Generating Pseudo-random Numbers

Linear Congruential Pseudo-random Number Generators

Consider the function

$$g(x) = (Cx + D) \text{ mod } M$$

where C, D and M are constants.

Starting with an initial value x_0, we generate a sequence of numbers, $x_0,\ x_1,\ x_2,\ x_3, \ldots$ by letting

$$x_{n+1} = g(x_n)$$

EXAMPLE: Let $M = 8,\ C = 5,\ D = 7,\ x_0 = 4$. Then

$$g(x) = (5x + 7) \text{ mod } 8$$

Using this we obtain

$$\begin{aligned} x_1 &= [(5)(4) + 7] \text{ mod } 8 = 3 \\ x_2 &= [(5)(3) + 7] \text{ mod } 8 = 6 \\ x_3 &= [(5)(6) + 7] \text{ mod } 8 = 5 \end{aligned}$$

Continuing in this way we find $x_4 = 0,\ x_5 = 7,\ x_6 = 2,\ x_7 = 1,\ x_8 = 4$. At this point the sequence starts over again and repeats the same 8 values over and over.

One thing to note about this example is that each of the values in $\{0, \ldots, 7\}$ occurs before the sequence begins repeating. To guarantee this, the values of $M,\ C$ and D must be carefully chosen.

A number theory result guarantees that with the conditions listed below, all the numbers in $\{0, \ldots, (M-1)\}$ will occur before the sequence repeats.

(i) D and M are relatively prime

(ii) $C - 1$ is divisible by every prime factor of M

(iii) If M is divisible by 4 then so is $C - 1$

Since we would like a long sequence of random numbers we should choose a very large value for M. Also, we would like our number generator to produce values between 0 and 1 (not between 0 and $M - 1$), so we will return the values $x_1/M,\ x_2/M,\ x_3/M, \ldots$. We call such a number generator a Uniform(0,1) random number generator. We will see that all the random behavior we would like to represent in a computer program can be derived from a Uniform(0,1) random number generator.

The Pascal code below implements the method described above. Note that the variable `Seed` is global and must be initialized at the beginning of the program execution.

```
var Seed : double

function Random : double;

const M = 1048576.0;
      C = 889925.0;
      D = 489459.0;

begin
Seed := C * Seed + D;
Seed := Seed - trunc(Seed / M) * M;
Random := Seed / M;
end;
```

Equivalent code in `C` is displayed below. Note that the `fmod` function in `<math.h>` and that you will need to use the `-lm` directive when compiling your code to link the math library.

```
#define M 1048576.0
#define C 889925.0
#define D 489459.0

double Seed;

double Random (void)
{
  Seed = fmod(C * Seed + D, M);
  return (Seed / M);
}
```

For more information on random number generation see:

- Knuth, Donald, *The Art of Computer Programming.*
- *Numerical Recipes* available at most bookstores.