Random Numbers

Random numbers are ubiquitous in physics – thermodynamics, radioactivity, particle collision and everything in between

Two basic methods to generate random number with varying degree of randomness

- *True* RNG
- *Pseudo* RNG

True RNG : uses natural phenomenon like coin flipping, dice rolling, radioactive decay, thermal noise, atmospheric radio-noise etc.

`Requires post-processing, slow` $\Rightarrow$ `not useful for regular usage`

Pseudo RNG : based on algorithms, generated iteratively

`Deterministic, finite sequence length, correlated but extremely fast and portable`

Sequence length can be made veryyy long by proper choice of parameters

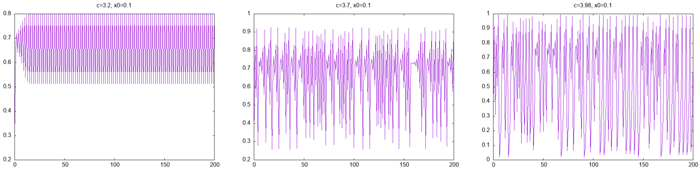
Basic goal : Write your own pRNG and use it for all assignments, exams and DIY

Example : a quick and dirty pRNG

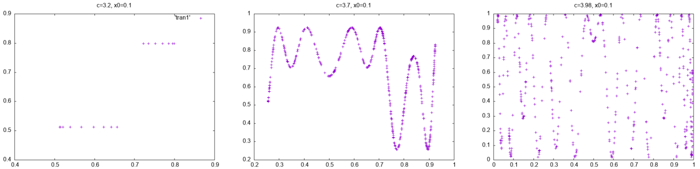
$$x_{i+1} = c\, x_i\, (1 - x_i)$$

x_0 is the seed which defines the random sequence.

An exercise with $x_0 = 0.1$ and $c = 3.2, 3.7, 3.98$



Plots of x_i vs. x_{i+5} show correlatedness (how to measure / quantify it?)



Bad pRNG for many choices of c and seed x_0 – settles down into regular pattern. However, no specific pattern for $c = 3.98$, $x_0 = 0.1$

Quantifying good and bad pRNG : Need mathematical tests for determining randomness $\Rightarrow$ if pRNG fails test, then don't use

Eyes are good at discerning patterns but can fool us too!

Basic test : correlation, moments
Advanced test : chi-square, Kolmogorov-Smirnov

Ideally random numbers generated have have no correlations and error statistical *i.e.* scale as $1/\sqrt{N}$

Correlations test :

$$\epsilon(n, N) = \frac{1}{N}\sum_{i=1}^{N} x_i\, x_{i+n} - \left(\frac{1}{N}\sum_{i=n}^{N} x_i\right)^2$$

Connected Correlations test :

$$\epsilon(n, N) = \frac{1}{N}\sum_{i=1}^{N} x_i\, x_{i+n} - \frac{1}{N}\sum_{i=1}^{N} x_i\, \frac{1}{N}\sum_{i=1}^{N} x_{i+n}$$

If tuplets of RN not correlated, $\epsilon(n, N) \to 0$ with statistical error $1/\sqrt{N}$.

Linear Congruential Generator

One of the oldest and most common choice of pRNG having a uniform distribution,

$$x_{i+1} = \Big(ax_i + c\Big) \bmod m \quad \equiv \quad x_{i+1} = \text{remainder}\left(\frac{ax_i + c}{m}\right)$$

The m determines the period of the generator *i.e.* produces random numbers between 0 and $m-1$, whereas x_i/m yields randoms in the interval [0,1].

- m is typically chosen to be 2^{32}
- a is *multiplier* and usually $0 < a < m$. Numerical Recipes uses $a = 1664525$ and gcc uses $a = 1103515245$
- c is *increment* and usually $0 < c < m$. Numerical Recipes uses $c = 1013904223$ and gcc uses $c = 12345$

Not all rosy with **LCG**

- $a,\ c,\ m,\ x_0 = 6, 7, 5, 2$: 4,1,2,0,2,4,1,2,0,2, ...
- $a,\ c,\ m,\ x_0 = 27, 11, 54, 2$: 11,38,11,38, ...

LCG : Hull-Dobell theorem

LCG is extremely sensitive to a, c, x_0, m. Particularly, a has to be chosen with great care else short / very short periodicity will set in.
Hull-Dobell theorem : LCG has a period m *iff* $c \neq 0$ and

1. c is coprime to m,
2. $a - 1$ is a multiple of p for every prime p dividing m
3. $a - 1$ is a multiple of 4, if m is a multiple of 4.

LCG works well for m having many repeated prime factors p, such as power of 2. But if m are square-free integer (having no n^2 factor for any n), then only $a = 1$ is allowed and it is a very bad pRNG.

LCG is extremely fast, least memory footprint but period severely limited by choice of m : for $m \sim 10^{32} \rightarrow 10^9$ pRN. Gets exhausted in seconds!!

$c = 0$ corresponds to Lehmer, Park-Miller pRNG

$$x_{i+1} = ax_i \cdot \mathrm{mod}\, m$$

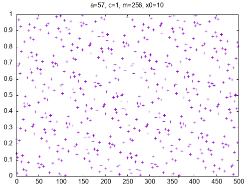
m can be a prime or a prime just less than a power of 2 (Mersenne primes $2^{31} - 1$, $2^{61} - 1$ etc.) or can be a simple power of 2.

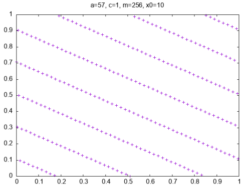
A few exercise in **LCG**

$a = 27,\ c = 11,\ m = 54,\ x_0 = 10$ (Ugly choice) :

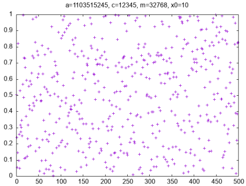
$0.204,\ 0.704,\ 0.204,\ 0.704,\ \ldots$

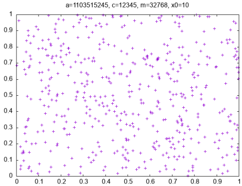
$a = 57,\ c = 1,\ m = 256,\ x_0 = 10$ (Bad choice)





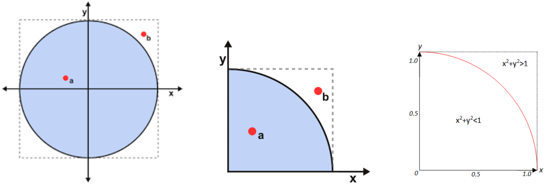
$a = 1103515245,\ c = 12345,\ m = 32768,\ x_0 = 10$ (Good choice)





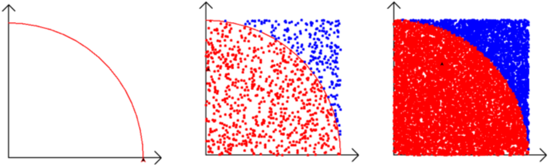
Application : Determination of π

Consider a unit circle $r = 1$ centered at origin $\Rightarrow$ area $= \pi$.
Put unit circle in a square of side $2r = 2$.
Confine to first quadrant and randomly generate points (x, y) and check for inside points $x^2 + y^2 \leq 1$. $\Rightarrow$ area $= \pi/4$.



Then over LARGE number of trials

$$\frac{\text{circle area}}{\text{square area}} \approx \frac{\text{total inside}}{\text{total trials}} \Rightarrow \pi \approx 4 \times \frac{\text{total inside}}{\text{total trials}}$$



n = 0 n = 1000 n = 10000

Different pRNG distribution

Standard **pRNG** generates uniform random integers $\ell \in [0, INT\_MAX]$ or floating point numbers $x = (\ell / INT\_MAX) \in [0, 1)$.

- **pRNG** uniformly distributed $u \in [a, b)$ from $x \in [0, 1)$

$$u = a + (b - a)\, x$$

Suppose $p(x)$ is **pdf** of a uniform RN x and target **pdf** is $q(y)$.

$$|q(y)\, dy| = |p(x)\, dx| \;\; \rightarrow \;\; q(y) = p(x) \left| \frac{dx}{dy} \right|$$

For uniform RN $x \in [0, 1) \;\Rightarrow\; p(x) = 1$.

- Exponentially distributed RN

$$q(y) = a\, e^{-ay} \;\; \text{for} \;\; y \geq 0,\; a > 0$$

From the transformation law

$$a\, e^{-ay} = \left| \frac{dx}{dy} \right| \;\rightarrow\; x = \int_0^y q(y)\, dy = 1 - e^{-ay} \;\Rightarrow y = -\frac{1}{a} \ln(1 - x) \equiv -\frac{1}{a} \ln x$$

because $(1 - x) \in [0, 1)$ as well.