

Structural Reliability
Prof. Baidurya Bhattacharya
Department of Civil Engineering
Indian Institute of Technology, Kharagpur

Lecture –71
Monte Carlo Simulations (Part - 03)

(Refer Slide Time: 00:27)

Monte Carlo simulations

Structural Reliability
Lecture 8
Monte
Carlo
simulations

True random number generators

True random number generators refer to the output of some unpredictable physical source. The output may need to be filtered in certain instances. The physical source can be:


- atmospheric noise
- radioactive decay
- thermal noise from an electronic circuit, etc.

- Some sequences generated from a “true” random number generator could pass all tests of randomness.
- A true random sequence can be stored but cannot be recreated.
- It is a slow (and more expensive) process compared to computer based **pseudo-random number generators**.

Further reading:
random.org

Massively parallel ultrafast random bit generation with a chip-scale laser, by K. Kim et al., in Science, Vol. 371, Issue 6532, pp. 948-952, 2021

©Baidurya Bhattacharya, IIT Kharagpur www.facweb.iitkgp.ac.in/~baidurya/



So it should be clear that in order for Monte Carlo simulations to give useful results one needs an abundant supply of random numbers and specifically IID uniforms standard uniforms taking values between 0 and 1. So, where does one obtain such a supply of random numbers. So, the true random number generators are outputs of some physical process or some device. And in some cases these outputs may need to be processed and filtered and then used.

These sources could be noise the sources could be radioactive decay for example you know we did discuss person processes briefly in our past lecture. And there is a very strong theoretical basis to suggest that the inter arrival times between these decay events are IID exponential random variables. So, if we could record these time instances we would have a sample of IID exponentials which then we can convert to IID uniforms which we will learn about this transformations later in this lecture and the next.

So we could use them for the purpose of Monte Carlo simulations. And in many cases the qualities of these true random numbers are very good they pass all important or all relevant tests of randomness. In fact these tests we are going to look later in this lecture. The problem is that a true random sequence can be stored for future use but it cannot be recreated and for our purpose in engineering we would need to be able to reproduce random sequence for verifications or for use by a peer or a colleague which is different from the needs of cryptography for example where you would not like this sequence to be recreated easily.

It is also in most cases true random numbers are slow and more expensive to generate compared to computer-based pseudo-random number generators. But there are notable exceptions just earlier this year there was this paper in science where the authors discussed a very fast generation of two random numbers of the order of 250 terabits per second using the chip scale laser that they described.


So, we should mention such advances. But for now it is the computer-based pseudo-random number generators that are almost exclusively used for science and engineering applications.

(Refer Slide Time: 04:04)

The slide is titled "Monte Carlo simulations" in red. It is part of a lecture series on "Structural Reliability", "Lecture 8", "Monte Carlo simulations". The main heading is "Pseudo random number generators".

- Sequence of (pseudo)-random numbers are computer-generated:
 - By a precise, deterministic reproducible and fast algorithm
 - Have all relevant appearance of a truly random sequence
- Properties of a good pseudo-random generator:
 - Accuracy
 - Long Period
 - High Speed
 - Short Setup time
 - Small Length of compiled code
 - Machine independence
 - Versatility in possible applications
 - Simplicity and readability

©Baidurya Bhattacharya IIT Kharagpur www.iitkgp.ac.in/~baidurya/ 216



So, what are these pseudo-random number generators? These are sequences of numbers that are generated by a very precise deterministic reproducible algorithm and fast very fast but even if

they are completely deterministic they have all the appearance all the required appearance for the application at hand of being a truly random sequence. So, that is very important. Now we would like such pseudo-random number generators to have a set of desirable properties.

Especially the ones that have highlighted in bold accuracy which means they must be IID uniforms they must have a very long period, period means the sequence length after which it starts to repeat itself. So, all the IID characteristic is lost and machine independence because we would like to test someone else's algorithm or someone else's results or have someone else test our results. So, the same sequence must be created regardless of the particular operating system or particular machine at hand.

(Refer Slide Time: 05:35)

Structural Reliability
Lecture 8
Monte Carlo
simulations

Monte Carlo simulations

Pseudo random number generators

Pseudo-random numbers are most commonly generated using "multiple recursive generators":

$$x_i = (a_1 x_{i-1} + a_2 x_{i-2} + \dots + a_k x_{i-k}) \bmod m$$

$$u_i = x_i / m$$

where m is the modulus and k is the order of the generator. The coefficients a_i are all less than m : $a_i \in Z_m = \{0, 1, 2, \dots, m-1\}$. The maximal period of this generator is $m^k - 1$. This is achieved iff the smallest positive integer n such that $z^n \pmod{P(z)} \bmod m = 1$ is $n = m^k - 1$. Here $P(z) = z^k - a_1 z^{k-1} - \dots - a_k$ is the characteristic polynomial of the recurrence.

A special case of the multiple recursive generator occurs with $k = 1$, yielding the linear congruential generator.

Take this idea to design RNGs based on recurrences modulo 2, i.e., directly in terms of bit strings and sequences:


$$\mathbf{x}_i = \begin{Bmatrix} x_{i,0} \\ \vdots \\ x_{i,k-1} \end{Bmatrix} = \mathbf{A}\mathbf{x}_{i-1}, \quad x_{i,j} \in \{0,1\}$$

$$\mathbf{y}_i = \begin{Bmatrix} y_{i,0} \\ \vdots \\ y_{i,m-1} \end{Bmatrix} = \mathbf{B}\mathbf{x}_i, \quad y_{i,j} \in \{0,1\}$$

$$u_i = \sum_{j=0}^{m-1} y_{i,j} 2^{-j}$$

Further reading:
Random Number Generation, by P. L'Ecuyer in Handbook of Computational Statistics. Eds Gentle et al. Springer, 2012.

Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator, by M. Matsumoto & T. Nishimura. ACM TOMACS 8, 3-30, 1998.



©Baidurya Bhattacharya #B Kharagpur www.facebook.com/baidurya/ 217

Now pseudorandoms are most commonly generated with the help of the multiple recursive generators which uses modular arithmetic m being the modulus and with each successive output of x_i it is normalized by m which produces the standard uniform deviate, k is the order of the generator and the maximum possible period of the generator depends on m and k . Now when k is equal to 1 we have the classical linear congruential generator which we are going to look at in the next slide in some detail.

But this idea of multiple recursive generator it can be taken to an RNG based on recurrence

modulo 2 which means directly in terms of bit strings and sequences uh. So, the operations are like this the first set is the twisting type operation. The second step is the tempering type operation and the third set of equation that you see it is basically the output of the random number generated, the standard uniform written out in terms of the y's which depends on the word size.

One of the most popular generators in this class is the Marsan twister algorithm which has very good properties in fact the default generator in Matlab which is what we are going to use in this course is the the MT19937 algorithm which has a huge period $2^{19937} - 1$ which is something like google to the power of 60. So, it is practically infinite. So, the issue of long period is solved and if you if you would like to read further I have listed two references here. In the next two slides we are going to look at the linear congruential generator and some examples of pseudorandom number generators.