Physics 115/242
Randu: a bad random number generator

Peter Young
(Dated: April 24, 2013)

We are not going to do into the details of the various techniques used to generate pseudo-random number generators but here we will briefly describe one routine, which turned out to be very poor, to illustrate the pitfalls that can one can fall into.

Many random number generators are what is called *linear congruential generators* which generate a sequence of integers, $I_1, I_2, I_3, \cdots$, each between 0 and $m - 1$ by the recurrence relation

$$I_{j+1} = aI_j + c \quad (\text{mod } m). \tag{1}$$

Here m is called the *modulus* and a and c are positive integers. Usually the I_j are converted to a real number between 0 and 1 as I_j/m. The routine is usually called as a function, say `ran(`*iseed*`)` where *iseed* is set as an initial "seed". On entry, the routine sets I_j to equal *iseed* and, on exit, the value of *iseed* is I_{j+1}, so *iseed* can be used on the next call to generate I_{j+2} etc.

IBM mainframes had such a random number generator, called `randu`, with $c = 0, a = 65539$ and $m = 2^{31}$. (Remember that on 32 bit machines like the mainframes, the largest positive integer is $2^{31}-1$.) Here is a few simple lines of Fortran code that implement it and which can easily be converted to C:

```
function randu(iseed)
    parameter (IMAX = 2147483647, XMAX_INV = 1./IMAX)
    iseed = iseed * 65539
    if (iseed < 0) iseed = iseed + IMAX + 1
    randu = iseed * XMAX_INV
end
```

The "mod m" operation is taken care of by adding 2^{31} if the left hand bit (the sign bit) is 1. Note that 2^{31} does not exist on 32 bit machines so we have to add $2^{31} - 1$ $(= IMAX)$ and then add 1.

If one generates a table of number and computes moments of the sample, the results look quite good with $\langle x^k \rangle \approx 1/(k + 1)$. However, this is *not sufficient*; the numbers must be **free of correlations** with each other and *randu fails badly* in this regard.

Let us generate triplets of numbers x, y, z and plot them in three-dimensional space. Ideally they should fill a cube of unit side uniformly. However, it turns out that *all* the triplets of random numbers generated by randu lie on only 15 planes. In fact the combination $9x - 6y + z$ is an *integer*(!). To see this, note that $65539 = 2^{16} + 3$ and consider the corresponding integers i_x, i_y and i_z (before dividing by `XMAX_INV` to get a float). We have

$$\begin{aligned} i_y &= (2^{16} + 3)i_x\,, \\ i_z &= (2^{16} + 3)i_y = (2^{16} + 3)^2 i_x = (2^{32} + 6 \times 2^{16} + 9)i_x \\ &= (2^{32} + 6 \times (2^{16} + 3) - 9)i_x = 6i_y - 9i_x \ (\text{mod } 2^{31})\,, \end{aligned} \tag{2}$$

since 2^{32} (mod 2^{31}) = 0. Hence $9i_x - 6i_y + i_z$ is a multiple of 2^{31} and so, dividing by 2^{31} to get a float, we find that $9x - 6y + z$ is an integer. In fact, this integer is restricted to values between -5 and 9. I demonstrate this in Figure 1, in which there are 1000 triplets of numbers. Here are the first three triplets starting with the *iseed* equal to 314159:

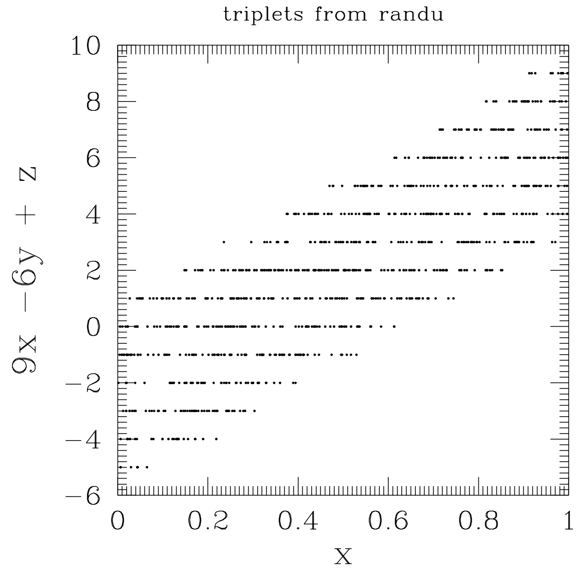

FIG. 1: Plot showing that all triplets of points generated by randu lie on one of the 15 planes $9x - 6y + z = m$ where $m = -5, -4, \cdots, 8, 9$.

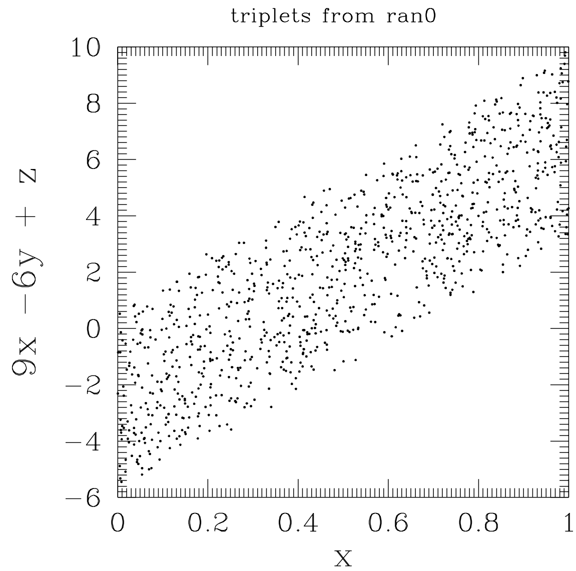

FIG. 2: Similar to Fig. 1 but for a "respectable" random number generator, ran0 in numerical recipes. The points now fill space in a random looking manner.

x	y	z	9x-6y+z
0.58781	0.52555	0.86299	3.00000
0.44801	0.92114	0.49477	-1.00000
0.67837	0.61729	0.59842	3.00000

If one uses a decent random number generator, however, the points fill space in a random-looking manner, as shown in the Figure 2, which used ran0 from numerical recipes. One can also get reasonable results by taking the randu generator but replacing 65539 by 16807.

How could you deduce that there is a problem with triplet correlations if you didn't know to look for this particular linear combination with factors $9, -6$ and 1? You could create a three dimensional array with $N = L^3$ elements (representing a discrete lattice of points in three-dimensions), generate triplets of random integers each between 0 and L using randu, and, for each triplet, "mark" the corresponding element of the array in some way, e.g. by replacing the initial value of 0 by 1. Then ask what fraction of points have been "marked" as a function of the number of triplets generated, which we will call M. The fraction marked should be, on average, $1 - e^{-M/N}$, if the numbers are uncorrelated. However, using randu, you will find that a finite fraction of the points are *never* marked.

We can use Mathematica to produce a three dimensional plot of triplets of points from randu:

Randu: a three dimensional plot

This *Mathematica* notebook produces a three−dimensional plot of the triplets of points from the randu random number generator. It is based on the code in Kinzel and Reents. One clearly sees that all the points lie on 15 planes. There are ‘‘npoint’’ points.

```
a = 65539; m = 2^31; i0 = 314159 ; npoint = 3000; nrand = 3 * npoint;

randu[i_] := Mod[a i, m]

uniform = Drop[ NestList[ randu, i0, nrand] , 1] / N[m];

triple = Table[Take[uniform, {n, n + 2}] , {n, 1, nrand - 2, 3}];

Show[ Graphics3D[ Map[ Point, triple]], ViewPoint → {10, 10, -23}];
```