псевдослучайных последовательностей [1, 2], каждый из которых имеет ряд достоинств и недостатков. Наиболее существенными недостатками метода таблиц являются ограниченный запас чисел, а также необходимость выделения значительных объемов памяти для хранения таблицы. Недостатками метода датчиков являются невоспроизводимость генерируемых последовательностей, а также необходимость введения специальных схем или методов для регулярной проверки характеристик генерируемых чисел. Недостатком метода псевдослучайных последовательностей является конечный объем последовательностей (конечная длина периода), наличие скрытых закономерностей и неравномерность распределения. Тем не менее, на практике наиболее часто пользуются именно псевдослучайными последовательностями, что обусловлено рядом объективных факторов [1].

Использование датчиков обусловлено невозможностью воспроизвести последовательность (что требует во время проверки и отладки вычислений сохранения всей используемой последовательности в памяти, и фактически лишает датчики их преимуществ, т.е. неограниченного запаса чисел и отсутствия необходимости использования элементов ОЗУ/ПЗУ). Кроме того, тщательное изучение выборок, полученных с помощью датчиков (основанных на преобразовании шумов Джонсона), показало, что значения полученной таким образом случайной величины не являются независимыми [5]. Следовательно, использование физических датчиков не дает удовлетворительных результатов.

С другой стороны, потребный объем генерируемой последовательности может быть очень велик. Так, например, при моделировании траекторий гамма-квантов в случае многократного рассеяния (типичного для плотных сред), на единственный квант требуются десятки реализаций базовой случайной величины ϖ. Следовательно, при моделировании траекторий частиц, излученных источником с активностью 1 МБк за 60 сек, может потребоваться до $1.6 \cdot 10^9$ независимых реализаций ϖ, что при затрачиваемом объеме памяти 16 или 32 бит на одну реализацию потребует соответственно 3.2-6.5 Гб дискового пространства (или ОЗУ) для хранения (с целью последующего воспроизведения) этих реализаций в

 ISSN 1562-9945

виде таблицы. Следовательно, для решения данной задачи табличный метод практически неприменим.

Существенной проблемой в использовании метода псевдослучайных последовательностей является то, что на данный момент предложено множество алгоритмов получения последовательностей, однако при этом обычно отсутствует сравнительный анализ свойств этих алгоритмов. В данной работе было проведено исследование статистических характеристик и быстродействия ряда распространенных в современной практике алгоритмов генерации псевдослучайных последовательностей.

Исследуемые алгоритмы

В качестве исследуемых алгоритмов были выбраны методы, представляющие все основные классы: конгруэнтные, сдвиговые, алгоритмы Фибоначчи и смешанные. Кроме того, в работе нам представлялось важным преимущественно представить используемые в научной и инженерной практике алгоритмы, по возможности избегая малораспространенных методов. С этой целью были выбраны наиболее часто используемые в практике программирования конгруэнтные алгоритмы rand и srand48, входящие в комплект библиотеки GSL (GNU C), а также все основные алгоритмы из состава библиотеки CLHEP института CERN, которая весьма широко используется в приложениях методов Монте-Карло. Наконец, нам показалось интересным включить в настоящую работу данные по одному из старейших конгруэнтных алгоритмов (Лемера), а так же по перспективным “тороидальным” алгоритмам (Щура) последних лет.

Всего в работе приведены данные исследований по 17 алгоритмам: конгруэнтный Лемера (реализация с параметрами $g = 5^{17}$, $M = 2^{42}$) [1]; конгруэнтные rand и rand48 (48-битный) [6]; сдвиговый Мацумото-Нишимуры (Mersenne Twister, известен также как MT) [7]; смешанный (конгруэнтно-сдвиговый) Тюссанта (DUAL) [8]; сдвиговые Дж.Хурда (HURD160, HURD288) [8]; Фибоначчи-алгоритм Марсальи-Замана (RANMAR) [8]; конгруэнтно-мультипликативный алгоритм Ф.Джеймса (RANEQU) [8]; конгруэнтный Марсальи-Замана (RANLUX) и он же, модифицированный Люшером (RANLUX64) [8]; конгруэнтный Гутброда (RANSHI) [8]; смешанный CLHEP (TRIPLE) [8]; “тороидальные” Щура (GS и GM19) [9]; сдвиговый П.Лекуйе (LFSR113) [10]; смешанный П.Лекуйе (MRG32K3A) [9].

Для каждого алгоритма генерировалось либо 6 последовательностей с разными начальными значениями (причем каждое значение представляло собой комбинацию цифр из таблицы случайных чисел), либо вся последовательность разбивалась на 6 частей. Объемы обрабатываемых последовательностей приведены ниже в описании тестов.

Проверка быстродействия

Проверка быстродействия алгоритмов была проведена путем сравнения времени, потребного каждому из них на генерацию последовательно-

сти из 10^7 32-битных слов. Результаты проверки приведены в таблице 1:

Таблица 1

Табл. 1 – Тест быстродействия

Алгоритм	Время	Алгоритм	Время	Алгоритм	Время
Лемера	3.6	rand	1.8	RANLUX64	8.7
rand48	4.7	HURD160	1.2	RANSHI	1.0
DUAL	1.9	HURD288	1.1	TRIPLE	2.8
RANMAR	3.6	RANEQU	2.6	GS	18.1
MT	1.1	RANLUX	11.7	GM19	35.9
MRG32K3A	5.7	LFSR113	2.1		

Здесь в качестве единицы выбрано время, показанное наиболее быстродействующим алгоритмом. К сожалению, приведенные результаты не могут служить объективным показателем быстродействия того или иного алгоритма, поскольку существенно зависят от эффективности выполнения их программных реализаций, а также от ряда других факторов (например, для конгруэнтного метода Лемера – от эффективности программной реализации методов работы с 64-битными словами в данном комплекте библиотек и т.п.). Кроме того, стоит отметить, что авторами алгоритмов GS и GM19 разработаны также оптимизированные программные реализации с использованием ассемблерных вставок, позволяющие значительно сократить время вычислений, однако мы намеренно рассматривали лишь неоптимизированные реализации, поскольку для большинства остальных алгоритмов такой оптимизации выполнено не было.

Проверка статистических характеристик

С помощью системы статистических тестов проверялась справедливость гипотезы H_0, заключающейся в том, что набор чисел $\mathbf{x}^{(L)} = ||x_0, x_1, \ldots, x_{L-1}||$, является реализацией L-мерного случайного вектора $\xi^{(L)} = ||\xi_0, \xi_1, \ldots, \xi_{L-1}||$, равномерно распределенного в L-мерном единичном кубе:

$$G_L(0 \leqslant \quad X_i < 1, i = 0, 1, \ldots, L-1)(1) \tag{1}$$

где X_i – возможные значения проекций случайного вектора $\xi^{(L)}$.

Проверка равномерности многомерного распределения проводилась по анализу гистограмм, для чего проверялись отрезки последовательности x_i, состоящие из s подряд идущих чисел, с использованием пересекающихся наборов

$$||x_0, x_1, \ldots, x_{s-1}||,\ ||x_1, x_2, \ldots, x_s||,\ ||x_2, x_3, \ldots, x_{s+1}|| \ldots \tag{2}$$

Поскольку зависимость выборок x не позволяет непосредственно использовать для проверки согласия, использовалось то обстоятельство, что в случае двумерного распределения квадратичная форма (3):

$$Q = Q_{r^2} - Q_r \tag{3}$$

имеет распределение χ^2 с $k \cdot (k-1)$ степенями свободы в том случае, если справедлива гипотеза H_0. Согласно [4], в общем виде это условие должно выполняться для квадратичных форм $Q_n^2 = Q^2 - Q^{r^2}$ безотносительно порядка n (4):

$$Q_n = Q_{r^n}^2 - Q_{r^{n-1}}^2 \tag{4}$$

где $Q_{r^n} = \sum\limits_{i_1,i_2,\ldots,i_n} \frac{(\omega_{i_1,i_2,\ldots i_n} - N \cdot p_{i_1} \cdot p_{i_2} \cdot \ldots \cdot p_{i_n})^2}{N \cdot p_{i_1} \cdot p_{i_2} \cdot \ldots \cdot p_{i_n}}$; $\omega_{i_1,i_2 \ldots i_n}$ – эмпирическая частота вектора с координатами $||i_1, i_2, \ldots, i_n||$; N – объем выборки.

В данной работе тест проверялся для порядков $n = 3$ и $n = 4$, результаты приведены в таблице 2.

В тесте автокорреляционных функций [2], рассматривая идеальную базовую последовательность как реализацию стационарной некоррелированной случайной последовательности ξ_1, при достаточно большой выборке N значения эмпирической автокорреляционной функции (5)

$$\tilde{B}(\tau) = \frac{1}{N-1} \sum_{i=0}^{N-1} x_i x_{i+\tau} - \frac{1}{N(N-1)} \left(\sum_{i-0}^{N-1} x_i \right)^2, \tau = 0, 1, \ldots, \tau_m \tag{5}$$

должны быть близки к значениям теоретической корреляционной функции (6):

$$B(\tau) = \begin{cases} \frac{1}{12}, & \tau = 0 \\ 0, & \tau \neq 0 \end{cases} \tag{6}$$

Тогда неравенством (7) можно определить область $G_S^{(0)}$ в $N + \tau$-мерном гиперкубе G_S:

$$|\tilde{B}(\tau) - B(\tau)| \leqslant t_p \frac{\chi}{12\sqrt{N-1}} \tag{7}$$

где $\chi = \sqrt{12}$ для $\tau = 0$ и $\chi = 1$ для $\tau \geqslant 1$, теоретическая вероятность попасть в которую равна p, если справедлива гипотеза H_0. В таблице 2 приведены результаты теста для $N = 2000$ и $N = 10000$.

В тесте битового потока случайная последовательность на выходе алгоритма рассматривается как поток 20-битных перекрывающихся последовательностей вида $\overline{i+1; i+20}$, $\overline{i+2; i+21}$, …. Тогда, рассматривая поток из 2^{21} таких последовательностей (общей длиной $2^{21} + 19$ бит), проверяется гипотеза о том, что число комбинаций, отсутствующих среди 2^{20} возможных в выборке объемом 2^{21}, распределено по нормальному закону с функцией плотности вероятности [5]:

$$f(x) = \frac{1}{428\sqrt{2\pi}} e^{-\frac{(x-141909)^2}{366368}} \tag{8}$$

Повторяя вычисления двадцатикратно для битового потока длиной $20 \cdot (2^{21} + 19)$ бит (5242927.5 байт), гипотеза нормальности распределения с указанными параметрами проверялась по критерию χ^2 (для уровня значимости $\alpha = 0.05$) на 6 различных фрагментах сгенерированной последовательности (итого для каждого алгоритма получено 120 значений). Всего для теста битового потока (17 алгоритмов) получена и обработана выборка из 2040 элементов, гистограммы см. рис. 1:

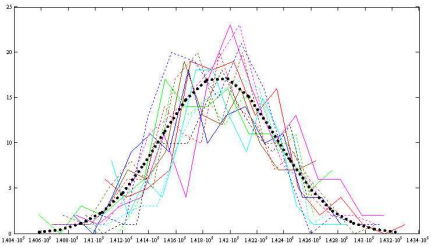

Рис. 1 – Гистограммы распределения комбинаций в тесте битового потока, эталонное распределение (11) выделено жирной линией.

В тесте кратчайших расстояний случайная последовательность на выходе алгоритма рассматривается как последовательность координат n точек, заполняющих квадрат случайным образом. В этом случае квадрат кратчайшего расстояния между всеми парами из множества точек есть случайная величина, распределенная экспоненциально, причем для квадрата стороной 10000 и $n = 8000$ математическое ожидание экспоненциального распределения есть $M = 0.995$) [5]. Отсюда функция плотности распределения величины кратчайшего расстояния между точками (9):

$$f(x) = \frac{1}{0.995} \cdot e^{-\frac{x}{0.995}} \tag{9}$$

Повторяя моделирование попаданий точек 100 раз для потока из $8000 \cdot 2 \cdot 100 = 1.6 \cdot 10^6$ случайных чисел, гипотеза экспоненциального распределения с указанным параметром проверялась по критерию χ^2 (для уровня значимости $\alpha = 0.05$) на 6 различных фрагментах сгенерированной последовательности (итого для каждого алгоритма получено 600 значений квадрата кратчайшего расстояния). Всего для теста битового потока (17 алгоритмов) получена и обработана выборка из 10200 элементов, гистограммы см. рис. 2:

Следующие три теста имели целью проверить равномерность одномерного распределения элементов. В общем виде [2] в тесте одномерного распределения множество возможных значений элементов вектора $x^{(L)}$ разбивается на n одинаковых интервалов $G^{(r)}$ и проверяется гистограмма распределения $\frac{r}{n} < X < \frac{r+1}{n}$, $r = 0, 1, \ldots, (n-1)$ на соответствие эмпирических частот $\frac{m_r}{N}$ теоретическим вероятностям $p_r = \frac{1}{n}$.

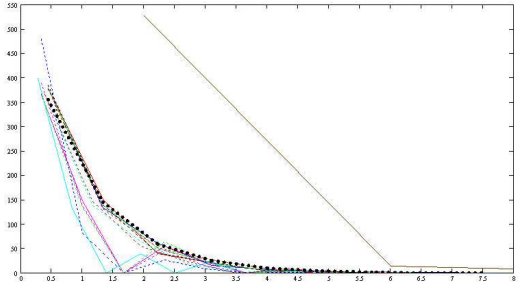

Рис. 2 – Гистограммы распределения комбинаций в тесте кратчайших расстояний, хорошо видно неудовлетворительное описание многими алгоритмами эталонного распределения (12) (выделено жирной линией).

Непосредственная реализация этого метода на ЭВМ требует значительных объемов памяти; это связано с необходимостью выделять большие массивы данных для хранения эмпирических частот, и, очевидно, массивы тем больше, чем меньше размерность интервала. Для устранения этого ограничения был использован метод [4], заключающийся в рассмотрении не частот как таковых, а числа интервалов, для которых эмпирическая частота $m_r = 0$, причем эта величина распределена по нормальному закону с вполне определенными параметрами. Для проверки использовались три варианта этого теста.

В тесте перекрывающихся пар случайная последовательность на выходе алгоритма рассматривается как поток из перекрывающихся пар 10-битных символов. Тогда, анализируя поток из 2^{21} таких 20-битных комбинаций (т.е. общая длина исследуемой последовательности составляет 5242880 байт), проверяется гипотеза о том, что число пустых интервалов распределено по нормальному закону с функцией плотности вероятности [4]:

$$f(x) = \frac{1}{290\sqrt{2\pi}} e^{-\frac{(x-141909)^2}{168200}} \qquad (10)$$

Рассматривая выход генератора как последовательность 32-битных слов, тест повторялся для 23 различных смещений стартового бита 10-битного отрезка в 32-битном исходном слове. Гипотеза нормальности распределения с указанными параметрами проверялась по критерию χ^2 (для уровня значимости $\alpha = 0.05$) на 6 различных фрагментах сгенерированной последовательности (итого для каждого алгоритма получено 138 значений). Всего для 17 алгоритмов получена и обработана выборка из 2346 элементов, гистограммы см. рис. 3 (хорошо видны плохие результаты, показанные конгруэнтными алгоритмами Лемера, rand и rand48):

В тесте перекрывающихся четверок случайная последовательность на выходе алгоритма рассматривается как поток из перекрывающихся четверок 5-битных символов. Анализируя поток из 2^{21} таких 20-битных

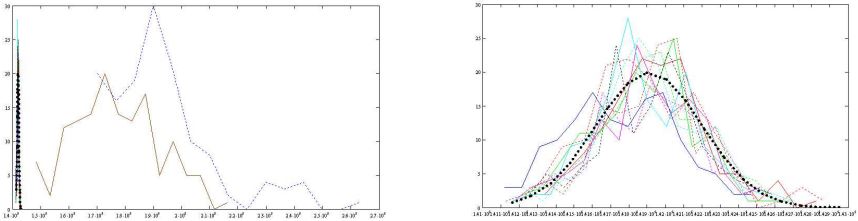

Рис. 3 – Общий вид (слева) и область $\mu \pm 3\sigma$ (справа) гистограмм распределения в тесте перекрытия пар. Эталонное распределение (13) выделено жирной линией.

комбинаций, проверялась гипотеза о том, что число пустых интервалов распределено по нормальному закону с функцией плотности вероятности [4]:

$$f(x) = \frac{1}{295\sqrt{2\pi}} e^{-\frac{(x-141909)^2}{174050}} \tag{11}$$

Рассматривая выход генератора как последовательность 32-битных слов, тест повторялся для 28 различных смещений стартового бита 5-битного отрезка в 32-битном исходном слове, итого для 17 алгоритмов получена и обработана выборка из 2856 элементов, гистограммы см. рис. 4 (хорошо видны плохие результаты, показанные конгруэнтными алгоритмами Лемера, rand и rand48):

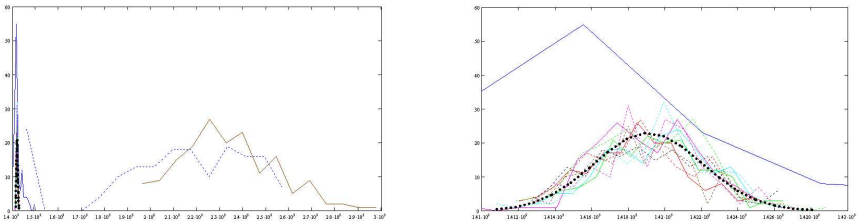

Рис. 4 – Общий вид (слева) и область $\mu \pm 3\sigma$ (справа) гистограмм распределения в тесте перекрытия четверок. Эталонное распределение (14) выделено жирной линией.

Наконец, в тесте выборочных пар случайная последовательность на выходе алгоритма рассматривается как поток из 10-символьных комбинаций 2-битных символов (итого 2^{20} возможных комбинаций). Тогда, рассматривая поток из 2^{21} таких комбинаций, проверяется гипотеза о том, что число пустых интервалов распределено по нормальному закону с функцией плотности вероятности [4]:

$$f(x) = \frac{1}{339\sqrt{2\pi}} e^{-\frac{(x-141909)^2}{229842}} \tag{12}$$

 ISSN 1562-9945

Рассматривая выход генератора как последовательность 32-битных слов, тест повторялся для 31 различных смещений парных битов, итого для 17 алгоритмов получена и обработана выборка из 3162 элементов (для 6 фрагментов сгенерированной последовательности), гистограммы см. рис. 5 (хорошо видны плохие результаты, показанные конгруэнтными алгоритмами Лемера, rand и rand48):

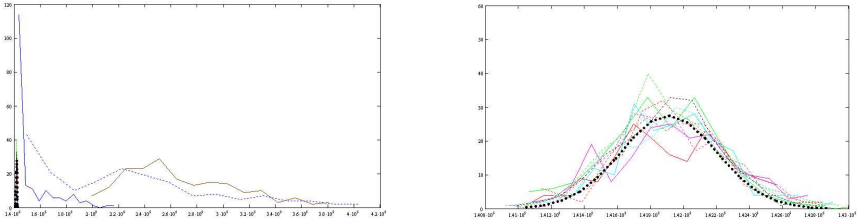

Рис. 5 – Общий вид (слева) и область $\mu \pm 3\sigma$ (справа) гистограмм распределения в тесте выборочных пар. Эталонное распределение (15) выделено жирной линией.

Обработка результатов тестов

Результаты всех тестов сравнивались с эталонными распределениями по критерию Пирсона, причем согласно [3] интервалы группировались в каждом случае так, чтобы расчетная частота в каждой группе была не меньше 5. По вычисленным значениям критерия χ^2 находилась вероятность $P = 1 - F_{\chi^2}(\chi^2, f)$, алгоритм отвергался при $P < 0.05$. Результаты обработки см. в таблице 2 (отвергнутые алгоритмы помечены знаком “+”).

Отметим, что некоторой условной мерой для сравнения алгоритмов по данной таблице может служить произведение равновзвешенных вероятностей P (по результатам тестов) и его среднее геометрическое (при этом отвергнутым тестам можно присваивать значение $P = 0.05$). Однако для использования этих величин в качестве критерия эффективности алгоритма необходимо допустить, что все тесты одинаково близко описывают задачу, в которой будет применяться выбранный алгоритм. На практике это, безусловно, не так, поэтому такая равновзвешенная оценка весьма условна. Тем не менее, по этому критерию можно выделить три “наилучших” алгоритма – это MT, RANEQU и DUAL (отметим, что это также одни из самых быстродействующих алгоритмов). Тремя “худшими” алгоритмами являются rand48, алгоритм Лемера и rand.

Выводы

Можно однозначно говорить о том, что наиболее широко распространенные алгоритмы rand и rand48 [6], а также весьма распространенный в предыдущие десятилетия алгоритм Лемера [1], не отвечают требованиям тех приложений, где требуется статистически достоверная равномерность распределения псевдослучайных последовательностей. По совокупности результатов теста быстродействия и статистических тестов

Таблица 2

Табл. 2 – Тесты равномерности распределения

Алгоритм	Время	k^{ac}_{2000}	k^{ac}_{10000}	$T_{3/2}$	$T_{4/3}$	$T_{\text{БП}}$	$T_{\text{КР}}$	$T_{\text{ПП}}$	$T_{\text{ПЧ}}$	$T_{\text{ВП}}$
DUAL	1.92	0.90	0.81	0.63	0.61	0.55	0.09	0.88	0.19	0.17
rand48	4.67	0.35	0.65	0.92	0.28	0.46	+	+	+	+
HURD160	1.21	0.87	0.55	0.76	+	0.46	0.23	0.48	0.81	0.29
HURD288	1.11	0.81	0.91	0.43	0.56	+	+	0.70	0.19	+
RANMAR	3.57	0.85	0.83	0.60	0.86	0.17	+	0.30	0.31	0.34
Лемера	3.64	0.43	0.92	0.30	+	0.43	+	+	+	+
MT	1.14	0.67	0.80	0.54	0.85	0.63	0.37	0.60	0.30	0.29
Rand	1.75	0.87	0.77	0.95	+	0.15	+	+	+	+
RANEQU	2.61	0.86	0.81	0.92	0.26	0.44	+	0.97	0.31	0.65
RANLUX	11.70	0.34	0.53	0.96	0.19	0.88	+	0.50	0.08	0.75
RANLUX64	8.70	0.36	0.73	0.86	0.71	0.13	+	0.59	0.10	0.86
RANSHI	1.00	0.85	0.88	0.17	0.54	0.37	+	0.31	0.11	0.98
TRIPLE	2.75	0.71	0.64	0.26	0.56	0.35	+	0.20	0.26	0.98
MRG32K3A	5.7	0.83	0.51	0.47	0.74	0.45	+	0.11	0.72	0.82
LFSR113	2.1	0.68	0.85	0.95	+	0.95	+	0.52	0.06	0.48
GS	18.1	0.90	0.61	0.79	0.86	0.19	0.07	0.23	0.79	+
GM19	35.9	0.64	0.92	0.73	0.81	0.71	+	0.09	+	0.44

мы считаем возможным рекомендовать к использованию в таких приложениях алгоритм Mersenne Twister [7], период которого весьма велик ($2^{19937}-1$), программная реализация не представляет сложностей и, кроме того, может быть оптимизирована. В начале 2007 г. создана более быстродействующая (в 2 раза) версия алгоритма (SFMT) с поддержкой векторной архитектуры SIMD.

Отметим, что к этим оценкам следует относиться с осторожностью, поскольку, строго говоря, они характеризуют свойства вполне конкретных последовательностей (с определенными начальными значениями), с применением вполне конкретных тестов.

Литература

1. Соболь И.М. Численные методы Монте-Карло. М., Наука, 1973. – с.312.
2. Полляк Ю.Г. Вероятностное моделирование на электронных вычислительных машинах. М., Советское радио, 1971. – с.400.
3. Шторм Р. Теория вероятностей. Математическая статистика. Статистический контроль качества. М., “Мир” 1970. – с.368.
4. Marsaglia G. Monkey Tests for Random Number Generators. Computers & Mathematics with Applications, 9, 1-10, 1993.

5. Marsaglia G. A Current View of Random Number Generators. Keynote Address, Computer Science and Statistics: 16th Symposium on the Interface, Atlanta, 1984.
6. http://www.gnu.org/software/gsl/manual/gsl-ref.html
7. http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html
8. http://cern.ch/clhep
9. Barash L., Shchur L.N. Periodic orbits of the ensemble of Sinai-Arnold cat maps and pseudorandom number generation. Physical Review E 73, 036701 (2006).
10. http://web.umr.edu/~vojtat/class_351//lfsr113/

Получено 12.11.2007