

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

Lecture –185
Capacity Demand Component Reliability (Part 33)

(Refer Slide Time: 00:27)

Monte Carlo simulations

Structural Reliability
Lecture 23
Capacity demand
component reliability

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
 - i.e., an IID sample of $U(0,1)$ variates
- 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
 - **Repeatability**

```
"rand" command in Matlab for generating U(0,1) deviates  
Marsenne twister is the default generator  
The default seed is 0  
use rng(n) command to reset the seed to n  
use rng(shuffle) to reseed from the system clock
```

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

91



It is clear by now that Monte Carlo simulation requires an abundant supply of IID random numbers. We did discuss this the various methods of generating random numbers earlier in this course in part A and it was quite clear that true random number generators those that use physical sources are not really appropriate for our purpose in science and engineering and structural reliability that we are looking at now and we need pseudo-random number generators.

Generators that run on a computer they produce a sequence of IID uniform deviates standard uniform deviates as the output of a very precise and deterministic computer algorithm. But very importantly this sequence has all the appearance of a true random number sequence. So properties of a good random number generator are quite a few and the ones that we are particularly interested in would be of course accuracy the ones that are highlighted in bold a long period which we have discussed.

But it means machine independence and repeatability basically a code that you write and get some results if I use that on my computer I should be getting the same exact results and thereby we can verify each other's results. We have been talking about MATLAB in this course and rand as we have seen already is the command in MATLAB for generating standard uniform deviates the default generator is the Mersenne twister algorithm.

The default seed is zero if you want to change the seed to any other particular value if you want to start with a random seed so to say you can use rng command rng n sets the c to the number n that you want and if you want to take the seed from the system clock then use rng shuffle. As you see on the screen. So with these standard uniform deviates we need to now set up and undertake Monte Carlo simulations for all kinds of problems.

So obviously we cannot stay confined to uniform deviates we have to convert them to distributions of our choice or the ones that we need.

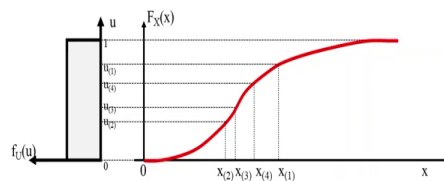
(Refer Slide Time: 03:33)

Monte Carlo simulations

Structural Reliability
Lecture 23
Capacity demand
component reliability

The inversion method

Theorem: If F is a continuous CDF, and U is a Uniform RV in $(0,1)$, then the RV X defined as $X = F^{-1}(U)$ is distributed according to F .



Drawbacks of the inversion method:

- Inversion method is the most general method. But it ideally requires a closed form expression for F^{-1} .
- In the absence of a closed form expression, a numerical approximation for the inverse is required:
 - may involve bijection, secant method, Newton Rhapson method etc.
 - may be computationally costly, and raise convergence issues.



So there are various methods of generating random deviates from non-uniform distributions and the inversion method is the most popular and it is the simplest to use and it is very general all it says is that if you are able to generate standard uniforms and then if you can invert any arbitrary

distribution function then that would give you the deviate from that arbitrary distribution. So here are the steps pictorially on the left you see the uniform density function the standard uniform density function that takes on variables between 0 and 1.

And then that basically corresponds to the cdf the arbitrary cdf F_X which also takes values between 0 and 1. And then suppose u_1 is the first uniform deviate that the computer gave us and then we take that come to the red line which is the cdf of x invert that at that value and we get x_1 . So, that is how we invert u_1 to x_1 the x_1 could be u_2 and that gives us x_2 likewise u_3 gives us x_3 and so on. This method can be very useful but there are certain drawbacks are you need to be able to invert the cdf f quickly and accurately.

If the inverse is not available in closed form so obviously there could be convergence issues. So in such cases and the normal distribution is obviously the most celebrated example of that we have to use other methods for accuracy and speed.

(Refer Slide Time: 05:56)

Monte Carlo simulations

The inversion method for discrete RVs

Theorem: If F is a discrete CDF, and U is a Uniform RV in $(0,1)$, then the RV X defined as $X = \min \{x \mid F(x) \geq U\}$ is distributed according to F .

$$x = F_X^{-1}(u) = \sup \{x_k : F_X(x_k) \geq u, k = 1, 2, \dots\}$$

i.e., set $X = x_k$ iff $p_0 + p_1 + p_2 + \dots + p_{k-1} < u \leq p_1 + p_2 + \dots + p_{k-1} + p_k$

where $p_i = P[X = x_i], p_0 = 0$

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

Structural Reliability
Lecture 23
Capacity demand
component reliability

When you have discrete random variables we can use the same inversion method but we just need to remember that it is not a smooth function the cdf is not a smooth function it is a step function and when we invert we have to keep that in mind. So for example you see a series of these steps the cdf of x the discrete random variable x and if we generate say u_1 as before then

we read off the cdf curve and see which step it belongs to so the all these steps are basically left continuous.

So we make sure to get that particular value you can see u_1 gives us x_1 u_2 gives us x_2 and so on. As we had in the continuous case but here the comparisons have to be done carefully. Here is the algorithm detailed out so you see you keep comparing until the first time the cumulative PMF the CDF exceeds the random number generated the standard uniformity we are generated for the very first time so that gives us the value of x to be returned. Let us look at a few examples of this inversion method how we can code that.

(Refer Slide Time: 07:34)

Monte Carlo simulations

The inversion method: example

Example: Generate $X \sim \text{Exp}(\lambda)$

- Generate $u \sim U(0,1)$
- Invert: $F_X(x) = 1 - \exp(-\lambda x) = u$
- $x = -(1/\lambda)\ln(1-u) \rightarrow -(1/\lambda)\ln(u)$
- Return x


```
n=100;
for mct=1:n
    x=-log(1-rand)/lambda;
end
```

Example: Generate $X \sim \text{Gumbel}(\alpha, m)$

- Generate $u \sim U(0,1)$
- Invert: $F_X(x) = \exp[-\exp\{-\alpha(x-m)\}] = u$
- $x = m - (1/\alpha)\ln\{-\ln(u)\}$
- Return x

```
n=100;
for mct=1:n
    x=m - log(-log(rand))/alpha;
end
```

Structural Reliability
Lecture 23
Capacity demand
component reliability



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

94

So let us take two examples from continuous type distributions. Let us say we want to generate an exponential with parameter lambda our starting point being a standard uniform deviate. So we will generate the uniform u and then equate that to the cdf of the exponential so that gives us $1 - \exp(-\lambda x) = u$ and that we can invert and write x as $-\ln(1-u)/\lambda$.

In some texts and this is also a very good practice you will find that instead of $-\ln(1-u)$ you will find $-\ln(u)$ and the reason is that $1-u$ has the same distribution as a standard uniform deviate obviously not identical to u but for our purpose it serves the same way.

And the benefit is that we for every time we generate x we do not have to do that subtraction operation.

We can just start with u so that saves off sometime from the computational load. We can next example we can look at would be to generate a gumball by the way this is the MATLAB code that we can use to generate such exponentials we have seen this actually earlier in part A. So suppose we want to generate 100 of these exponential deviates and this would be the command you see the rand command which is the MATLAB call to the uniform generator.

Let us next look at how to generate gumball deviates let us say we have x a gumball random variable defined by the two parameters α and m and we need to sample deviates from that distribution so same process using inversion we generate the uniform u standard uniform and then invert the cdf of the gumball at u so going through the algebra x would be $m - 1$ over α times \log of minus \log of u and this is the value that we would return.

The code would look something similar as you see on the left we would have a call to again rand and keep doing it until we have generated the desired number of samples.

(Refer Slide Time: 10:46)

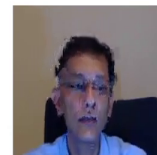
Monte Carlo simulations

Structural Reliability
Lecture 23
Capacity demand
component reliability

Generating independent normal deviates

The normal CDF cannot be inverted in closed form.

- Independent $Z \sim N(0,1)$ deviates can be generated in various ways:
 - numerical inversion of CDF: $z = \Phi^{-1}(u)$ where $u \sim U(0,1)$
 - **Box Muller transform (polar method)**
 - Acceptance-rejection algorithm from Laplace distribution
 - etc.
 - Matlab command is "randn"
- The IID Z 's can be linearly transformed to yield normal deviates with arbitrary mean and variance, $x = \mu + \sigma z$



We could use the inversion method also to generate normals but that would not be very desirable

because the normal cdf cannot be inverted in closed form although you can call such functions but it is generally a little expensive and sometimes there could be convergence issues. So there are more efficient ways of generating normal deviates one is the box Muller transform we did discuss this earlier in this course.

The other is the acceptance rejection algorithm which uses the Laplace distribution etcetera. The MATLAB command is rand n and it also uses the Martian Twister algorithm. Now this rand n gives us or all the other methods that you see they give us the iid standard normals. So, if we call them z then we need to transform these z's to any arbitrary normal and there we would use the property that the normal distribution is closed under linear transformation. So, if we want to generate a normal with mean mu and standard deviation sigma then we would just multiply that z with sigma and add mu and obtain the desired normal deviate.

(Refer Slide Time: 12:36)

Monte Carlo simulations

Generating correlated normals

Recall: Linear combination of joint normals

A vector of n jointly distributed random variables:

$$\underline{X} = [X_1, X_2, \dots, X_n]^T$$

The mean vector is $\underline{\mu}_X$

The covariance matrix is V_{XX}

Consider the linear transformation:

$$\underline{Y} = \underline{a}_0 + \underline{a} \underline{X}$$

where \underline{a}_0 is an m -dimensional vector and \underline{a} is an $m \times n$ coefficient matrix, both non-random

The mean and covariance of \underline{Y} :

$$\underline{\mu}_Y = \underline{a}_0 + \underline{a} \underline{\mu}_X$$

$$V_{YY} = \underline{a} V_{XX} \underline{a}^T$$

If \underline{X} is jointly normal, then \underline{Y} is jointly normal too.

Consider the special case:

$$m = n$$

$$\underline{\mu}_X = \underline{0}$$

$$V_{XX} = [I]$$

i.e., $\underline{X} = \underline{Z}$ is IID standard normal n -vector


Then

$$\underline{Y}$$
 is n dimensional normal with:
$$\underline{\mu}_Y = \underline{a}_0$$

$$V_{YY} = \underline{a} \underline{a}^T$$

- Hence, we can
 - start with a vector of IID standard normals \underline{Z}
 - choose \underline{a} to be lower Cholesky factor of V_{YY} .
 - multiply \underline{a} with \underline{Z}
 - add \underline{a}_0
- And Obtain
 - an n dimensional correlated normal vector \underline{Y}
 - whose mean is \underline{a}_0
 - variance is $\underline{a} \underline{a}^T$

Structural Reliability
Lecture 23
Capacity demand
component reliability



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

96

We could generate correlated normal starting from independent standard normal and let us say we want to generate a vector of n jointly distributed normal variables whose mean vector is $\underline{\mu}$ and the covariance matrix is V_X . So we are going to again use the same linear transformation that we saw but in reverse order. So if we if we have any random vector \underline{X} and we do a linear transformation on that we would get a new random vector \underline{y} with mean and covariance that you see on the screen.

Now if the x's are jointly normal then y is joint normal as well that we know and now we use the special case that the size of y and x are same so m equals n the mean vector of x is 0 and the covariance matrix of x is the identity matrix. So what we are getting at is that the x's are basically the independent standard normal vector. So if that is so then y has a mean which is a 0 and y has a covariance matrix which is a transpose and a was the matrix that we used to create y.

So this tells us that we can actually take this approach to generate any correlated normal vector all we have to do is we start with the IID standard normals and then choose a to be the lower chords key factor of v_y which is what we desire and then multiply a with the z vector and add to that a 0 that is the mean vector that we want. And we have the desired and dimensional correlated norm.

(Refer Slide Time: 15:07)

Monte Carlo simulations

The inversion method for discrete RVs

Algorithm (assuming X is integer valued):

- generate $u \sim U(0,1)$
- set $x = 0, F = p_0$
- while $u > F$
 - $x = x + 1$
 - $F = F + p_x$
- return x

Mean number of comparisons in above algorithm
= $\mu_X + 1$

Example: Bernoulli (p)

- generate $u \sim U(0,1)$
- if $u < p, x = 1$
- else $x = 0$
- return x.

Example: Binomial, X-B(n,p)


$p_x = \text{pmf at } x, F = \text{cdf}$

- set $x = 0, F = p_0, p_x = p_0$
- generate $u \sim U(0,1)$
- while ($F < u$)
 - $x = x + 1$
 - $p_x = p_x (p/q) (n - x + 1) / x$
 - $F = F + p_x$
- do
- return x

Example: Poisson

- set $x = 0, F = \exp(-\mu)$
- generate $u \sim U(0,1)$
- while $F < u$
 - $x = x + 1$
 - $p_x = p_x \mu / x$
 - $F = F + p_x$
- end while
- return x

Structural Reliability
Lecture 23
Capacity demand
component reliability



©Baidurya Bhattacharya IIT Kharagpur www.facweb.iitkgp.ac.in/~baidurya/ 97

Let us move on to Monte Carlo simulations using the inversion method for discrete random variables so the general algorithm is this that we generate the standard uniform u and then let us say x is integer valued so we set x equals 0 and then F is let us set F at p 0 the very first pmf and then as long as the cdf has not exceeded u we keep going to the loop and the very first time that cdf exceeds the uniformly we are generated we exit the loop and return the value of x the integer.

The mean number of comparisons is mean of $x + 1$ so it could get a bit computationally intensive but that is the price of using the inversion method we could generate a Bernoulli random variable very simply in one step we generate the uniform u and if p is greater than u we return x is one or we return x is 0. The next set that you see is the binomial we do a similar comparison until we succeed but here we have taken a slight change in the algorithm which speeds up the process a little more.

We that cdf that you see p^x being replaced by its current value times p over q times n minus x plus 1 over x so that speeds up the computation of the cdf. We could set up a similar algorithm for the Poisson deviate as well. So, all of these; use the inversion method.