Таким образом, разработана методика расчета уровней ослабления побочных электромагнитных излучений технических средств на основе измерений в ближней зоне излучателя не только поперечных, но и продольной компоненты напряженности электрического поля.

**Литература**

1. Kuhn Markus. Compromising emanations: eavesdropping risks of computer displays // Technical Report № 577, UCAM-CL-TR-577, University of Cambridge, Computer Laboratory, 2003.

2. Хорев А.А. Оценка возможности по перехвату побочных электромагнитных излучений видеосистемы компьютера. Часть 2 // Специальная техника. 2011. № 4. С. 51-62.

3. Бузов Г.А., Калинин С.В., Кондратьев А.В. Защита от утечки информации по техническим каналам. Учебное пособие. М.: Горячая линия – Телеком, 2005. 416 с.

4. Никольский В.В. Электродинамика и распространение радиоволн. Учебное пособие. М.: Наука. 1973. 607 с.

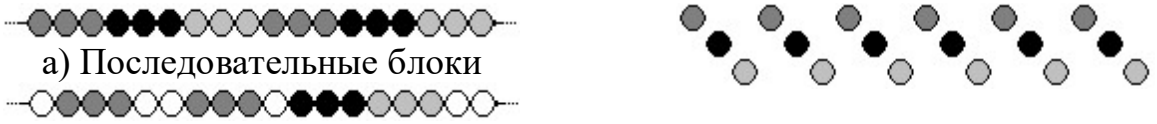
К.Е. Климентьев

## ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗИРОВАННОЙ ПАРАМЕТРИЗАЦИИ ЛИНЕЙНЫХ КОНГРУЭНТНЫХ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

(Самарский университет)

**Введение.** На кафедре ИСТ Самарского Университета силами преподавателей и студентов продолжается разработка, реализация и модификация программной среды для моделирования поведения «саморазмножающихся» сущностей [1]. Попытки моделирования эпидемии вируса 2019–nCov продемонстрировали количественные ограничения существующей на текущий момент реализации и, как следствие, необходимость доработки ее в направлении распараллеливания вычислений. В настоящей работе приводится описание концепции программного генератора псевдослучайных чисел, ориентированного, в отличие от рассмотренных в работе [2], на вычисления, распределенные на большое количество процессоров.

**1. Предварительный обзор и постановка задачи**. Задача параллельного порождения множества потоков псевдослучайных чисел давно и хорошо известна (см., например [17]). Должны выполняться два условия: 1) потоки чисел, используемые разными процессорами, должны иметь сходные (в идеале – идентичные) статистические свойства; 2) эти потоки не должны пересекаться.



а) Последовательные блоки

б) Случайные блоки

в) Лягушачьи прыжки

Рис. 1. Методы распараллеливания числовых потоков

На практике для выполнения этих условий используются разные подходы с использованием общего генератора: 1) разбиение числового потока на последовательно расположенные блоки (см. рис. 1,а); 2) разбиение числового потока на случайно расположенные блоки (см. рис. 1,б); 2) «лягушачьи прыжки» (см. рис. 1,в). Используется также подход, имеющий в виду применение множества независимых, но однотипных (фактически, одинаковых) генераторов с различными числовыми параметрами. Процесс построения генератора путем наполнения общего «шаблона» индивидуальным набором числовых параметров называется его «параметризацией».

Такими «шаблонами» могут служить XORSHIFT (в работе [8] опубликованы несколько сотен числовых параметров для построения различных генераторов) и MT (в работе [18] описано средство для генерации различных числовых параметров «упрощенных» версий генератора). Но проще всего «параметризации» поддаются линейные конгруэнтные генераторы, работа которых основана на формуле

$$x_i := (a \times x_{i-1} + c) \bmod m,$$

где константы «a», «c» и «m» – суть числовые параметры генератора. Генераторы с $a \neq 0$, $c \neq 0$ называются «смешанными», генераторы с $c=0$ – «мультипликативными» Общие критерии выбора множителя «a» описаны в [3], там же приводится правило выбора: $a=8\times t+5$ или просто $a=4\times t+1$, где t – некое целое – для случая, если «m» есть степень двойки. Сдвиг «c» при этом может быть любым нечетным числом. Однако не все множители «a», соответствующие этой формуле, «равноценны». Важным критерием качества числовой последовательности, порожденной линейным конгруэнтным генератором, является равномерность распределения в n–мерном пространстве точек, координаты которых образованы группами соседних элементов последовательности. Например, для случая двумерной плоскости координаты точек: $(x_0,x_1)$, $(x_1,x_2)$, $(x_2,x_3)$ и т.д. Известно (см. в [3] со ссылкой на [7]), что несмотря на хаотичный порядок появления точек, все они лежат на параллельных гиперплоскостях размерности n–1, например, если речь идет о двумерной плоскости, то точки лежат на параллельных прямых. Эффект «неравноценности» различных множителей проиллюстрирован рис. 2, из которого можно заключить, что множитель $a=5$ (см. рис. 2,б) в 2–мерном пространстве лучше, чем $a=1$ (см. рис. 2,а).


а) $x_i = (1x_{i-1}+5) \mod 8$ б) $x_i = (5x_{i-1}+5) \mod 8$

Рис 2. Эффект «неравноценности» множителей

Для исследования равномерности распределения точек и, соответственно, «качества» параметра «a», разработано несколько формальных критериев [16]: критерий расстояния между точками (подробно описан в [3]); критерий расстояния между параллельными гиперплоскостями; критерий количества гиперплоскостей; критерий «степени расхождения» (discrepancy). В принципе, с точки зрения классификации множителей «a» на «хорошие» и «плохие», все они приводят к одинаковым результатам. Поэтому достаточно применить только один из критериев, например, описанный в [3].

За последние десятилетия проведены многочисленные, часто очень ресурсоемкие исследования, направленные на поиск «хороших» множителей для разных «m» (см., например [10–15] и др.). Часть результатов доступна в Интернете непосредственно (например, [10]), малую часть иных можно обнаружить в многочисленных литературных источниках (например, в [5, 6, 9]), часть (например, результаты работы [13]) отсутствует в открытом доступе. Однако общее количество опубликованных и доступных конкретных числовых значений множителя «a» (а именно, всего несколько десятков) недостаточно для построения сотен и тысяч параллельно работающих вариантов.

Отсюда вытекает задача: воспроизвести и программно реализовать методику поиска «хороших» множителей «a» для «смешанных» конгруэнтных генераторов с модулями вида $m=2^t$, где t=32 или t=64. Отметим также, что эта методика будет непригодна для поиска множителей «a» в случае «мультипликативных» генераторов с простыми (неразложимыми на сомножители) модулями. Также упомянем, что программы расчетов по упомянутым выше критериям «качества» параметра «a» доступны в Интернете, в том числе и в виде исходных текстов, но они базируются на математической библиотеке «gmp» и реализованы в среде UNIX–подобных ОС, что сильно ограничивает их практическую применимость.

**2. Описание метода.** Итак, существует формальный метод, позволяющий рассчитать «качество» линейного конгруэнтного генератора, основываясь только на значениях множителя «a» и модуля «m» (см. [3]). Предполагается, что все псевдослучайные числа расположены в диапазоне от 0 до 1. «Наихудшее» расстояние между всевозможными парами точек в n–мерном пространстве есть $D_n=1/v_n=1/(d_1^2+d_2^2+\ldots d_n^2)^{1/2}$, где $-m/2 \le d_i \le m/2$ и $d_i \neq 0$. Метод вычисляет минимум «волнового числа»: $v_n=\min(d_1^2+d_2^2+\ldots d_n^2)^{1/2}$, где $d_i$ выбираются из соотношений:

$(d_1 + a \times d_2) \bmod m = 0$, для n=2;

$(d_1 + a \times d_2 + a^2 \times d_3) \bmod m = 0$, для n=3;

$(d_1 + a \times d_2 + a^2 \times d_3 + a^3 \times d_4) \bmod m = 0$, для n=4 и т.д.

Очевидно, что при этом всегда $D_1=1/m$ и $v_1=m$. Обычно для практических нужд хватает значений $v_2$, $v_3$ и $v_4$. Однако иногда рассчитывают так же $v_5$, $v_6$ и даже значения более высоких порядков. ЛКМ–генератор, у которого все $v_i$ большие, «качественней» генератора, у которого все они малы. Чтобы сравнивать «качество» генераторов с различными модулями «m», величины $v_i$ приводят к единому масштабу, пересчитывая их в $\mu_i$ (см. табл. 1).

Таблица 1 – Масштабирование «волновых чисел»

| **Размерность *i*** | **2** | **3** | **4** |
|---|---|---|---|
| $\mu_i$ | $\pi v_2^2/m$ | $4\pi v_3^3/3m$ | $\pi^2 v_4^4/2m$ |

В соответствии с [3], считается вполне «удовлетворительным» и пригодным к практическому использованию, если $\mu_i \geq 0.1$, однако мы ужесточим требования и будем считать «хорошими» генераторы с множителями «a», для которых $\mu_i \geq 1.0$.

**3. Описание реализации**. Поиск $v_i$ можно выполнять и полным перебором вариантов $d_i$, но при i>2 такой подход становится чрезвычайно ресурсоемким, поэтому был использован алгоритм, подробно описанный в [3]. При реализации этого алгоритма на C/C++ потребовалось применение «длинной арифметики», с этой целью были доработаны примеры, опубликованные в работе [4]. Исходные тексты разработаны в соответствии со стандартом ANSI C (он же C89), так что могут равно компилироваться как в среде Windows (на компиляторах Microsoft и Borland), так и в среде UNIX–подобных ОС (на компиляторах GCC). В частности, для Windows реализована DLL–библиотека, которая может быть подключена к любому программному проекту, разработанному на любом компилируемом в машинный код языке программирования (например, C/C++, Pascal, Modula-2 и т.п.).

**4. Обсуждение некоторых результатов**. В процессе тестирования реализованного программного средства были найдены несколько десятков тысяч «хороших» множителей «a» для $m=2^{32}$ и $m=2^{64}$. В табл. 2. приведены некоторые из них, причем особое внимание уделено «красивым», то есть легко запоминаемым числам. Кроме того, в таблице, с целью сравнения, приведен ряд «хороших» значений константы «a», почерпнутых из литературных источников, а также «плохой», наобум выбранный множитель.

Анализ Табл. 2 показывает, что самостоятельно найденные множители с точки зрения «спектрального критерия» мало уступают лучшим известным образцам, но превосходят результат «случайного» выбора. Разработанное средство позволяет находить такие множители массово. Измеренное по методике [19] время поиска очередной константы даже на процессорах с тактовой частотой 1.6МГц не превышает 0.001 с.

Таблица 2 – «Волновые» свойства некоторых множителей

| **a** | **m** | $\mu_2$ | $\mu_3$ | $\mu_4$ | $\mu_5$ | $\mu_6$ | **Комментарий** |
|---|---|---|---|---|---|---|---|
| 3141592653 | $2^{32}$ | 0.03 | 1.0 | 0.8 | 0.05 | 1.6 | наобум |
| 1664525 | $2^{32}$ | 3.6 | 3.4 | 4.7 | 1.3 | 1.3 | [3,6] |
| 69069 | $2^{32}$ | 3.1 | 2.9 | 3.2 | 5.0 | 0.02 | Super Duper |
| 333888333 | $2^{32}$ | 1.6 | 2.6 | 1.0 | 2.7 | 6.0 | самостоятельно |
| 333333333 | $2^{32}$ | 1.2 | 2.3 | 3.5 | 1.9 | 2.4 | самостоятельно |
| 77777777 | $2^{32}$ | 3.2 | 1.0 | 5.5 | 1.9 | 1.2 | самостоятельно |
| 500000005 | $2^{32}$ | 1.5 | 3.5 | 3.3 | 3.1 | 3.1 | самостоятельно |
| 6364136223846793005 | $2^{64}$ | 1.5 | 3.7 | 4.7 | 4.0 | 1.8 | [3] |
| 6906969069 | $2^{64}$ | 1.8 | 4.3 | 3.0 | 2.7 | 2.3 | Super Duper |
| 194519451945 | $2^{64}$ | 1.9 | 1.7 | 3.2 | 1.5 | 2.9 | самостоятельно |
| 1945194519451945 | $2^{64}$ | 1.6 | 3.2 | 3.5 | 1.3 | 2.4 | самостоятельно |

**Заключение**. Достоинства и недостатки линейных конгруэнтных генераторов подробно обсуждены, например, в [3]. Описанное средство позволяет облегчить:

- построение сотен и тысяч однотипных генераторов, используемых в системах распределенных вычислений;
- увеличение периода отдельных генераторов путем «горячей замены» параметрических констант по истечении текущего периода.

## Литература

1. Климентьев К.Е. Мультиагентное моделирование процессов распространения и взаимодействия инфицирующих сущностей // Программные продукты и системы. – Тверь, 2018. – 1(31) – С. 744–748.

2. Климентьев К.Е. Выбор и реализация программного генератора псевдослучайных чисел для системы мультиагентного моделирования // Международная научно–техническая конференция «Перспективные информационные технологии (ПИТ 2019)», 2019. — С. 52–58.

3. Кнут Д. Искусство программирования, том 2. Получисленные алгоритмы, 3 изд. – М.: Издательский дом «Вильямс», 2001. – 832 с.

4. Домашев А.В., Грунтович М.М., Попов В.О. и др. Программирование алгоритмов защиты информации. – М.: Нолидж, 2002. – 416 с.

5. Press E. et al. Numerical Recipes. Third Edition. – Cambridge University Press, 2007. – 1235 pp.

6. ГОСТ Р ИСО 28640–2012. Статистические методы. Генерация случайных чисел. – М.: Стандартинформ, 2014. – 35 с.

7. Marsaglia G, Random numbers fall mainly in the plane // Proc. Nat. Acad. Sei, USA, 1968 – Pp. 25–28.

8. Marsaglia G. Xorshift RNGs // Journal of Statistical Software. Vol. 8 (14), 2003.

9. L’Ecuyer P. Tables of linear congruential generators of different sizes and good lattice structure // Mathematics of Computation. 68 (225), 1999. – Pp. 249–260.

10. Dyadkin I., Hamilton K. A study of 128–bit multipliers for congruential pseudorandom number generators // REF. IN COMP. PHYS. COMMUN. 125, 2000.

11. Dyadkin I., Hamilton K. A study of 64–bit multipliers for pseudorandom number generators // Computer Physics Communications. 103, 1997. – Pp. 103–130

12. Sezgin, F. A random number generator for 16–bit microcomputers // Computers and Operations Research. Vol. 23, No. 2, 1996. – Pp. 193–198.

13. Borosh I., Niederreiter H. Optimal multipliers for pseudo–random number generation by the linear congruential method // BIT 23, 1983. – Pp. 65–74.

14. Fishman G., Moore L. An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31}$ // SIAM Journal on Scientic and Statistical Computing 7, no. 1, 1986 – Pp. 24–45.

15. Fishman G. Multiplicative congruential random number generators with modulus $2^{\beta}$. An exhaustive analysis for $2^{32}$ and a partial analysis for $2^{48}$ // Mathematics of Computation 54, no. 189, 1990. – Pp. 331–344.

16. L'Ecuyer P., Couture R. An implementation of the lattice and spectral tests for multiple recursive linear random number generators // INF ORMS Journal on Computing 9, no. 2, 1997. – Pp. 206–217.

17. Бараш Л.Ю., Щур Л.Н. Генерация случайных чисел и параллельных потоков случайных чисел для расчетов Монте–Карло // Моделирование и анализ информационных систем. 2012; 19(2). – С. 145–162.

18. Matsumoto M., Nishimura T. Dynamic Creation of Pseudorandom Number Generators // Monte Carlo and Quasi–Monte Carlo Methods, Springer, 2000. – Pp 56–69.

19. Измерение времени работы фрагментов программ: метод. указания / сост. К.Е. Климентьев. – Самара: Изд–во Самар. ун–та, 2018.

С.С. Козунова

## УПРАВЛЕНИЕ РИСКАМИ БЕЗОПАСНОСТИ В ПРОМЫШЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМАХ

(АО «ФНПЦ «Титан-Баррикады)

За последнее десятилетие текущая ситуация в отечественной промышленности позволяет отметить, что в Российской Федерации рынок производственно-технологического оборудования и промышленных информационных систем успешно сформировался. В промышленной отрасли и различных промышленных видах деятельности образовываются и активно развиваются новые более крупные министерства, корпорации, федеральные органы исполнительной власти, объединённые заводы и консорциумы: Государственная корпорация по космической деятельности «Роскосмос», Министерство промышленности и торговли Российской Федерации, Государственная корпорация «Ростех», АО «Корпорация «СПУ – ЦКБ ТМ», ПАО «Корпорация «Иркут», и другие,