

## Lab # 2 - Generation of Random Variates (Implementation in C – 1 lab session)

### 1) Find “Matsumoto Home page” and the last C implementation of the original Mersenne Twister (MT)

Download and test the newest version at present of **MT 19937** (introduced in 1997 and revised at the beginning of the millennium). Untar and unzip the archive (“`tar -zxvf yourfile.tgz`”) and use the example. Compare the result you obtain locally on your computer session with the expected output (reproducibility). From now, always use a fine generator like MT or other very good generators.

### 2) Generation of uniform random numbers between A and B

Using a pseudo-random number generator like MT providing numbers between [0..1], propose a C function named “uniform” with 2 parameters ‘a’ and ‘b’ (real numbers) and generate pseudo-random numbers between ‘a’ and ‘b’.

### 3) Reproduction of discrete empirical distributions

Suppose we have field data giving the distribution probability of 3 species (A, B and C):

50% for A, 10% for B et 40% for C.

To reproduce (simulate) a population of individuals with the same distribution, we can use the following procedure using a uniform pseudo-random number generator between 0 and 1.

When we draw a random number, if this number is strictly below 0.5, we can consider that the individual is of the A category (we will call it class A for the A species), if the number is between 0.5 and 0.6 (strictly), the individual is considered of the B class and if the number drawn is above 0.6, the individual is set in the C class.

Figure 1, below presents the histogram corresponding to this situation.

In practice we build an array with cumulated probabilities (giving the repartition function):

Array [1] = 0.5

Array [2] = 0.5 + 0.1 = 0.6

Array [3] = 0.5 + 0.1 + 0.4 = 1

The particular case of a 2 value histogram is similar to a biased coin tossing simulation.

- Implement a function to reproduce the histogram given below with 1 000, and 1 000 000 drawings. Cumulate the number of individual of each species in 3 variables and display the percentage obtained.
- Implement another function with the following input: the size of an array of classes, then the array itself with the number of individuals in each class. Compute the corresponding array with the probability of being in each class (distribution function) and then compute another array giving the

cumulated probabilities (repartition function). This function outputs the latter array. Test this function with your own data and check a simulated distribution with 1000 and 1000 000 drawings.

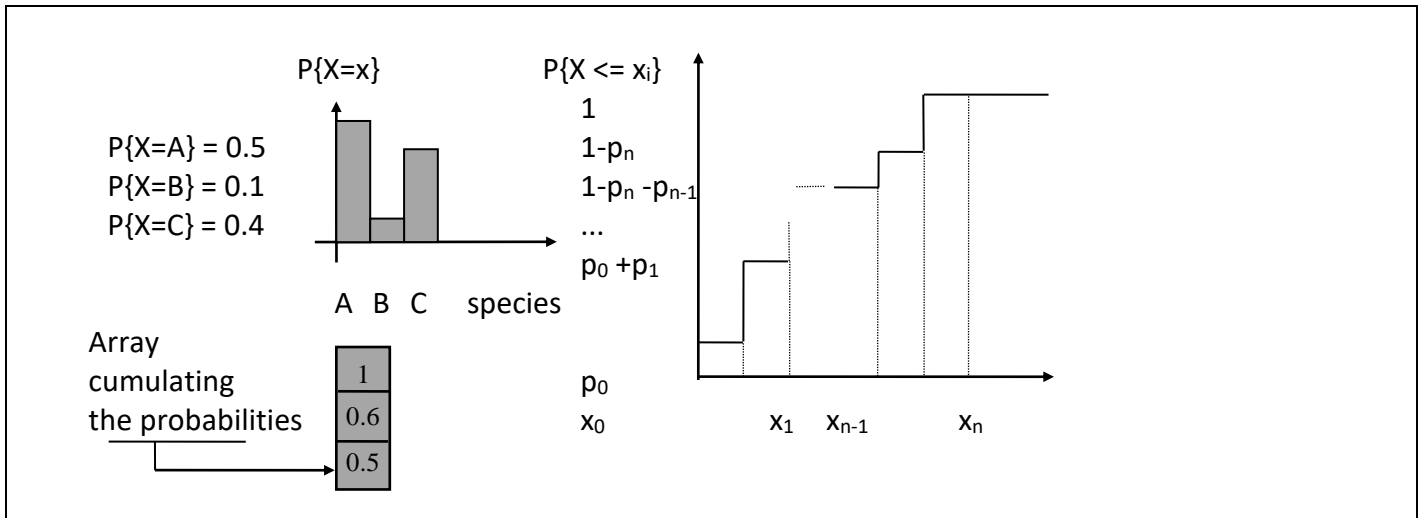


Figure 1. Sample histogram for 3 species

#### 4) Reproduction of continuous distributions

It is possible to reproduce continuous distributions by inverting the distribution function. When drawing a pseudo-random number between 0 and 1, it is possible to obtain a number distributed according to a given continuous distribution function (F) supposing that the latter is reversible.

$$x = F^{-1} (\text{Random number drawn})$$

This technique, named anamorphosis is not completely generic, but we it can be applied to many distribution laws (Negative exponential law, uniform law, Weibull law. For instance, the distribution function of a negative exponential law is given in equation (8) leading to the inverse law of equation (9) which has to be implemented.

$$F(x) = \int_0^x \frac{1}{M} e^{-\frac{1}{M}z} dz = 1 - e^{-\frac{1}{M}x} \quad (8)$$

$$\begin{aligned} \text{RandomNumberDrawn} &= 1 - e^{-\frac{1}{M}x} \\ \Rightarrow 1 - \text{RandomNumberDrawn} &= e^{-\frac{1}{M}x} \\ \Rightarrow \ln(1 - \text{RandomNumberDrawn}) &= -\frac{1}{M}x \\ \Rightarrow x &= -M \ln(1 - \text{RandomNumberDrawn}) \end{aligned} \quad (9)$$

Uniform law between A and B	: $x = F^{-1}(\text{Random number drawn}) = A + (B - A) * \text{Random number drawn}$
Mean = $(B + A) / 2$	Variance = $1/12 * (B - A)^2$
Negative exponential law(Average)	: $x = F^{-1}(\text{Random number drawn}) = - \text{Mean} * \text{Log}(1 - \text{Random number drawn})$
Mean = M	Variance = M

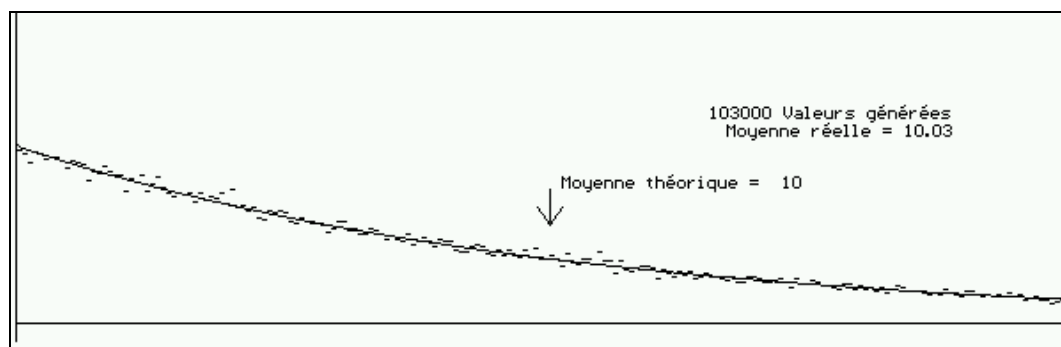
*Figure 2. Inverse function of the uniform and negative exponential law*

For this question we have to generate 1000 (then 1000 000) drawings according to a negative exponential distribution with a mean equal to 10. This supposes using random numbers between 0 and 1 in equation (9) to obtain the correct distribution. Such numbers could for instance correspond to inter-arrival time between two jobs submitted to a computing cluster.

Use an array with 20 bins and test the frequency of numbers between 0 and 1, between 1 and 2,... and between 19 and 20. For each number drawn, count in which bin it appears and cumulate this for all your drawings (1 000, 1 000 000)

```
Test20bins[ (int) negExp(10) ] ++;
```

With a Mean set to 10, you will produce many numbers between 0 and 1, a bit less between 1 and 2, etc. If you display an histogram, it can produce something that looks like figure 3 (with a different slope).



*Figure 3. Simulation of the Inverse function of the uniform and negative exponential law.*

## 5) Simulating non reversible distribution laws

In the case of non reversible distribution laws, we use the rejection technique which is a Monte Carlo inspired technique. Below is a standard rejection algorithm for generating a number according to a probability distribution  $f(x)$  between 2 values MinX and MaxX (+ Min Y and MaxY which are the values providing a box around the probability distribution (density) function (PDF).

- (1) Generate 2 random numbers  $Na_1$  and  $Na_2$
- (2) Compute  $X = \text{MinX} + Na_1 * (\text{MaxX} - \text{MinX})$
- (3) Compute  $Y = \text{MaxY} * Na_2$
- (4) If  $Y \leq f(X)$   
Then  $X$  is considered as distributed according a law with  $f(x)$  as density function  
Else reject  $X$  and goto (1) ie : draw again 2 pseudo-random numbers between 0 and 1, etc..  
EndIf

Figure 4. Generic rejection algorithm

#### The Special case of the Gaussian distribution:

The density of a normal law (reduced and centred : average = 0, standard deviation = 1) is noted  $N(0,1)$  and given by (10) hereafter.

$$p(x) = \frac{1}{\sqrt{(2\pi)}} e^{-\frac{x^2}{2}} \quad (10)$$

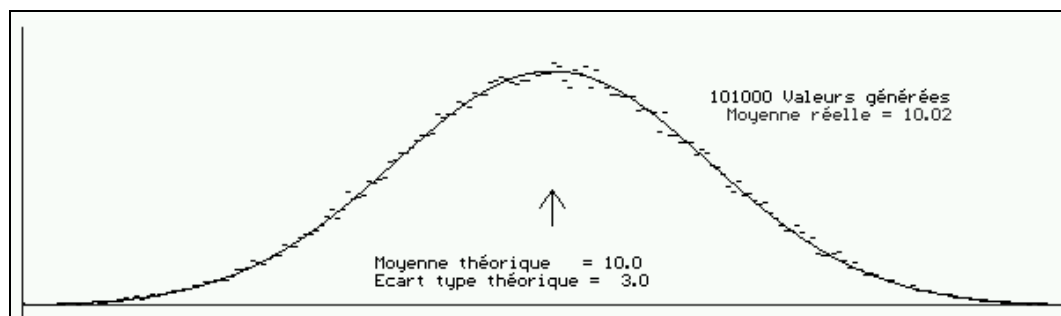


Figure 5. Generation of valued following a Gaussian distribution (average = 10, std. Dev. = 3)

In 1958, Box and Muller presented an exact method without using the Central Limit theorem and using two pseudo random numbers. Equation (14) uses two random numbers  $Rn_1$  and  $Rn_2$  and produces two numbers distributed on both sides of the centred and reduced Gaussian law -  $N(0,1)$ . Many variants exist to approximate a Gaussian distribution, some are faster, some more precise...

$$\begin{aligned} x_1 &= \cos(2\pi Rn_2) (-2 \ln(Rn_1))^{\frac{1}{2}} \\ x_2 &= \sin(2\pi Rn_2) (-2 \ln(Rn_1))^{\frac{1}{2}} \end{aligned} \quad (14)$$

Find a technique to simulate 1000 drawings according to a Gaussian distribution of average 10 and with a standard deviation at 3.

**6) Find libraries in C/C++ and Java that generate random variates like you did in the previous questions.**