Random number generators and random processes

Eugeniy E. Mikhailov

The College of William & Mary

Lecture 13

Notes

Statistics and probability intro

Mathematical and physical definition of probability (p) of event 'x'

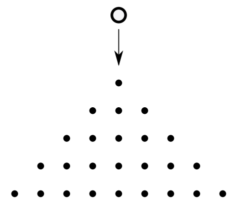
$$p_x = \lim_{N_{total}\to\infty} \frac{N_x}{N_{total}}$$

where

N_x	the number of registered event 'x'
N_{total}	the total number of all events

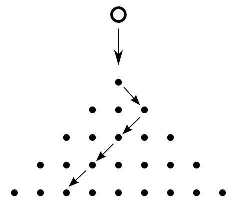
Notes

Peg board example



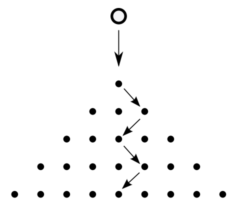
Notes

Peg board example



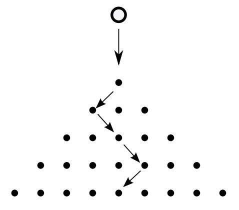
Notes

Peg board example



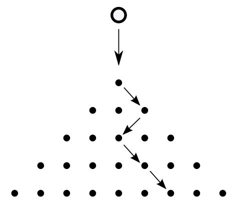
Notes

Peg board example



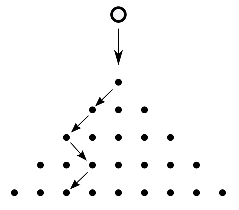
Notes

Peg board example



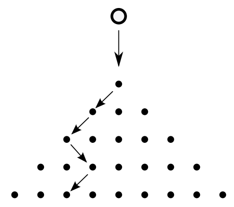
Notes

Peg board example



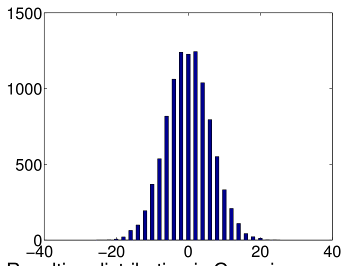
Notes

Peg board example



Example of 10^4 balls run over 40 layers of nails

```
x=pegboard(10^4, 40);
hist(x, [-25:25]);
```



Resulting distribution is Gaussian

Notes

Probability for occurrence of the real number

The interval [0..1] has infinite amount of real numbers. This is true for any other non zero interval.
Since we cannot run ∞ number of tests, there is very little or maybe zero chance that an event will repeat or even happen.
In this case, we should speak about probability density $p(x)$.
The best way to estimate it is to make a histogram.
Run N tests with numbers distributed between 0 and 1, split the interval of interest into m bins and calculate the number of events which occur in a given bin $h(x_b)$. Plot this vs. bin positions.

The probability density estimate

$$p(x) = \lim_{N,m\to\infty} \frac{h(x_b \text{ nearest to } x)}{N}$$

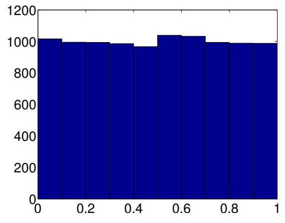
Easy to do with Matlab

```
r=rand(1,N); hist(r,m);
```

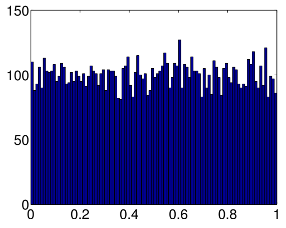
Notes

Uniform random distribution

```
r=rand(1,10000);
hist(r,10);
```



```
r=rand(1,10000);
hist(r,100);
```



Notes

Random number generators (RNG)

How can a computer, which is very accurate, precise, and deterministic, generate random numbers?

It cannot!

The best we can do is to generate a sequence of pseudo random numbers.
By pseudo we mean that starting from the same initial conditions the computer will generate exactly the same sequence of numbers (*very handy for debugging*). But otherwise, it will look like random numbers and will have statistical properties of random numbers.

Notes

Linear Congruential Generator (LCG)

Recursive formula

$$r_{i+1} = (ar_i + c) \mod m$$

here

m	the modulus
a	multiplier, $0 < a < m$
c	increment, $c \leq c < m$
r_1	seed value, $0 \leq r_1 < m$

All pseudo random generators have a period and this one is no exception. Note that once r_i repeats one of the previous values the sequence will restart.
This one can have at most a period of m distinct numbers ($0..m$).

Notes

Linear Congruential Generator (LCG) continued

A bad choice of a, c, m will lead to an even a shorter period.

Example

$m = 10$, $a = 2$, $c = 1$,$r_1 = 1$
$r = [1, 3, 7, 5, 1]$

The LCG is fast and simple, but it is a very bad random numbers generator.

Do not use the LCG whenever your money or reputation is at stake!

While Matlab does not use LCG, many other programming languages use it as default in their libraries, so be aware of it.

Notes

Random number generators period

Even the best pseudo random generators cannot have a period larger than 2^B, where B is number of memory bits. Can you prove it?

While the period can be huge its not infinite. For example, default MATLAB random number generator has the period $2^{19937} - 1$.

Why bother?
Recall that the Monte Carlo integration method error is $\sim 1/\sqrt{N}$.
This holds true only when $N <$ than the period of random number generator (T).
For $N > T$, the Monte Carlo method cannot give uncertainty better than $\sim 1/\sqrt{T}$.
Further increase of the number of random points will waste CPU cycles and will not bring any extra precision.

Notes

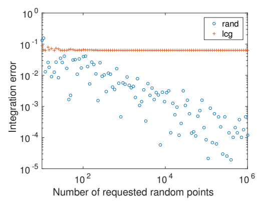
How to check the random generator

- Generally it is uneasy and probably impossible. NIST provides several guidelines and software for RNG tests.
- However for us only statistical properties are of importance.
- The easiest is to check that integral deviation calculated with the Monte Carlo algorithm drops as $1/\sqrt{N}$.

Notes

Simple random generator check

Compare the MC error with the MATLAB `rand` and the LCG



Circles correspond to the MC with `rand` and crosses to `lcgrand` seeded with coefficients $m = 10$, $a = 2$, $c = 1$,$r_1 = 1$

```
function r= ...
lcgrand(Nrows,Ncols, ...
a,c,m, seed)

r=zeros(Nrows, Ncols);
r(1)=seed;
cntr=1;
for i=2:Nrows*Ncols;
r(i)= mod( (a*r(i-1)+c), m);
end
r=r/(m-1); %normalization
end
```

```
check_lcgrand
```

Notes

Notes

Notes

Notes