НАУЧНОЕ ПРИБОРОСТРОЕНИЕ, 1998 том 8, № 1-2 с. 56-59

ОРИГИНАЛЬНЫЕ СТАТЬИ

УДК 519.216

ОШИБКИ ОКРУГЛЕНИЯ ПРИ МОДЕЛИРОВАНИИ СЛУЧАЙНЫХ ВЕЛИЧИН

© А.С.Бердников, С.Б.Туртиа, А.М.Ларионов

Институт аналитического приборостроения РАН, Санкт-Петербург.

Поступила в редакцию 16 июня 1998г.

Описывается ошибка равномерности распределения, присущая большинству используемых в настоящее время датчиков случайных чисел (ДСЧ), анализируются причины, к ней приводящие, и даются рецепты исправления указанного недостатка.

ВВЕДЕНИЕ

Случайные последовательности с заданным законом распределения, будь то целочисленные последовательности $r_i \in [0,M)$ или эквивалентные им вещественные $x_i \in [0,1)$, столь широко используются при моделировании самых различных физических процессов, что нет необходимости доказывать важность идеального совпадения теоретического и реально получаемого законов распределения (для определенности остановимся на равномерном законе). Несмотря на то, что ДСЧ, моделирующие такие последовательности, имеют достаточно давнюю историю [1, 2, 3, 4], даже самые современные [5, 6] оказываются не свободны от ошибки, возникающей при преобразовании целочисленной арифметики, используемой для моделирования последовательностей с экстремально длинным периодом, в арифметику с плавающей точкой, которая используется в компьютерном моделировании физических процессов. Таким образом, даже когда сам алгоритм ДСЧ великолепен, его программная реализация может свести на нет все его достоинства. И речь идет не о каких-либо самодеятельных разработках, а об алгоритмах, включенных, например, в библиотеку MICROSOFT POWER STATION FORTRAN и многих других [7].

Этих ошибок, однако, легко можно избежать, что и будет показано далее.

ПОСТАНОВКА ПРОБЛЕМЫ

Большинство алгоритмов, как уже было сказано, используют для моделирования случайных последовательностей целочисленную арифметику, так как в ней намного проще показать, что данный ДСЧ действительно моделирует последовательности со сверхдлинным периодом и ничтожной корреляцией [1, 2, 3, 8]. Пусть, таким образом, имеется последовательность целых чисел в интервале $[0,M-1]$ где M — максимально представимое для данного компьютера целое число (к примеру, для 4-х байтного компьютера со знаковым битом это число будет равно $M=2^{31}$). После деления на M мы получаем псевдослучайное *число с плавающей точкой*, равномерно-распределенное в интервале [0,1), причем правая граница 1 исключена. Конечно, распределение не непрерывно, а дискретно с дискретностью $1/M$, но поскольку значение M очень велико, с достаточной степенью точности можно говорить о непрерывном распределении.

На рисунках Рис. 1 (а) и Рис. 1 (б) представлены, соответственно, идеальная и реальная совокупная функция распределения. Однако данная иллюстрация в большинстве ДСЧ верна лишь в теории, на практике же, при преобразовании 4-х байтного целого числа (использующего 31 бит) в 4-х байтное вещественное (использующее 24 числовых бита) процедура *округления* часто вносит изменения в функцию распределения.

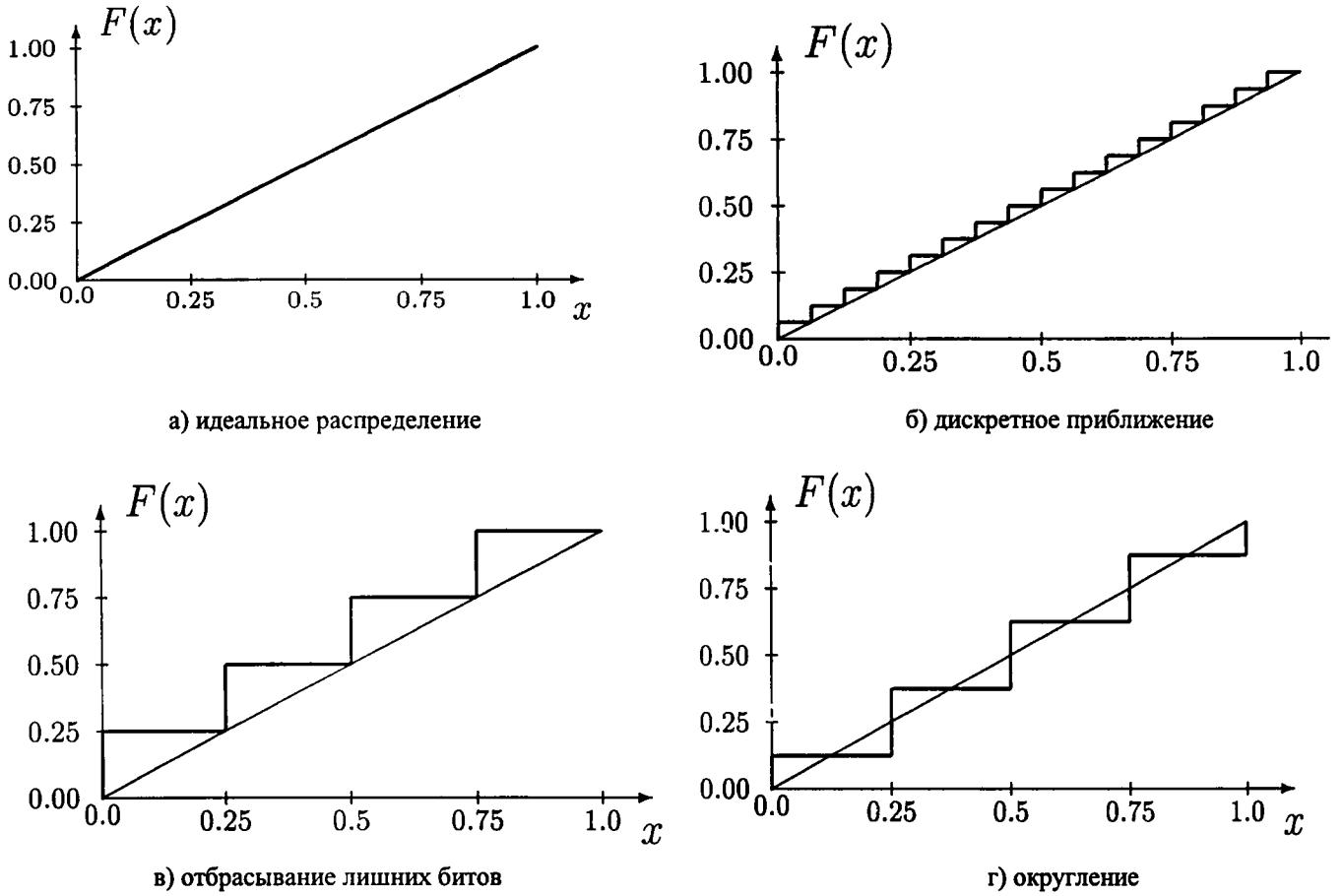
Рассмотрим простейший пример. Пусть некий алгоритм моделирует 4-х битные целые числа. После *идеального* деления на 2^4 получаем следующее равномерное распределение в интервале [0,1):

0.0000_2	0.0001_2	0.0010_2	0.0011_2
0.0100_2	0.0101_2	0.0110_2	0.0111_2
0.1000_2	0.1001_2	0.1010_2	0.1011_2
0.1100_2	0.1101_2	0.1110_2	0.1111_2

Далее преобразуем эти 4-х битные вещественные в 2-х битные. Если преобразование осуществляется путем отбрасывания ненужных битов, получается следующая таблица, соответствующая действительно равномерному закону распределения:

0.00_2	0.00_2	0.00_2	0.00_2
0.01_2	0.01_2	0.01_2	0.01_2
0.10_2	0.10_2	0.10_2	0.10_2
0.11_2	0.11_2	0.11_2	0.11_2

Однако, если вместо этого применяется процедура *округления* (используемая большинством известных в настоящее время ДСЧ), результат будет иной:



а) идеальное распределение

б) дискретное приближение

в) отбрасывание лишних битов

г) округление

Рис. 1. Совокупная функция распределения для различных способов преобразования целого случайного равномерно-распределенного числа в вещественное число с плавающей точкой.

0.00_2	0.00_2	0.01_2	0.01_2
0.01_2	0.01_2	0.10_2	0.10_2
0.10_2	0.10_2	0.11_2	0.11_2
0.11_2	0.11_2	1.00_2	1.00_2

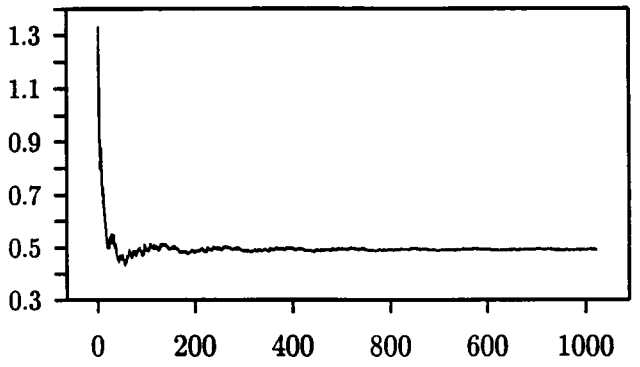
Здесь значение 0.0 встречается в два раза реже (что, по большому счету, не является особенным криминалом), однако "свободные" места заняты паразитными значениями 1.0, что абсолютно недопустимо. Функции распределения в этих двух случаях представлены на Рис. 1 (в,г).

2. ПРОВЕРОЧНЫЙ ТЕСТ

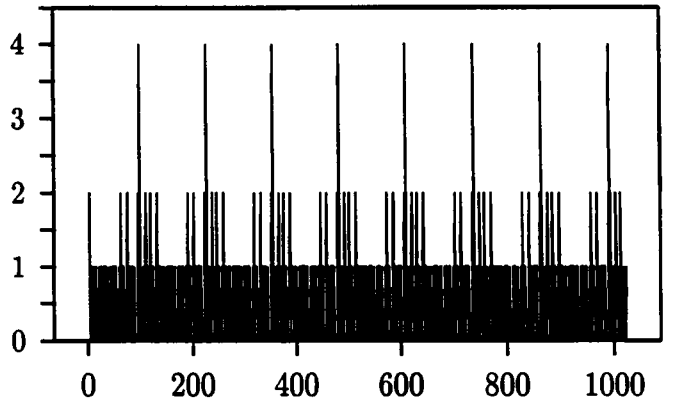
Для проверки данного предположения была выбрана программа генерации случайных чисел **RANDOM**, включенная в библиотеку программ *Microsoft Power Station Fortran*. Использовалось 24-х битное представление дробной части числа с плавающей точкой. В соответствии с нашими предположениями, хотя бы один раз из 2^{24} вызовов величина очень близкая к 1.0 будет смоделирована целочисленной арифметикой и в 50% случаев эта величина будет округлена до значения 1.0 в арифметике с плавающей точкой.

Следующая программа подсчитывает относительную частоту появления значения 1.0 в результате работы функции **RANDOM** после каждой серии из 2^{24} вызовов и, умножив эту величину на 2^{24}, выводит на печать.

```
      INTEGER*2  iseed
      REAL*4   rand
      INTEGER*4  I100, J100, k, kp, N, NN
C
      iseed=0
      CALL SEED(iseed)
C
      NN=1024
      N=2**24
      I100=0
      open(12,file='OUT.DAT')
      do 90 kp=1,NN
      J100=I100
      do 85 k=1,N
         call RANDOM(rand)
         if (rand.ge.1.0)  I100=I100+1
 85   continue
      write(*,*) I100
      write(12,*) kp, I100/float(kp),
```



а) Относительная частота появления значения 1.0 для 2^{24} вызовов функции **RANDOM**



б) Количество появлений 1.0 в нескольких сериях из 2^{24} вызовов каждая

Рис. 2. Результаты проверки ДСЧ **RANDOM** из библиотеки *Microsoft Fortran Power Station.* Значение **I100** равно 504.

```
*              I100-J100
90 continue
   close(12)
   stop
   end
```

На Рис. 2 (а) изображена величина **I100/KP** в зависимости от **KP** — числа внешних циклов теста. На Рис. 2 (б) показано возрастание значения **I100** для последовательности серий 2^{24} выборок случайных чисел.

Как и предполагалось, счетчик **I100** переполняется приблизительно один раз за 2^{25} вызовов функции **RANDOM** (асимптота на Рис. 2 (а) близка к величине 0.5). Рисунок 2 (б) демонстрирует периодичность ДСЧ в предположении, что период равен $128 \cdot 2^{24} = 2^{31}$. Строгая проверка показывает, что реальный период равен $2^{31}-2$, и значение 1.0 встречается 63 раза в течение полного периода. В результате асимптота для Рис. 2 (а) равна 0.492.

ЗАКЛЮЧЕНИЕ

Упоминая о библиотеке программ Microsoft Fortran, следует, однако, заметить, что при использовании библиотеки встроенных функций для компилятора *Microsoft Fortran 5.10*, использующем 15-ти битовые целые числа, этот эффект отсутствует: значение 1.0 никогда не встречается. В целом же, как было показано, процедура округления искажает функцию распределения случайного числа с плавающей точкой, время от времени выдавая значение 1.0. Наша коллекция ДСЧ [7,9] имеет тот же дефект. Только такие детально проработанные программные реализации ДСЧ как **ACARRY** [2] с самого начала имеющие дело с 24-х битными мантиссами, свободны от этого недостатка (правда имеют множество других, что показано в [10]). Практически все программные реализации ДСЧ, использующие конверсию целых чисел в вещественные, болеют указанной болезнью, хотя вылечиться от нее, модифицируя ключевой код, несложно. К примеру, следующая программа является модификацией хорошо известной функции **URAND** [11] и позволяет избежать влияния округления на качество распределения случайного вещественного числа:

```
function URAND2(IY)
      real     URAND2
      integer  IY
C
      integer IA, IC, M2, MM
      real  S
      save  IA, IC, M2, MM, S
C
      integer M, KM, KS
      real  SS
C
      double precision HALFM
      double precision DATAN, DSQRT
C
      data M2/0/
C
      if (M2.ne.0) goto 20
C
C --- при первом вызове
C --- длина машинного слова и проч.
C
    M=1
    KM=0
 10 continue
    M2=M
    M=2*M2
    KM=KM+1
    if (M.gt.M2) goto 10
C
    S=1.0
    KS=0
 15 continue
    KS=KS+1
```

```
      S=0.5*S
      SS=1.0+S
      if (SS.gt.1.0) goto 15
C
      HALFM=M2
C
C --- вычисление констант ДСЧ
C
      IA=8*idint(HALFM*datan(1.0D0)/8.0D0)
     *   + 5
      IC=2*idint
     * (HALFM*(0.5D0-dsqrt(3.0D0)/6.0D0))
     *   + 1
C
C --- вычисление float-множителя
C
      MM=1
      if (KM.gt.KS) MM=2**(KM-KS)
      S=0.5D0*MM/HALFM
C
C --- вычисление след. случайного числа
C
 20   continue
C
C --- основной алгоритм
C
      IY=IY*IA+IC
      if (IY.lt.0) IY=(IY+M2)+M2
C
C --- вывод: удаление последних битов
C --- в целом числе, чтобы избежать
C --- эффекта округления
C
      URAND2=float(IY/MM)*S
      return
      end
```

Следует заметить, что искажение функции распределения, вызванное округлением, конечно не носит особо опасный характер, поскольку встречается на уровне дискретности компьютерного представления вещественного числа. Факт появления значения 1.0 может вносить серьезную неразбериху в случае дискретного моделирования, так как в этом случае он может привести к выходу за границу массивов, тем самым существенно изменить поведение программы. Поэтому, полезным было бы проводить предварительную проверку, используя промежуточный код, подобный следующему:

```
      subroutine MYRAND(rand)
      real*4 rand
      real*4 res
C
 10   continue
      call RANDOM(res)
      if (res.ge.1.0) goto 10
      rand=res
      return
      end
```

Эта программа просто исключает значение 1.0, если оно появляется в результате моделирования.

ЛИТЕРАТУРА

1. *Кнут Д.* Искусство программирования, т.2, 1981. М.: Наука. 627 с.
2. *James F.* // Computer Physics Communications. 1990. №60 С.329–344.
3. *Marsaglia G.* // Computer Science and Statistics: The Interface, Ed. L.Billard. 1985. Amsterdam: Elsevier.
4. *L'Ecuyer P.* // Communications of the ACM. 1990. V.33, № 10. P.85–97.
5. *Marsaglia G. and. Zaman A.* // Ann.Appl.Probab. 1991. V.1 P.462–480.
6. *Niederreiter H.* // SIAM CBMS-NFS Regional Conference Series in Applied Mathematics. 1992. V. 63. Philadelphia: SIAM P. 32–37
7. MultiRand collection of Random Number Generators at: http://wwwdo.tn.tudelft.nl/bbs/software/mrand.zip
8. *L'Ecuyer P.* // Mathematics of Computation. 1996. V.65, №213. P.203–213.
9. *Berdnikov A.S., Turtia S.B., Compagner A.*// Proc. 4th Int. Conf. Physics Computing'92. Eds. R.A.de Groot, J.Nadrhal. 1993. World Scientific. P.264–265.
10. *Tesuka S., L'Ecuyer P., Couture R.* // ACM Trans.Model. Comput. Simul. 1993. V.3 P.315–331.
11. *Forsythe G.E., Malcolm M.A., Moler C.V.* Computer Methods for Mathematical Computations. 1977. Englewood Cliffs: Prentice-Hall Inc. 134 p.

ROUNDING ERRORS AT SIMULATION OF RANDOM NUMBERS

A.S.Berdnikov, S.B.Turtia, A.M.Larionov

Institute for Analytical Instrumentation RAS, Saint-Petersburg

The rounding error of uniformity of distribution in Random Number Generators is described. Its causes are analysed and recipes to avoid them are given.