**Part I: PRNG and Random Variables**

1. Create a function that implements a linear congruential generator (LCG), accepting as input the parameters: seed, m, a, and c. Hint: It is better if you do not attempt to modify the rng module of ns-3; instead, create a function in your simulation file (e.g., mysimulation.cc) and call the function from the main.

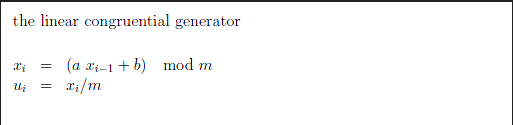


Figure 1: Linear congruential generator formula

2. Generate 1000 values uniformly distributed in the range [0,1] using your PRNG. For this case use m=100, a=13 c=1 and seed =1;

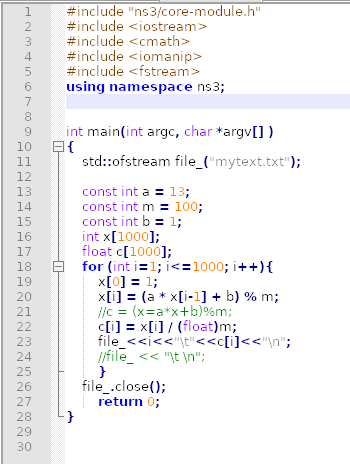


Figure 2: Code for Linear congruential generator

3. Compare the distribution of your values with the distribution of values generated using the UniformRandomVariable() of ns-3. Hint: To compare the distribution of values, you can use a histogram plot.

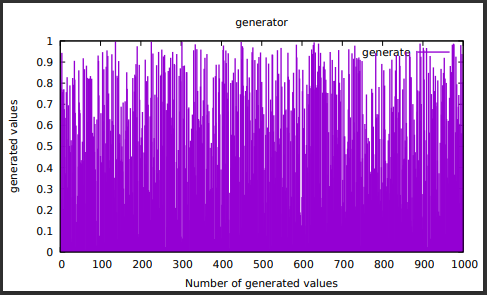
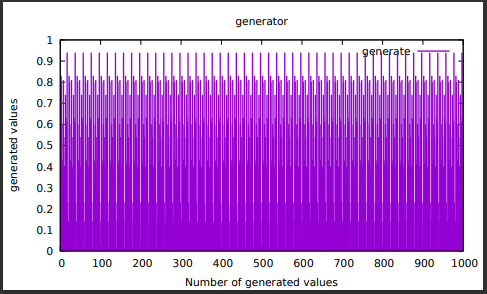
The LCG function generates 1000 values in a sequence of 20 values, so every 20 values it gets back to the start value (figure 3) and in the UniformRandomVariable() there is no visible sequence for 1000 generated values (figure 4). Comparing the two figures it looks much more random in the histogram for UniformRandomVariable() (figure 4) then the LCG (figure 3).

Figure : Histogram of LCG

Figure : Histogram of UniformRandomVariable()

4. Comment on the difference in the results and propose values of m, a, and c which gives you better results.

Because the linear congruential generator uses modulos, the m should be a prime so no matter what the value of a is x can’t be 0 if b also is an even number or 0.

The bigger the m-a is the bigger sequence we will get in the generation.

So, if b is equal to 0 and then we have a small number a and a big prime number for m, the output will be a much bigger sequence of random generated values then having a smaller m and a bigger value for a.

5. What PRNG does ns-3 use? What method does ns-3 use to generate a normal random variable?

*ns-3* contains a built-in pseudo-random number generator (PRNG); each RandomVariableStream used in *ns-3* has a virtual random number generator associated with it; all random variables use either a fixed or random seed based on the use of the global seed.

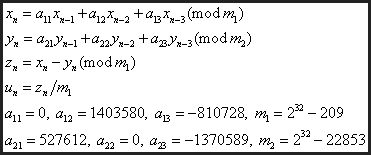
The PRNG method for ns-3 is the MRG32k3a generator from Pierre L’Ecuyer and it’s a multiple recursive generator combined with two components of order 3.

Figure : Formula for MRG32k3a

6. Using the time system command of Linux compare the execution time for the generation of the uniform distribution using your function and ns-3 function.

Mysimulation is the code for LCG and newa is the UniformRandomVariable() in figure 6.

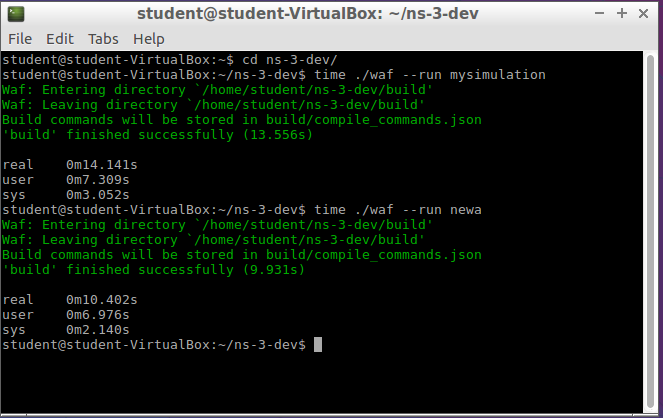


Figure 6: Time for running LCG and UniformRandomVariable()

7. Write a second function that generates an exponential distribution with mean 𝛽 > 0 from a uniform distribution generated using the LCG; Choose one of the methods for generating RV covered in the course and motivate your choice with respect to the specific task.

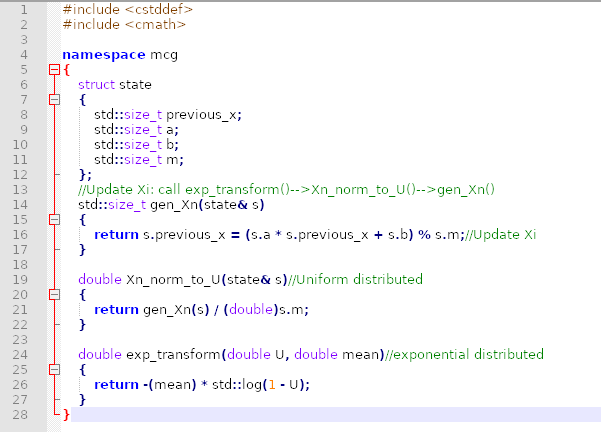


Figure 7: MCG.h for Exponential distribution

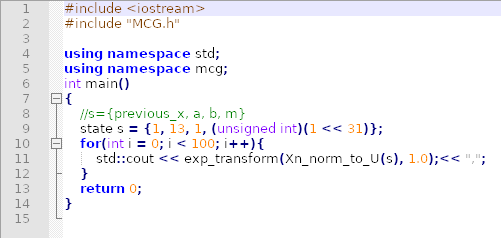


Figure 8: Main.cc for Exponential distribution

8. Compare your exponential distribution with ns-3 ExponentialRandomVariable()and the theoretical expression of the probability density function.

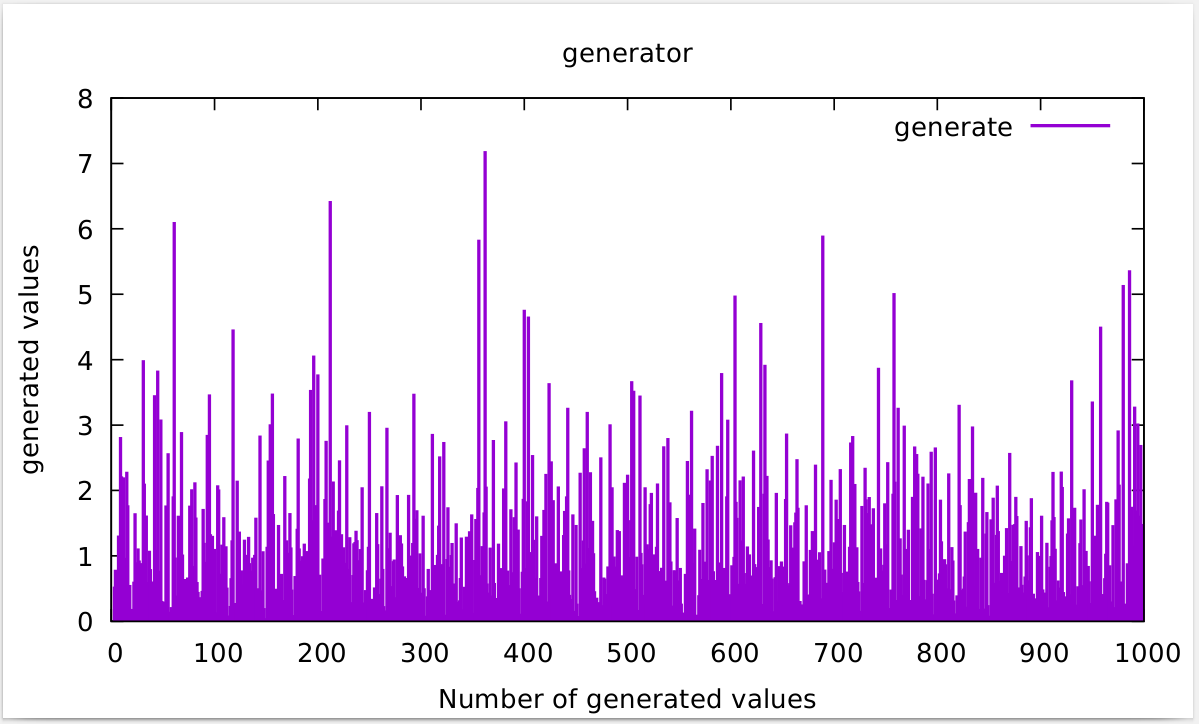
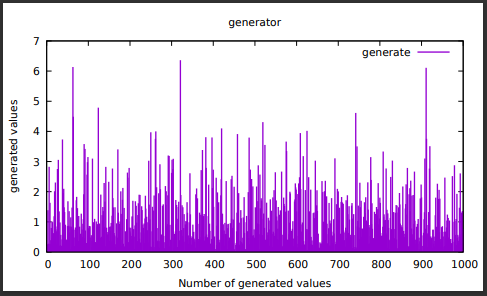


Figure :ExponentialRandomVariable()

Figure 10: Histogram of Exponential distribution from LCG