Peter Alfeld, --- Department of Mathematics, --- College of Science --- University of Utah

Random Number Generators

Computer generated "Random Numbers" are used in many applications. Indeed, there is a whole set of numerical "Monte Carlo" techniques based on them. This page describes how (most) random number generators work and most importantly it lets you design and test your own random number generator. To do so just click on the applet nearby.

What is a random number generator?

Most random number generators generate a sequence of integers by the following recurrence:

$$x_0 = \text{given}, \qquad x_{n+1} = P_1 x_n + P_2 \quad (\text{mod } N) \quad n = 0,1,2,\ldots \quad (*)$$

The notation **mod N** means that the expression on the right of the equation is divided by N, and then replaced with the remainder.

To understand the mechanics consider the following simple Example. (Choose **Example** on the applet to study this example further.)

$$x_0 = 79,\ N = 100,\ P_1 = 263, \text{ and } P_2 = 71$$

Then

$$x_1 = 79*263 + 71 \ (\text{mod } 100) = 20848 \ (\text{mod } 100) = 48,$$

$$x_1 = 48*263 + 71 \ (\text{mod } 100) = 12695 \ (\text{mod } 100) = 95,$$

$$x_1 = 95*263 + 71 \ (\text{mod } 100) = 25056 \ (\text{mod } 100) = 56,$$

$$x_1 = 56*263 + 71 \ (\text{mod } 100) = 14799 \ (\text{mod } 100) = 99,$$

Subsequent numbers are: **8, 75, 96, 68, 36, 39, 28, 35, 76, 59, 88, 15, 16, 79, 48.** The sequence then **repeats** . (This indicates a weakness of our example generator: If the random numbers are between 0 and 99 then one would like **every** number between 0 and 99 to be a **possible** member of the sequence.

The parameters P_1, P_2, and N determine the characteristics of the random number generator, and the choice of x_0 (the **seed**) determines the particular sequence of random numbers that is generated. If the generator is run with the same values of the parameters, and the same seed, it will generate a sequence that's identical to the previous one. In that sense the numbers generated certainly are not random. They are therefore sometimes referred to as **pseudo random numbers.**

Transformation of the original sequence.

Of course one may want random numbers not as integers in a given range, but for example as uniformly distributed real numbers in a certain interval, or perhaps as real numbers of (almost) arbitrary size, but clustered around the origin. Distributions of that sort can be obtained by suitably **transforming** the original random numbers. For example, to transform a sequence defined as above into an evenly distributed set of real numbers in the interval from 0 to 1 simply divide each of the original numbers by N. In the remainder of this page, though, we just consider the sequence defined by **(*)** itself.

What makes a good random number generator?

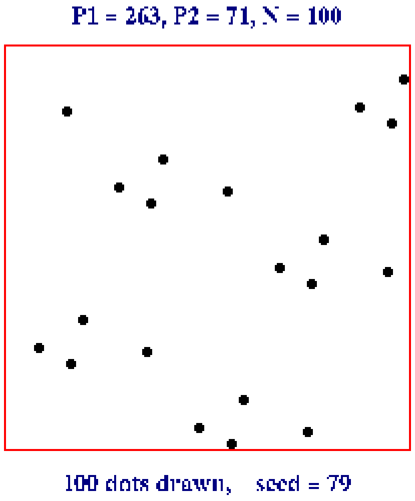
That's a good question! Several answers are possible, for example:

- The sequence generated by (*) isn't random at all, so there is no good random number generator of that form.
- A sequence (*) is good if it passes several well established statistical tests.
- Or, it's good if it gives good results in particular applications (where of course the meaning of "good results" is heavily dependent upon the context).

The applet on this page takes a different tag. It plots a certain number of points

$$(\mathbf{x}_{\mathbf{i}}, \mathbf{x}_{\mathbf{i+k}})$$

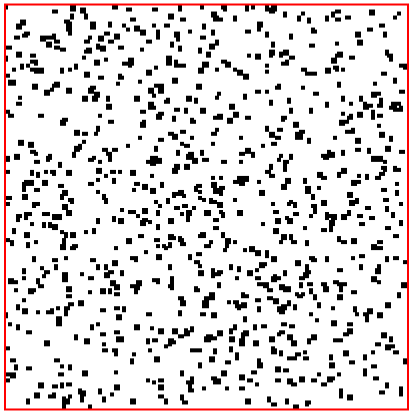
for certain values of k = 1,2,3,... . Intuitively, for a random sequence, one should obtain a set of points distributed "evenly" , "randomly" or "uniformly " over a square. It is not easy to make these concepts precise, but it is sometimes glaringly apparent when a set of points is **not** distributed in this way. Plotting 100 points with **k=1** for our example generator above generates the picture nearby. (It's is shown here at half its original size.) In the figure the first coordinate measures horizontal distance from the left margin of the red box, and the second coordinate measures vertical distance downwards from the upper margin of the red box. Thus some of the points in this box have coordinates (79,48), (48,95), (95,56), etc. The values of the coordinates are scaled to fill the entire red box (which in this case measure 200 by 200 pixels). It's clear that there are only 20, points, and, since 100 were drawn, five lie on top of each other for each of the black dots. Moreover, the dots appear to lie along six slanted lines. As pointed out above, for a " good" random number generator there should be 100 points, and the distribution should be " random".



The second figure nearby shows a distribution of 1,000 points obtained with a widely used and well tested and analyzed random number generator using

$P_1 = 16807$, $P_2 = 0$, and $N = 2^{31} - 1 = 2147483647$.

P1 = 16807, P2 = 0, N = 2147483647

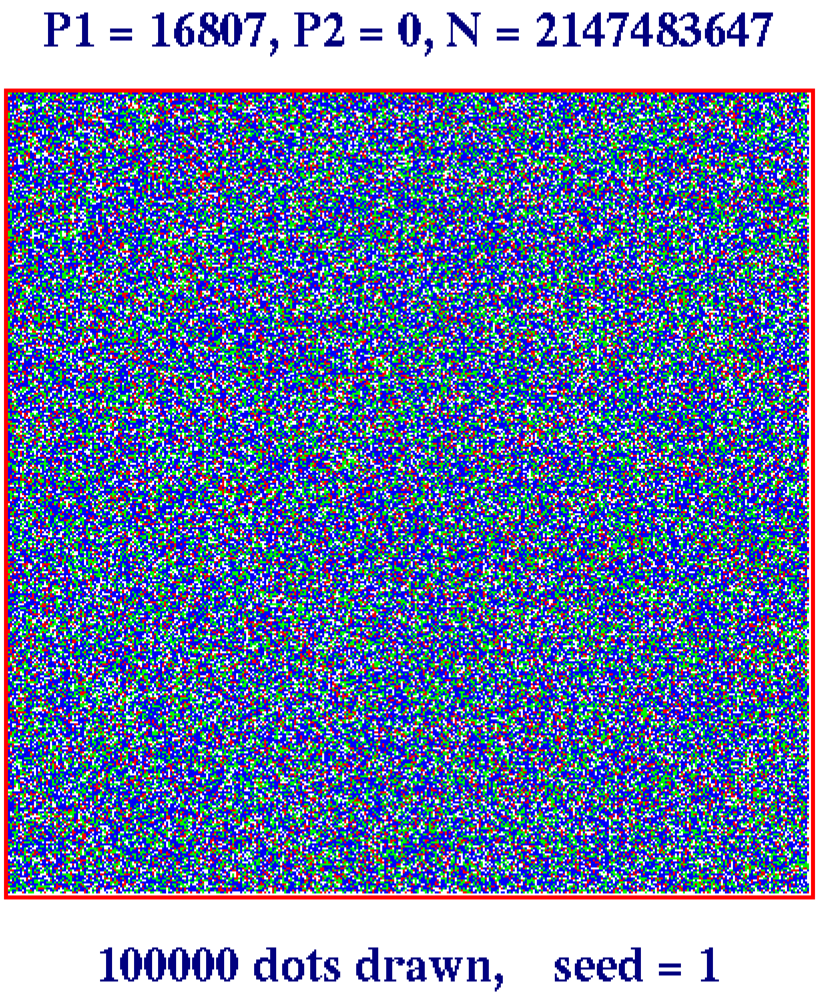


1000 dots drawn, seed = 1

This generator is described in the reference by Park and Miller given below.

One reason for the seemingly peculiar choice of N is that that particular number is the largest integer than can be represented on a Unix machine or in Java.

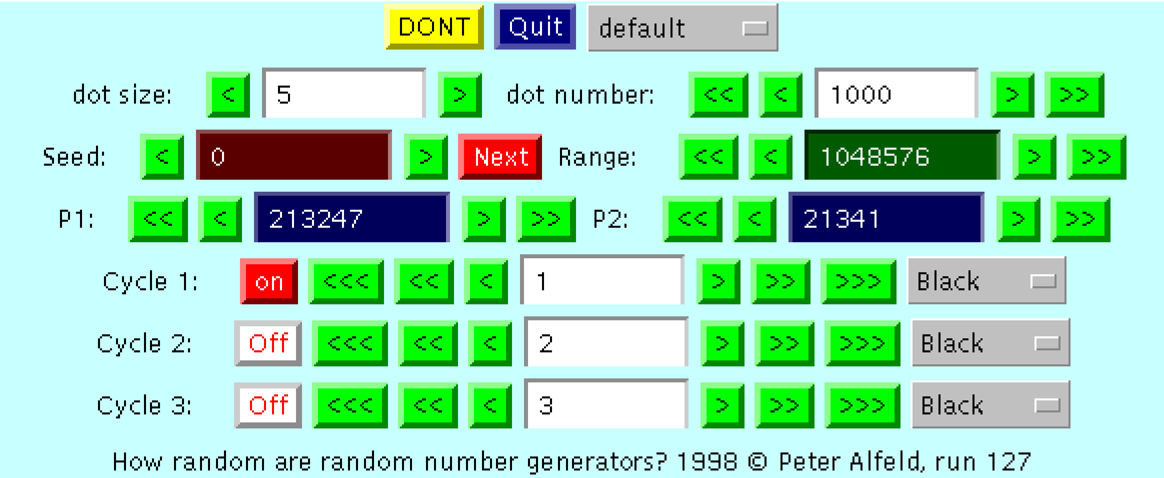
To illustrate the abilities of this applet consider the following Figure which shows three sequences of 100,000 points each, using the same generator, for k=1 (red), k2 (green), and k=3 (blue). Reassuringly, no systematic patterns are readily apparent.



Operating the Random Applet

Clicking on the box at the beginning of this page will cause a control window to pop up that looks much as illustrated in the image nearby. A drawing window that will contain the plots also appears.

There are mostly text input windows and buttons marked ">", "<<" etc.. To change an entry of a text window highlight the text and write a new number. You can also change parts of the number. It is important to terminate every change by pressing the **Return** or **Enter** key. The green buttons increment or decrement the text windows they surround. "< " and ">" decrement or increment by 1, the other buttons by larger amounts, e.g., by going



dot size: 5 dot number: 1000

Seed: 0 Range: 1048576

P1: 213247 P2: 21341

Cycle 1: on 1 Black

Cycle 2: Off 2 Black

Cycle 3: Off 3 Black

How random are random number generators? 1998 © Peter Alfeld, run 127

to the closest prime number or the closest power of 2. Points of the form

$$(x_i, x_{i+k})$$

can be drawn for up to three different values of k, as determined in the three lines labeled Cycle 1, 2, and 3.

Following is a description of some less obvious features:

- **DONT/DRAW**: Normally the picture in the drawing window gets redrawn every time a change is made in the control window. You can suppress the drawing (e.g., if you want to make many changes) by pressing the yellow button in the top row. Its label "DONT" indicates whether drawing currently takes place. Click on "DONT" to suppress it or on "DRAW " to activate it.
- **Quit**. As you might expect, you can dispose of the control wind and the drawing window by clicking on the Quit button. You can obtain the same effect by clicking on the applet at the top of this page, or by typing "q", "Q", "x", or "X" in either the control window or the drawing window
- The **Menu** in the top row lets you set the applet to analyze specific random number generators. More may be added in the future, but right now four are available:
  - The default generator that does not work very well and lets you discover strange patterns/
  - The high quality generator proposed by Park and Miller in the reference quoted above.
  - A random number generator I use in my **Microscope** Software. Interestingly that generator does quite well in the tests performed by this applet, but it shows non-random effects in the Microscope package. (The reason I have not removed it is that doing so would effect all the examples in the manual.)
  - The **Example** generator used for illustration at the beginning of this web page.
  - The 16 bit version of the infamous **RANDU** discussed at length in Park and Miller (referenced below).
  - A random number generator published in 1978 by P. Grogono. Display it with **k=14** or **k=38**.

Downloading the software and using it standalone

You can download the Java Binary and use it standalone (e.g., by typing **java Random** on your Unix system). You'll need three class files that you can obtain by clicking on the links below. They point to binary files and the pages will look strange in your browser, but you should be able to download them nonetheless. Let me know if you have any trouble.

- Random.Class
- DrawRandom.Class
- ControlRandom.Class

Links and Literature

A standard paper is S.K. Park and K.W. Miller, " Random Number Generators: Good Ones Are Hard To Find", Communications of the ACM, October 1988, pp. 1192-1201.

A recent article on random number generators is by Pei-Chi Wu: Multiplicative, Congruential Random-Number Generators with Multiplier +/-2 ^{k1} +/-2 ^{k2} and Modulus 2 ^{p} -1, ACM Transactions on Mathematical Software, June 1997, v. 23, No. 2, pp. 255-265.

The Grogono generator is described in P. Grogono, Programming in Pascal, 2nd Ed., Addison-Wesley, 1984, pp. 135-137.

I gratefully acknowledge a very helpful discussion I had with Nelson Beebe.

[15-Aug-2011]

Go to Peter Alfeld's Home Page.