Module-6:

Random Number Generation- Properties. Generation of Pseudo-Random Numbers, Techniques for Generation of Pseudo-Random Numbers: Linear Congruential, Combined Linear Congruential, Random Number Streams. Tests for Random Numbers: Frequency Tests and Tests for Autocorrelation. Random Variate Generation- Inverse Transform Techniques for Exponential, Uniform, Weibull, Triangular and for Empirical Continuous Distributions. Acceptance-Rejection Techniques for Poisson (Stationary and Non- Stationary) Distribution and Gamma Distribution. Special Properties like the Direct Transformation for the Normal and Lognormal Distributions, Convolution Method and others.

Random numbers

Random numbers are numbers that occur in a sequence such that two conditions are met:
(1) The values are uniformly distributed over a defined interval or set, and
(2) It is impossible to predict future values based on past or present ones. Random numbers are important in statistical analysis and probability theory.

The most common set from which random numbers are derived is the set of single-digit decimal numbers {0, 1, 2, 3, 4, 5, 6, 7, 8, and 9}.

Use of Random numbers:

- Random numbers can be given as input to some simulation model to test that model. By giving random numbers to model we can find out at which input our simulation model fails to calculate proper result in short it can be used for testing the simulation model.
- Random numbers are used to model timings and behaviour of event.
- Random numbers are important constituent of mathematical modelling.
- Random numbers are also used in simulation of discrete system.

It is necessary to test random numbers because the random numbers we generate are pseudo random numbers and not real and pseudo random number generator must generate a sequence of such random numbers which are uniformly distributed and they should not be correlated, they should not repeat itself.

- In short we can say test is necessary to determine whether the stream of number is random.
- Hypothesis testing is used to test uniformity and independence properties of random numbers. Kolmogorov Smirnov (K-S) test and Chi-Square is used to compare distribution of the set of numbers generated to a uniform distribution.
- Methods for generation of pseudo Random numbers are as follows

1. Linear Congruential Method

2. Combined Linear Congruential Method.
    - Linear Congruential method can be divided into Mixed L.C.M and Multiplicative L.C.M Method.
    - Linear Congruential Method:The linear method was initially proposed by lehmer in 1951. This method produces a sequence of integers, X1, X2… between zero and m-1 by following a recursive relationship.

Xi+1Xi+1=(aXiXi+c)mod m, i=0,1,2,…
Where X0X0 is called as seed.
a is the multiplier

c is increment,

and m is modulus.

- If c is not equal to 0 then the form is called as mixed congruential method.
- And when c is equal to 0 the form is called as multiplicative congruential method. The selection of values for a, c, m, and X0 drastically affects the statistical properties and cycle length.
- Combined Linear Congruential Method: Due to increase in complexity, reliability and problem size the generator with longer period is required. This problem is overcome by combine L.C.M. Here random numbers are generated by following relation.

Xi= ∑((−1)j−1Xi,j(−1)j−1Xi,j) mod m1m1-1
Then Ri=X1/m1Ri=X1/m1 ; XiXi>0
=(m1−1)/m1(m1−1)/m1 ; Xi=0Xi=0
Maximum possible period for such generator is

P=((m1−1)∗(m2−1)∗...(mk−1))/2k−1((m1−1)∗(m2−1)∗...(mk−1))/2k−1

Random number generation

A **random number generator** (**RNG**) is a device that generates a sequence of numbers or symbols that cannot be reasonably predicted well than by a random chance. Random number generators can be true **hardware random-number generators** (HRNG), which generate genuinely random numbers, or pseudo-random number generators (PRNG), which generate numbers that look random, but are actually deterministic, and can be reproduced if the state of the PRNG is known.

Random number generators are important in many kinds of technical applications, including physics, engineering or mathematical computer studies (e.g., Monte Carlo simulations), cryptography and gambling (on game servers).

Essential Properties of a Random Number Generator

Repeatability -- the same sequence should be produced with the same initial values (or *seeds*). This is vital for debugging etc.

Randomness -- should produce independent uniformly distributed random variables that pass all statistical tests for randomness.

Long period -- a pseudo-random number sequence uses finite precision arithmetic, so the sequence must repeat itself with a finite period. This should be much longer than the amount of random numbers needed for the simulation.

Insensitive to seeds -- period and randomness properties should not depend on the initial seeds or starting point/number.

Pseudo Random Number Generator (PRNG)

Pseudo Random Number Generator(PRNG) refers to an algorithm that uses mathematical formulas to produce sequences of random numbers. PRNGs generate a sequence of numbers approximating the properties of random numbers. A pseudorandom number generator (PRNG), also known as a deterministic random bit generator (DRBG), is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.

A PRNG starts from an arbitrary starting state using a seed state. Many numbers are generated in a short time and can also be reproduced later, if the starting point in the sequence is known. Hence, the numbers are deterministic and efficient. It is not possible to generate truly random numbers from deterministic thing like computers so PRNG is a technique developed to generate random numbers using a computer.

How PRNG works?

Linear Congruential Generator is most common and oldest algorithm for generating pseudo-randomized numbers. The generator is defined by the recurrence relation:

$$\mathbf{X_{n+1} = (aX_n + c) \bmod m}$$

where X is the sequence of pseudo-random values

m, $0 < m$ - modulus

a, $0 < a < m$ - multiplier

c, $0 \le c < m$ - increment

x_0, $0 \le x_0 < m$ - the seed or start value

We generate the next random integer using the previous random integer, the integer constants, and the integer modulus. To get started, the algorithm requires an initial Seed, which must be provided by some means. The appearance of randomness is provided by performing **modulo arithmetic.**.

Characteristics of PRNG

- **Efficient:** PRNG can produce many numbers in a short time and is advantageous for applications that need many numbers
- **Deterministic:** A given sequence of numbers can be reproduced at a later date if the starting point in the sequence is known.Determinism is handy if you need to replay the same sequence of numbers again at a later stage.
- **Periodic:** PRNGs are periodic, which means that the sequence will eventually repeat itself. While periodicity is hardly ever a desirable characteristic, modern PRNGs have a period that is so long that it can be ignored for most practical purposes

Applications of PRNG

PRNGs are suitable for applications where many random numbers are required and where it is useful that the same sequence can be replayed easily. Popular examples of such applications are **simulation and modeling applications**. PRNGs are not suitable for applications where it is important that the numbers are really unpredictable, such as **data encryption and gambling.**

Some desired properties of pseudo-random number generators:

- The routine should be fast.
- The routine should be portable across hardware platforms and programming languages.
- The routine should have sufficiently long cycle.
  - A *cycle length* represents the length of the random number sequence before previous numbers begin to repeat themselves in an earlier order. For example
  - `4,9,5,6,9,3,8, 4,9,5,6,9,3,8, 4,9,5,6,9,3,8, ...`

    appears to have a cycle length of 7 (this is just an example of cycle, a random number of cycle 7 is completely unaccetable!).

  - A special case of cycling is *degenerating* where the same random numbers appear repeatedly.
  - Because we use an algorithm to generate random number, cycling cannot be avoided. But long cycles (e.g. a few millions or a few billions) serve the purpose of general simulations.
- The random numbers should be replicable.
- Most importantly, the generated random numbers should closely approximate the ideal statistical properties of uniformity and independence.

Techniques for Generating Random Numbers

The linear congruential method, initially proposed by Lehmer [1951], produces a sequence of integers, X\, X2,... between zero and m — 1 according to the following recursive relationship:

$$X_{i+1} = (a X_i + c) \bmod m, \quad i = 0,1,2,... \quad (7.1)$$

The initial value X0 is called the seed, a is called the constant multiplier, c is the increment, and m is the modulus.

If c _ 0 in Equation (7.1), the form is called the mixed congruential method. When c = 0, the form is known as the multiplicative congruential method. The selection of the values for a, c, m

and Xo drastically affects the statistical properties and the cycle length. . An example will illustrate how this technique operates.

EXAMPLE 4.1
Use the linear congruential method to generate a sequence of random numbers with X0 = 27,
a= 17, c = 43, and m = 100. Here, the integer values generated will all be between zero and 99
because of the value of the modulus . These random integers should appear to be uniformly
distributed the integers zero to 99.Random numbers between zero and 1 can be generated by
Ri =Xi/m, i= 1,2,…… (7.2)
The sequence of Xi and subsequent Ri values is computed as follows:
X0 = 27
X1 = (17.27 + 43) mod 100 = 502 mod 100 = 2
R1=2⁄100=0. 02
X2 = (17 • 2 + 43) mod 100 = 77 mod 100 = 77
R2=77⁄100=0. 77
*[3pt] X3 = (17•77+ 43) mod 100 = 1352 mod 100 = 52
R3=52⁄100=0. 52

First, notice that the numbers generated from Equation (7.2) can only assume values from
the set I = {0,1 /m, 2/m,..., (m — l)/m), since each Xi is an integer in the set {0,1,2,..., m — 1}.
Thus, each Ri is discrete on I, instead of continuous on the interval [0, 1], This approximation
appears to be of little consequence, provided that the modulus m is a very large integer. (Values
such as m = 231 — 1 and m = 248 are in common use in generators appearing in many simulation
languages.) By maximum density is meant that the values assumed by Ri = 1, 2,..., leave no large
gaps on [0,1]
Second, to help achieve maximum density, and to avoid cycling (i.e., recurrence of the same sequence of generated numbers) in practical applications, the generator should have the
largest possible period. Maximal period can be achieved by the proper choice of a, c, m, and X0 .
• For m a power of 2, say m =2b and c _ 0, the longest possible period is P = m = 2b, which
is achieved provided that c is relatively prime to m (that is, the greatest common factor of
c and m i s l), and =a = l+4k, where k is an integer.
• For m a power of 2, say m =2b and c = 0, the longest possible period is P = m⁄4 = 2b-2,
which is achieved provided that the seed X0 is odd and the multiplier ,a, is given by
a=3+8K , for some

K=0,1,..
• For m a prime number and c=0, the longest possible period is P=m-1, which is achieved
provided that the multiplier , a, has the property that the smallest integer k such that a k- 1is divisible by m is k= m-1.

Combined Linear Congruential Generators

Combined linear congruential generators, as the name implies, are a type of PRNG (pseudorandom number generator) that combine two or more LCGs (linear congruential generators). The combination of two or more LCGs into one random number generator can result in a marked increase in the period length of the generator which makes them better suited for simulating more complex systems. The combined linear congruential generator algorithm is defined as:

$$X_i \equiv \left(\sum_{j=1}^{k} (-1)^{j-1} Y_{i,j} \right) \pmod{m_1 - 1}$$

where:

m_1 is the "modulus" of the first LCG

$Y_{i,j}$ is the i^{th} input from the j^{th} LCG

X_i is the i^{th} generated random integer

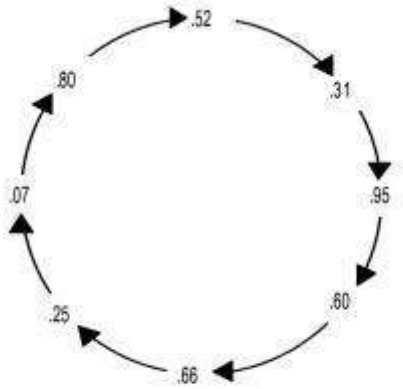
with:

$$R_i \equiv \begin{cases} X_i / m_1 & \text{for } X_i > 0 \\ (m_1 - 1)/m_1 & \text{for } X_i = 0 \end{cases}$$

where R_i is a uniformly distributed random number between 0 and 1.

Random Number Streams

A stream is a sequence of independently cycling, unique random numbers uniformly distributed between 0 and 1 (see the figure on next page). Random number streams are used to generate additional random numbers from other probability distributions (Normal, Beta, Gamma). After sequencing through all of the random numbers in the cycle, the cycle starts over again with the same sequence. The length of the cycle before it repeats is called the *cycle period* and is usually very long.



Example of a Random Stream Cycle with a Very Short Period

A random stream is generated using a random number generator or equation. The random number generator begins with an initial seed value after which, each successive value uses the previous value as input to the generator. Each stream used in a simulation has its own independent seed and tracks its own values for subsequent input to the generator. Where the sequence begins in the cycle depends on the initial seed value used by the generator.

Any time a particular number seeds a stream, the same sequence of values will be repeated every time the same seed is used to initialize the stream. This means that various elements within a model can be held constant with respect to their performance while other elements vary freely. Simply specify one random number stream for one set of activities and another random number stream for all other activities.

Because the same seed produces the same sequence of values every time it is used, completely independent functions within a model must have their own streams from the start. For example, arrival distributions should generally have a random number stream used nowhere else in the entire model. That way, activities added to a model that sample from a random number stream will not inadvertently alter the arrival pattern because they do not affect the sample values generated from the arrival distribution.

To show an example of how multiple streams can be useful, consider two copy machines, Copy1 and Copy2, which go down approximately every 4 hours for servicing. To model this, the frequency or time between failures is defined by a normal distribution with a mean value of 240 minutes and a standard deviation of 15 minutes, N(240,15). The time to repair is 10 minutes. If no stream is specified in the normal distribution, the same stream will be used to generate sample values for both machines. So, if the next two numbers in the stream number are .21837 and .86469, Copy1 will get a sample value from the normal distribution that is different from Copy2. Therefore, the two machines will go down at different times.

Suppose, however, that the resource servicing the machines must service them both at the same time, so we would like to have the machines go down at the same time. Using the same stream to determine both downtimes will not bring them down at the same time, because a different random number will be returned from the stream with each call to generate a random normal variate. Using two different streams, each dedicated to a machine's downtime and each having the same initial seed, will ensure that both machines go down at the same time every

time. The two streams have the same starting seed value so they will produce exactly the same sequence of random numbers.

Using Random Number Streams

One of the most valuable characteristics of simulation is the ability to replicate and isolate probabilistic functions and activities within a system for specific study. In the real world, events tend to occur randomly, according to a certain statistical pattern or distribution. To help you model this randomness, ProModel uses distributions.

When you include a distribution (e.g., Normal, Beta, and Gamma) in your model, ProModel uses a random number generator to produce a set sequence or *stream* of numbers between 0 and 1 ($0 <= x < 1$) to use in the distribution. Before it can select any numbers, however, ProModel requires an *initial seed value* to identify the point in the stream at which to begin. Once you specify a seed value, ProModel "shifts" the random number selection (in increments of 100,000 numbers) by that number of positions and starts sampling values. Since there is only one random number stream, this will ensure that the selected values do not overlap. ProModel includes 100 seed values, and each seed produces a unique set of random numbers. If you do not specify an initial seed value, ProModel will use the stream number as the seed value (i.e., stream 3 uses seed 3).

When you use a specific seed value (e.g., 17), ProModel produces a unique sequence of numbers to use each time you apply that seed value. This allows you to maintain the consistency of some model elements and permit other elements to vary. (To do this, specify one random number stream for the set of activities you wish to maintain constant, and another random number stream for all other sets of activities.) In fact, because each seed value produces the same sequence of values every time, completely independent model functions must use their own streams. For example, Arrival distributions specified in the Arrival Module should have a random number stream used nowhere else in the model. This will prevent activities that sample random stream values from inadvertently altering the arrival pattern (i.e., the activities will not affect the sample values generated from the arrival distribution).

Tests for Random Numbers

The desirable properties of random numbers — uniformity and independence. To insure that these desirable properties are achieved, a number of tests can be performed The tests can be placed in two categories according to the properties of interest, The first entry in the list below concerns testing for uniformity. The second through fifth entries concern testing for independence. The five types of tests

1. **Frequency test**: Uses the Kolmogorov-Smirnov or the chi- square test to compare the distribution of the set of numbers generated to a uniform distribution.
2. **Runs test**: Tests the runs up and down or the runs above, and below the mean by comparing the actual values to expected values. The statistic for comparison is the chi-square.

3. **Autocorrelation test** Tests the correlation between numbers and compares the sample correlation to the expected correlation of zero.
4. **Gap test**. Counts the number of digits that appear between repetitions of particular digit and then uses the Kolmogorov-Smirnov test to compare with the expected size of gaps,
5. **Poker test** Treats numbers grouped together as a poker hand. Then the hands obtained are compared to what is expected using the chi-square test.

Frequency test

- The frequency test is a test of uniformity.
- Two different methods available, Kolmogorov-Smirnov test and the chi-square test. Both tests measure the agreement between the distribution of a sample of generated random numbers and the theoretical uniform distribution.
- Both tests are based on the null hypothesis of no significant difference between the sample distribution and the theoretical distribution.

The Kolmogorov-Smirnov test

This test compares the cdf of uniform distribution *F(x)* to the empirical cdf $S_N(X)$ of the sample of *N* observations.

- $F(x) = x \quad 0 \leq x \leq 1$
- $S_N(x) = \frac{\text{number of } R_1, R_2, \ldots, R_N \leq x}{N}$
- $S_N(x)$
- As *N* becomes larger, $S_N(x)$ should be close to *F(x)*
- Kolmogorov-Smirnov test is based on the statistic

$$D = \max|F(x) - S_N(x)|$$

that is the absolute value of the differences.

- Here *D* is a random variable, its sampling distribution is tabulated in Table A.8.
- If the calcualted *D* value is greater than the ones listed in the Table, the hypothesis (no disagreement between the samples and the theoretical value) should be rejected; otherwise, we don't have enough information to reject it.
- Following steps are taken to perform the test.
    1. Rank the data from smallest to largest

$$R_{(1)}, \leq R_{(2)}, \leq \ldots \leq R_{(N)}$$

2. Compute

$$D^+ = \max_{1 \leq i \leq N} \left\{ \frac{i}{N} - R_{(i)} \right\}$$

$$D^- = \max_{1 \leq i \leq N} \left\{ R_{(i)} - \frac{i-1}{N} \right\}$$

3. Compute $D = \max(D^+, D^-)$
4. Determine the critical value, D_α, from Table A.8 for the specified significance level α and the given sample size *N*.
5. If the sample statistic *D* is greater than the critical value D_α, the null hypothsis that the sample data is from a uniform distribution is rejected; if $D \leq D_\alpha$, then there is no evidence to reject it.

- Example 8.6 on page 300.

Chi-Square test

The chi-square test looks at the issue from the same angle but uses different method. Instead of measure the difference of each point between the samples and the true distribution, chi-square checks the ``deviation'' from the ``expected'' value.

$$x_0^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

where *n* is the number of classes (e.g. intervals), O_i is the number of samples obseved in the interval, E_i is expected number of samples in the interval. If the sample size is *N*, in a uniform distribution,

$$E_i = \frac{N}{n}$$

What is autocorrelation?

Autocorrelation is a statistical test that determines whether a random number generator is producing independent random numbers in a sequence.

What does autocorrelation test?

- The tests for autocorrelation are concerned with the dependence between numbers in a sequence.
- The test computes the autocorrelation between every m numbers (m is also known as the lag) starting with the ith number (i is also known as the index).

What does autocorrelation look like?

Here is a list of random integers generated. See if you can identify a pattern by looking at the values in the sequence.

1	2	3	4	5	6	7	8	9	10
0.63	0.28	0.30	0.42	0.97	0.05	0.71	0.63	0.17	1.0
0.61	0.19	0.94	0.64	0.84	0.54	0.56	0.57	0.09	0.99
0.01	0.10	0.69	0.38	0.93	0.85	0.68	0.14	0.18	0.84
0.19	0.71	0.44	0.72	0.95	0.28	0.96	0.51	0.50	0.89
0.66	0.31	0.50	0.33	0.89	0.54	0.73	0.76	0.62	0.92

If you look carfully at the following table of random integers generated, you'll notice that every number in the 5th, 10th, 15th, and 20th postion is a larger value.

1	2	3	4	5	6	7	8	9	10

0.63	0.28	0.30	0.42	0.97	0.05	0.71	0.63	0.17	1.0
0.61	0.19	0.94	0.64	0.84	0.54	0.56	0.57	0.09	0.99
0.01	0.10	0.69	0.38	0.93	0.85	0.68	0.14	0.18	0.84
0.19	0.71	0.44	0.72	0.95	0.28	0.96	0.51	0.50	0.89
0.66	0.31	0.50	0.33	0.89	0.54	0.73	0.76	0.62	0.92

Tests for Auto-correlation

- The tests for auto-correlation are concerned with the dependence between numbers in a sequence.
- The list of the 30 numbers on page 311 appears to have the effect that every 5th number has a very large value. If this is a regular pattern, we can't really say the sequence is random.
- The test computes the auto-correlation between every *m* numbers (*m* is also known as the lag) starting with the *i*th number.

Thus the autocorrelation ρ_{im} between the following numbers would be of interest.

$$R_i, R_{i+m}, R_{i+2m}, \ldots R_{i+(M+1)m}$$

The value *M* is the largest integer such that $i + (M+1)m \leq N$ where *N* is the total number of values in the sequence.

E.g. N = 17, i = 3, m = 4, then the above sequence would be 3, 7, 11, 15 (M = 2). The reason we require *M+1* instead of *M* is that we need to have at least two numbers to test (M = 0) the autocorrelation.

- Since a non-zero autocorrelation implies a lack of independence, the following test is appropriate

$$H_0: \quad \rho_{im} = 0$$

$$H_1: \quad \rho_{im} \neq 0$$

- For large values of *M*, the distribution of the estimator ρ_{im}, denoted as $\hat{\rho_{im}}$, is approximately normal if the values $R_i, R_{i+m}, R_{i+2m}, \ldots R_{i+(M+1)m}$ are uncorrelated.
- Form the test statistic

$$Z_0 = \frac{\hat{\rho_{im}}}{\sigma_{\hat{\rho_{im}}}}$$

which is distributed normally with a mean of zero and a variance of one.

- The actual formula for $\hat{\rho_{im}}$ and the standard deviation is

$$\hat{\rho_{im}} = \frac{1}{M+1}\left[\sum_{k=0}^{M} R_{i+km}R_{(k+1)m}\right] - 0.25$$

and

$$\sigma_{\hat{\rho_{im}}} = \frac{\sqrt{13M+7}}{12(M+1)}$$

- After computing Z_0, do not reject the null hypothesis of independence if

$$-z_{\alpha/2} \leq Z_0 \leq z_{\alpha/2}$$

where α is the level of significance.

Drawbacks when using autocorrelation

- Autocorrelation is not very sensitive to small values of M, when the values being tested are on the low side. For example, if all the values were equal to zero, then the resulting value would be -1.95, which is not enough to reject the hypothesis.
- Many sequences can be formed in a set of date (the sequence of all numbers, the sequence from the first, fifth, ..., numbers and so forth. If the alpha is equal to 0.05, then there is a possibility of rejecting a true hypothesis. For example, if these are 10 independent events, the possibility of finding of finding no autocorrelation alone is

$(0.95)^{10}$ or 0.60. Thus, 40% of the time signifigant autocorrelation would be detected when it does not exist.

The Conclusion

When using autocorrelation tests, be cautious. Autocorrelation may be detected after numerous tests even when no autocorrelation exists.

Random Vitiate Generation

In the mathematical fields of probability and statistics, a **random variate** is a particular outcome of a random variable: the random variates which are other outcomes of the same random variable might have different values. A **random deviate** or simply **deviate** is the difference of random variate with respect to the distribution central location (e.g., mean), often divided by the standard deviation of the distribution. Random variates are used when simulating processes driven by random influences (stochastic processes). In modern applications, such simulations would derive random variates corresponding to any given probability distribution from computer procedures designed to create random variates corresponding to a uniform distribution, where these procedures would actually provide values chosen from a uniform distribution of pseudorandom numbers.

Procedures to generate random variates corresponding to a given distribution are known as procedures for *random variate generation* or pseudo-random number sampling.

In probability theory, a random variable is a measurable function from a probability space to a measurable space of values that the variable can take on. In that context, those values are also known as random variates or random deviates, and this represents a wider meaning than just that associated with pseudorandom numbers.

Inverse Transform Technique

- The inverse transform technique can be used to sample from exponential, the uniform, the Weibull and the triangle distributions.
- The basic principle is to find the inverse function of F, F^{-1} such that $F\,F^{-1} = F^{-1}\,F = I$.
- F^{-1} denotes the solution of the equation $r = F(x)$ in terms of *r*, not $1/F$. For example, the inverse of y = x is x = y, the inverse of y = 2 x + 1 is x = (y-1)/2 ...

Exponential Distribution

- pdf

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- cdf

$$F(x) = \int_{-\infty}^{x} f(t)dt = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

- The idea is to solve $y = 1 - e^{-\lambda x}$ for *x* where *y* is uniformly distributed on (0,1) because it is a cdf. Then *x* is exponentially distributed.
- This method can be used for any distribution in theory. But it is particularly useful for random variates that their inverse function can be easily solved.
- Steps involved are as follows.

Step 1.

Compute the cdf of the desired random variable X.

For the exponential distribution, the cdf is $F(x) = 1 - e^{-\lambda x}$.7

Step 2.7

Set *R = F(X)* on the range of X.

For the exponential distribution, $R = 1 - e^{-\lambda x}$ on the range of $x \geq 0$.

Step 3.

Solve the equation *F(X) = R* for X in terms of R.

For the exponential distribution, the solution proceeds as follows.

$$\begin{aligned} 1 - e^{-\lambda X} &= R \\ e^{-\lambda X} &= 1 - R \\ -\lambda X &= ln(1 - R) \\ X &= \tfrac{-1}{\lambda} ln(1 - R) \end{aligned}$$

Step 4.

Generate (as needed) uniform random numbers $R_1, R_2, \ldots$ and compute the desired random variates by

$$X_i = F^{-1}(R_i)$$

In the case of exponential distribution

$$X_i = \frac{-1}{\lambda} ln(1 - R_i)$$

for *i = 1, 2, 3, ...* where R_i is a uniformly distributed random number on (0,1).

In practice, since both R_i AND $1 - R_i$ are uniformly distributed random number, so the calculation can be simplified as

$$X_i = \frac{-1}{\lambda} ln\ R_i$$

- To see why this is correct, recall

$$X_i = F^{-1}(R_i)(*)$$

and

$$R_i = F(X_i)(**)$$

Because $X_i \leq x_0$ is equivelant to $F^{-1}(R_i) \leq x_0$, and F is a non-decreasing function (so that if $x \leq y$ then $F(x) \leq F(y)$) we get $X_i \leq x_0$ is equivelant to $F^{-1}(R_i) \leq x_0$, which implies that $F(F^{-1}(R_i)) \leq F(x_0)$ which is equivelant to $R_i \leq F(x_0)$.

This means

$$P(X_i \leq x_0) = P(R_i \leq F(x_0))$$

Since R_i is uniformly distributed on (0,1) and $F(x_0)$ is the cdf of exponential function which is between 0 and 1, so

$$P(R_i \leq F(x_0)) = F(x_0)$$

which means

$$P(X_i \leq x_0) = F(x_0)$$

This says the $F(x_0)$ is the cdf for X and X has the desired distribution.

Uniform Distribution

- If we want a random variate X uniformly distributed on the interval [a,b], a reasonable guess for generating X is given by

$$X = a + (b - a)R$$

where R is uniformly distributed on (0,1).

If we follow the steps outlined in previous section, we get the same result.

- pdf for X

$$f(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & otherwise \end{cases}$$

- steps:

Step 1.
the cdf

$$F(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x \le b \\ 1 & x > b \end{cases}$$

Step 2.

Set $F(X) = (X - a)/(b - a) = R$

Step 3.

Solve for X is terms of R yields

$$X = a + (b - a)R$$

which is the same as the earlier guess.

Step 4.

Generate R_i as needed, calcualte X_i using the function obtained.

Weibull Distribution

- pdf

$$f(x) = \begin{cases} \frac{\beta}{\alpha^\beta} x^{\beta-1} e^{-(x/\alpha)^\beta}, & x \geq 0 \\ 0, & otherwise \end{cases}$$

- Steps:

  Step 1.
  cdf

$$F(x) = 1 - e^{-(x/\alpha)^\beta}$$

  for $x \geq 0$

  Step 2.

  $F(x) = 1 - e^{-(x/\alpha)^\beta} = R$
  let

  Step 3.

  Solve for X in terms of R yields

$$X = \alpha[-ln(1-R)]^{1/\beta}$$

  Step 4.

  Generate uniformly distributed R_i from (0,1) feeding them to the function in Step 3 to get X.

Triangular Distribution

- cdf

$$f(x) = \begin{cases} x & 0 \le x \le 1 \\ 2 - x & 1 < x \le 2 \\ 0 & otherwise \end{cases}$$

Its curve is shown on page 328

- Steps

  Step 1.
  cdf

$$F(x) = \begin{cases} 0 & x \le 0 \\ \frac{x^2}{2} & 0 < x \le 1 \\ 1 - \frac{(2-x)^2}{2} & 1 < x \le 2 \\ 1 & x > 2 \end{cases}$$

  Step 2.

let $R = \frac{X^2}{2} \text{ for } 0 < X \le 1$ and $R = 1 - \frac{(2-x)^2}{2} \text{ for } 1 < x \le 2$

  Step 3.

  Solve X in terms of R

$$X = \begin{cases} \sqrt{2R} & 0 \le R \le \frac{1}{2} \\ 2 - \sqrt{2(1-R)} & \frac{1}{2} < R \le 1 \end{cases}$$

Empirical Continuous Distributions

- If the modeler has been unable to find a theoretical distribution that provides a good model for the input data, it may be necessary to use the empirical distribution of the data.
- A typical way of resolving this difficult is through ``curve fitting''.
- Steps involved: (see Exmaple 9.2 on page 328)
  - Collect empirical data and group them accordingly.
  - Tabulate the frequency and cumulative frequency.

- Now assume the value of cumulative frequency as a function of the empirical data, i.e. F(x) = r
- Establish a relation between *x* and *r* using linear interpolation

$$X_1 = x_a + \frac{R_1 - y_a}{y_b - y_a}(x_b - x_a)$$

for each of the intervals.

Direct Transformation for the Normal Distribution

In the previous section, a simple, but less accurate method of generating a normal distribution was presented. Here we consider a direct transormation.

- Let Z_1, Z_2 be two standard normal random variates.
- Plot the two as a point in the plane and represent them in a polar coordinate system as

$$Z_1 = B\cos\theta$$
$$Z_2 = B\sin\theta$$

- It is known that $B^2 = {Z_1}^2 + {Z_2}^2$ has the chi-square distribution with 2 degrees of freedom, which is equivalent to an exponential distribution with mean 2

$$Y = \lambda e^{-\lambda t}, \quad t \geq 0$$

$$E[Y] = 2 = \lambda$$

thus the raidus *B* can be generated using (9.3)

$$B = \sqrt{-2 \ln R}$$

- So a normal distribution can be generated by any one of the following.

$$Z_1 = \sqrt{-2 \ln R_1} \cos(2\pi R_2)$$

$$Z_2 = \sqrt{-2 \ln R_1} \sin(2\pi R_2)$$

where R_1 and R_2 are uniformly distributed over (0,1).

- To obtain normail variates X_i with mean μ and variance σ^2, transform

$$X_i = \mu + \sigma Z_i$$

Acceptance-Rejection Technique to Generate Random Variate

- Example: use following steps to generate uniformly distributed random numbers between 1/4 and 1.

  Step 1.
  Generate a random number *R*
  Step 2a.
  If $R \geq 1/4$, accept *X* = *R*, goto Step 3
  Step 2b.
  If $R < 1/4$, reject *R*, return to Step 1
  Step 3.
  If another uniform random variate on [1/4, 1] is needed, repeat the procedure begining at Step 1. Otherwise stop.

  - Do we know if the random variate generated using above methods is indeed uniformly distributed over [1/4, 1]? The answer is Yes. To prove this, use the definition. Take any $1/4 \leq a < b \leq 1$,

$$P(a < R \leq b | 1/4 \leq R \leq 1) = \frac{P(a < R \leq b)}{P(1/4 \leq R \leq 1)} = \frac{b-a}{3/4}$$

which is the correct probability for a uniform distribution on [1/4,1].

- The efficiency: use this method in this particular example, the rejection probability is 1/4 on the average for each number generated. The number of rejections is a geometrically distributed random variable with probability of ``success" being *p = 3/4*, mean number of rejections is *(1/p - 1) = 4/3 - 1 = 1/3* (i.e. 1/3 waste).
- For this reason, the inverse transform (X = 1/4 + (3/4) R) is more efficient method.

- **Poisson Distribution**
  - pmf

$$p(n) = P(N = n) = \frac{e^{-\alpha}\alpha^n}{n!}, n = 0, 1, 2, ...$$

where *N* can be interpreted as the number of arrivals in one unit time.

- From the original Poisson process definition, we know the interarrival time $A_1, A_2, ...$ are exponentially distributed with a mean of α, i.e. α arrivals in one unit time.
- Relation between the two distribution:

$$N = n$$

if and only if

$$A_1 + A_2 + ... + A_n \leq 1 < A_1 + A_2 + ... + A_n + A_{n+1}$$

essentially this means if there are *n* arrivals in one unit time, the sum of interarrival time of the past *n* observations has to be less than or equal to one, but if one more interarrival time is added, it is greater then one (unit time).

- The A_i s in the relation can be generated from uniformly distributed random number $A_i = (-1/\alpha) \ln R_i$, thus

$$\sum_{i=1}^{n} \frac{-1}{\alpha} \ln R_i \leq 1 < \sum_{i=1}^{n+1} \frac{-1}{\alpha} \ln R_i$$

both sides are multiplied by $-\alpha$

$$\ln \prod_{1}^{n} R_i = \sum_{i=1}^{n} \ln R_i \geq -\alpha > \sum_{i=1}^{n+1} \ln R_i = \ln \prod_{1}^{n+1} R_i$$

that is

$$\prod_{1}^{n} R_i \geq e^{-\alpha} > \prod_{1}^{n+1} R_i$$

- Now we can use the Acceptance-Reject method to generate Poisson distribution.

Step 1.

Set *n = 0, P = 1*.

Step 2.

Generate a random number R_{n+1} and replace *P* by $P * R_{n+1}$.

Step 3.

If $P < e^{-\alpha}$, then accept *N = n*, meaning at this time unit, there are *n* arrivals. Otherwise, reject the current *n*, increase *n* by one, return to Step 2.

- Efficiency: How many random numbers will be required, on the average, to generate one Poisson variate, *N*? If *N = n*, then *n+1* random numbers are required (because of the (n+1) random numbers product).

$$E(N+1) = \alpha + 1$$

- Example 9.10 on page 346, Example 9.11 on page 347
- When α is large, say $\alpha \leq 15$, the acceptance-rejection technique described here becomes too expensive. Use normal distribution to approximate Poisson distribution. When α is large

$$Z = \frac{N - \alpha}{\sqrt{\alpha}}$$

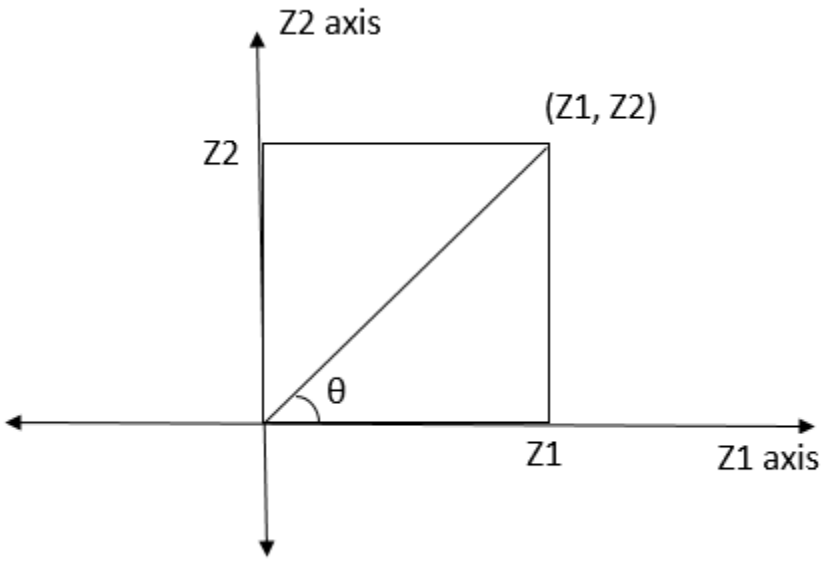
is approximately normally distributed with mean 0 and variance 1, thus

$$N = \lceil \alpha + \sqrt{\alpha} Z - 0.5 \rceil$$

can be used to generate Poisson random variate.

Special Properties like the Direct Transformation for the Normal and Lognormal Distributions

Given std. normal variable z ? N (0,1). we can obtain XσN(μ,62)XσN(μ,62) by setting x=μ+62x=μ+62
consider 2 std. normal variables 2, and Z2, plotted as a point in the plane as shown below:



Z1=BcasθZ2=BsinθZ1=BcasθZ2=Bsinθ -------- (1)
B2=Z21+Z22B2=Z12+Z22
∴B2∴B2 has chi-square dist'n with 2 d of, which is equivalent to exp. dist'n with mean 2.
B can be generated using inverse transform tech. for exp dist'n.

∴B2=−2lnR∴B2=−2lnR
∴B=−2lnR−−−−−−√∴B=−2lnR
Angle θθ is uniformly dist. on (o,2π)(o,2π) and independent on B.
From (1) and (2) gives direct method for generating 2 independent std. normal
var. Z1andZ2Z1andZ2 from R1andR2R1andR2
∴Z1=−2lnR1caszπR2−−−−−−−−−−−−−√∴Z1=−2lnR1caszπR2
Z2=−2lnR1sin2πR2−−−−−−−−−−−−−√Z2=−2lnR1sin2πR2
obtain normal variate xi

xi=μ+?zixi=μ+?zi
finally obtain lag-normal variate by using following direct transf'n

yi=exiyi=exi
e.g. let R1=0.1758,R2=0.1489mean=10,σ2=4R1=0.1758,R2=0.1489mean=10,σ2=4
Z1=−2lnR1−−−−−−√cas2πR2Z1=−2lnR1cas2πR2
=−2ln0.1758−−−−−−−−−√cas2π×0.1489=1.11=−2ln0.1758cas2π×0.1489=1.11
Z2=−2lnR1−−−−−−√sin2πR2Z2=−2lnR1sin2πR2
=−2ln0.1758−−−−−−−−−√sin2π×0.1489=−2ln0.1758sin2π×0.1489
= 1.5

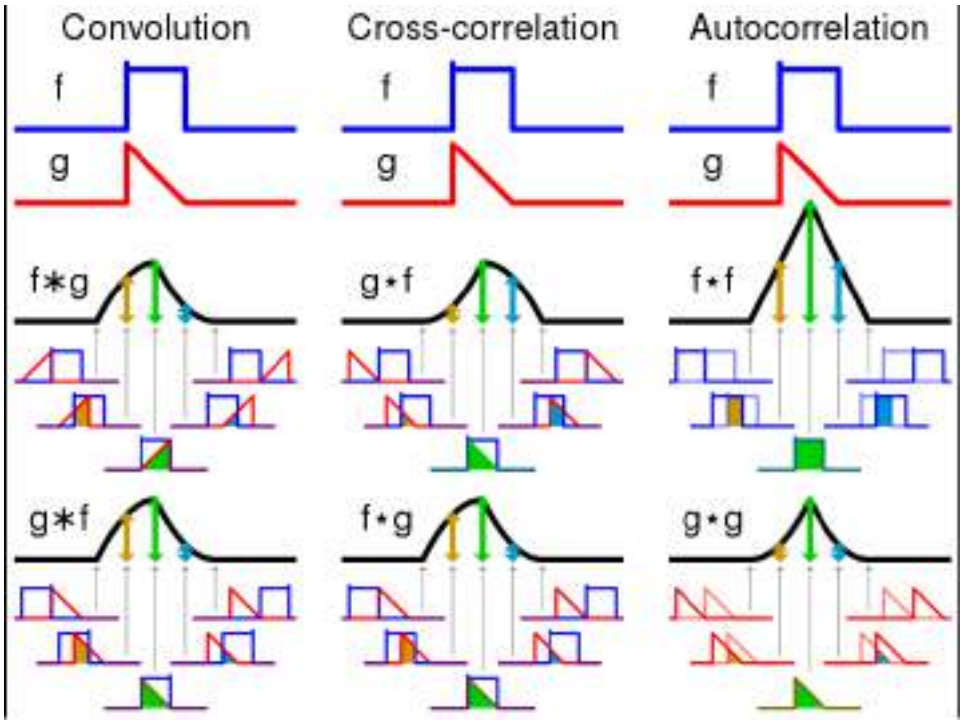
Normal variate,

xi=μ+σ2ixi=μ+σ2i
μ=10,σ2=4∴σ=2μ=10,σ2=4∴σ=2
∴x1=μ+σ21∴x1=μ+σ21
=10+2(1.11)=12.22=10+2(1.11)=12.22
X2=μ+σ22X2=μ+σ22
= 10 + 2 (1.5) = 13

Convolution

In mathematics (in particular, functional analysis) **convolution** is a mathematical operation on two functions (*f* and *g*) that produces a third function expressing how the shape of one is modified by the other. The term *convolution* refers to both the result function and to the process of computing it. It is defined as the integral of the product of the two functions after one is reversed and shifted.

Some features of convolution are similar to cross-correlation: for real-valued functions, of a continuous or discrete variable, it differs from cross-correlation only in that either $f(x)$ or $g(x)$ is reflected about the y-axis; thus it is a cross-correlation of $f(x)$ and $g(-x)$, or $f(-x)$ and $g(x)$.[A] For continuous functions, the cross-correlation operator is the adjoint of the convolution operator.



Convolution has applications that include probability, statistics, computer vision, natural language processing, image and signal processing, engineering, and differential equations.The convolution can be defined for functions on Euclidean space, and other groups.[*citation needed*] For example, periodic functions, such as the discrete-time Fourier transform, can be defined on a circle and convolved by *periodic convolution*. (See row 18 at DTFT § Properties.) A *discrete convolution* can be defined for functions on the set of integers.

Generalizations of convolution have applications in the field of numerical analysis and numerical linear algebra, and in the design and implementation of finite impulse response filters in signal processing.[*citation needed*]

Computing the inverse of the convolution operation is known as deconvolution.

Convolution Method

- The probability distribution of a sum of two or more independent random variables is called a *convolution* of the distributions of the original variables.
- Erlang distribution.
  - an Erlang random variable *X* with parameters (K,θ) can be shown to be the sum of *K* independent exponential random variables $X_i \quad (i = 1,...,K)$, each having a mean $1/K\theta$

$$X = \sum_{i=1}^{K} X_i$$

- Using equation (9.3) that can generate exponential variable, an Erlang variate can be generated by

$$\begin{aligned} X &= \sum_{i=1}^{K} \frac{-1}{K\theta} \ln R_i \\ &= \frac{-1}{K\theta} \ln \left(\prod_{i=1}^{K} R_i \right) \end{aligned}$$