Exercise 3:

Random Number Generators

Based on Exercise 2, implement the random number generators to generate the random Inter-arrival time T.

1) Implement the Linear Congruential random number Generator (LCG) in OMNeT++ based on the method:

$$d_i = 16807\, d_{i-1} \bmod \left(2^{31} - 1\right)$$

2) Generate the uniform distributed Inter-arrival time T (e.g. uniform (0,2)) based on the LCG RNG.

3) Generate the exponential distributed T (e.g. with exponential(1)).

4) Compare the new simulation results with the ones using the OMNeT++ default RNG (Mersenne Twister).

5) Which characteristics should “good” random number generators have? Do you think the LCG RNG is a “good” RNG and why?

6) In a simulation program for a cellular mobile radio system it is required to initially distribute N mobile stations uniformly in the area of a circle with radius R. For each mobile station the angle φ_i and radius r_i has to be generated. Describe the algorithm

Hints

1) For generating the exponential distribution, please refer to exercise 4 and 5 in the optional tasks part (inverse CDF method to generate other distributions based on uniform distribution).

2) Optional tasks are not required to finish. But they are quite useful for you to get further understanding of random number generators. You can only get the solutions for the optional tasks if you have done them.

Optional Tasks

1) Compare the advantages and disadvantages of linear congruential generators and the proposed generator in (1.1) – a variant of the Lagged Fibonacci Generator – consider implementation and application aspects (excluding the statistical characteristics).

$$\left.\begin{aligned} v_i &= \left(v_{i-97} - v_{i-33}\right) \bmod 2^{32} \\ c_i &= \left(c_{i-1} - 362436069\right) \bmod 2^{32} \\ d_i &= \left(v_i - c_i\right) \bmod 2^{32} \\ N &= \left(2^{32} - 1\right) 2^{96} \left(\text{cycle length}\right) \end{aligned}\right\} \quad (1.1)$$

2) Use the following linear congruential random number generator:

$$d_i = 16807\, d_{i-1} \bmod \left(2^{31} - 1\right)$$

(a) How many bits are needed for the largest possible multiplication?
What is the implication for the implementation of this generator on a computer with 32 bit arithmetic?

(b) The following identity holds (Schrage 1979):

$$ax \ \bmod\ m = g(x) + mh(x) \quad \text{with}$$
$$g(x) = a\left(x \ \bmod\ q\right) - r\left(x \ \operatorname{div}\ q\right) \quad \text{and} \quad h(x) = \left(x \ \operatorname{div}\ q\right) - \left(ax \ \operatorname{div}\ m\right)$$
$$q = m \ \operatorname{div}\ a \quad \text{and} \quad r = m \ \bmod\ a$$

For *r<q* we have *h(x)=1* only if *g(x)<0* otherwise *h(x)=0.*

How many bits are now used for the largest possible multiplication?

Give an implementation of the generator in a (pseudo-) programming language using the algorithm of Schrage.

3) Prove the following relationship (3.3) by using the recursive equation (3.2) using complete induction (*vollständige Induktion*)

$$d_{i+1} = (ad_i + c)(\text{mod } m) \qquad (3.2)$$

$$d_i = [a^i d_0 + \frac{c(a^i - 1)}{a-1}](\text{mod } m) \qquad for \qquad i = 1,2,... \qquad (3.3)$$

Hint: use *((u mod m) +/- (v mod m)) mod m = (u +/- v) mod m*
Which practical applications are possible for the relationship (3.3)?

4) Assume F^{-1} is the inverse of a monotone continuous distribution function. Show that if U is a uniformly distributed random variable in the interval [0, 1], then $F^{-1}(U)$ has the distribution function F.

How can this method be used in simulation programs?
Where are the advantages and disadvantages of this method?

5) Calculate the inverse functions for the generation from a uniformly distributed random variable U of:

- a random variable X that is neg. exponentially distributed
- a random variable Y that has a triangle PDF: $f_Y(y)$ = triang(0,1,c)

The neg. exponential CDF is defined as: $F_X(x) = 1 - e^{-\lambda x}$