SIMULATIONS OF CONTINUOUS RANDOM VARIABLES AND MONTE CARLO METHODS

Sanda Micula [1*]
Ioana D. Pop [2]

ABSTRACT

In this paper we describe algorithms for computer simulations of some common continuous distributions and their implementation in MATLAB. We use Monte Carlo methods for estimating probabilities and other characteristics of random variables. The paper concludes with some interesting applications.

AMS SUBJECT CLASSIFICATION: *60E05, 60G99, 60J10, 65C05, 65C60.*
KEYWORDS: *continuous random variables, computer simulations, Monte Carlo methods, MATLAB.*

1. INTRODUCTION

Monte Carlo methods are used to describe any technique that approximates solutions to quantitative problems through statistical sampling. This process involves performing many simulations using random numbers and probability to get an approximation of the answer to a problem which is otherwise too complicated, expensive, time consuming, dangerous, or simply impossible to solve analytically. Such methods use approximations which are based on "long run" simulations. With the help of random number generators, computers can actually simulate a "long run". The longer the run is simulated, the more accurate the predictions are. Monte Carlo methods can be used for (but are not restricted to) computation of probabilities, expected values and other distribution characteristics.

Although we briefly discuss simulations of some discrete distributions, the main focus of this paper is to present methods of simulation for continuous random variables and their applications.

1.1. Preliminaries

Definition 1.1. The set of all possible outcomes of an experiment is called the **sample space** of that experiment and is denoted by S. Its elements are called elementary events. An **event** is a collection of elementary events, i.e. a subset of S.

Definition 1.2. A collection of events $K \subseteq S$ is called a σ-**field** (or σ-**algebra**) on the sample space S, if it satisfies the conditions

[1]* corresponding author, Lecturer PhD., Department of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania, smicula@math.ubbcluj.ro
[2] Professor PhD., Department of Land Measurements and Exact Sciences, University of Agricultural Sciences and Veterinary Medicine, Cluj-Napoca, Romania, popioana@usamvcluj.ro

$(i)\ K \neq \emptyset;$

$(ii)\ A \in K \implies \bar{A} \in K;$

$(iii)\ A_1, A_2, \dots, A_n \in K \implies \bigcup_{i=1}^{n} A_i \in K.$

Definition 1.3. Let S be a sample space and $K \subseteq S$ a σ-field on it. **Probability** is a function $P: K \to R$ satisfying the conditions

$(i)\ P(S) = 1;$

$(ii)\ P(A) \geq 0, \forall A \in K;$

$(iii)\ P(A_1 \cup A_2 \cup \dots) = P(A_1) + P(A_2) + \cdots$ for any finite or countably infinite collection of mutually exclusive (disjoint) events in K.

Then (S, K, P) is called a **probability space**.

Definition 1.4. A random variable is a function $X: S \to \mathbb{R}$ for which the inverse image $X^{-1}\big((-\infty, x]\big) = \{e \in S | X(e) \leq x\} \in K$, for all $x \in \mathbb{R}$.

If $X(S)$ is a most countable in $\mathbb{R}$, then X is a **discrete random variable**.

Definition 1.5. Let X be a (discrete or continuous) random variable. The function $F: \mathbb{R} \to \mathbb{R}$ given by

$$F(x) = P(X \leq x) \tag{1.1}$$

is called the **cumulative distribution function (cdf)** of X.

Definition 1.6. Let X be a random variable with F. If there exists a function $f: \mathbb{R} \to \mathbb{R}$ such that

$$F(x) = \int_{-\infty}^{x} f(t)dt, \tag{1.2}$$

then X is a **continuous random variable** and f is called its **probability density function (pdf)**.

If X is a discrete random variable, then a better way of describing it is to give its **probability distribution function** (**also pdf**), an array that contains all its values x_i, and the corresponding probabilities with which each value is taken $p_i = P(X = x_i)$,

$$X \begin{pmatrix} x_i \\ p_i \end{pmatrix}_{i \in I}. \tag{1.3}$$

So, either way (the discrete or the continuous case), the pdf is what describes a random variable. Although in both cases we call it generically a pdf, the word distribution emphasizes a discrete behavior, whereas density suggests a continuous set. However, not all authors make the distinction between the two cases. In the case where X is a discrete random variable, it can be easily seen that

$$F(x) = \sum_{x_i \leq x} p_i, \tag{1.4}$$

hence the name.

1.2. Some Common Distributions

Although many more probability distributions have a wide variety of applications and the simulation methods we will discuss apply to all of them, we do not intend to make an exhaustive list of common distributions and, thus, only mention the ones that will be used in applications in the sequence.

Bernoulli distribution $Bern(p)$, with parameter $p \in (0,1)$. This is the simplest of distributions, with pdf

$$X\begin{pmatrix} 0 & 1 \\ 1-p & p \end{pmatrix}. \tag{1.5}$$

It is used to model "success/failure" (i.e. a Bernoulli trial), since many distributions are described in such terms.

Binomial distribution $B(n,p)$, with parameters $n \in \mathbb{N}$, $p \in (0,1)$. Consider a series of n Bernoulli trials with probability of success p in every trial ($q = 1-p$). Let X be the number of successes that occur in the n trials. Then X has a Binomial distribution, with pdf

$$X\begin{pmatrix} k \\ C_n^k p^k q^{n-k} \end{pmatrix}_{k=\overline{0,n}}. \tag{1.6}$$

Note that a Binomial $B(n,p)$ variable is the sum of n independent $Bern(p)$ variables and $Bern(p) = B(1,p)$.

Poisson distribution $P(\lambda)$, with parameter $\lambda > 0$, with pdf

$$X\begin{pmatrix} k \\ \frac{\lambda^k}{k!}e^{-\lambda} \end{pmatrix}_{k\in\mathbb{N}}. \tag{1.7}$$

Such a variable is defined in the context of a Poisson process: a process in which discrete events are observed in a continuous medium (length, aria, volume, time, etc.). Such events are called rare events, because they are extremely unlikely to occur simultaneously or within a short interval (of time, length, etc.). The Poisson variable X denotes the number of such rare events that occur in a given interval of the continuous medium. The parameter of the Poisson distribution, λ, represents the average number of the considered rare events per unit (of time, length, etc.). This distribution is used to model number of jobs in an interval of time, such as arrival of messages, earthquakes that happen in an area, errors found in software, traffic accidents, etc.

Next, let us recall some important continuous random variables. For their pdf's, we only mention their expression on the region where they are non-zero (meaning they are equal to 0 everywhere else).

Uniform distribution $\mathcal{U}(a,b)$, with parameters $a < b \in \mathbb{R}$, has pdf

$$f(x) = \frac{1}{b-a}, x \in (a,b). \tag{1.8}$$

It is used whenever a value is picked "at random" from an interval, in situations when all values from an interval are equally probable to be taken by a random variable.

Standard Uniform distribution $\mathcal{U}(0,1)$, with pdf

$$f(x) = 1, x \in (0,1) \text{ and } F(x) = \begin{cases} 0, & x < 0 \\ x, & x \in [0,1) \\ 1, & x > 1 \end{cases}. \quad (1.9)$$

Standard Uniform variables are particularly important in generating random variables with various distributions. If $U \in \mathcal{U}(0,1)$, then $X = a + (b-a)U \in \mathcal{U}(a,b)$.

Normal distribution $N(\mu, \sigma)$, with parameters $\mu \in \mathbb{R}$, $\sigma > 0$, with pdf and cdf

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, x \in \mathbb{R}, \quad F(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\sigma^2}} dt = \Phi\left(\frac{x-\mu}{\sigma}\right). \quad (1.10)$$

Standard (Reduced) Normal distribution $N(0,1)$, with pdf

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}, x \in \mathbb{R} \text{ and } cdf\ F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt = \Phi(x), \quad (1.11)$$

known as Laplace's function (or the error function), whose values can be found in tables.

Exponential distribution $Exp(\lambda)$, with parameter $\lambda > 0$, has pdf

$$f(x) = \lambda e^{-\lambda x}, x > 0 \text{ and } cdf\ F(x) = 1 - e^{-\lambda x}, x > 0. \quad (1.12)$$

An Exponential variable models time: waiting time, interarrival time, failure time, time between rare events, etc. The parameter λ represents the frequency of rare events, measured in time^{-1}. In fact, in a sequence of rare events, where the number of such occurrences in an interval of time of length t has $P(\lambda t)$ distribution, the time between rare events has $Exp(\lambda)$ distribution.

Expectation and variance

These are two of the most important numerical characteristics associated with random variables. The *expectation (expected value, mean value)* gives a "long term" average value, an equilibrium value of a random variable. Below are the computational formulas for a discrete random variable with pdf $X\begin{pmatrix} x_i \\ p_i \end{pmatrix}_{i \in I}$ and for a continuous random variable with pdf $f: R \to R$, respectively:

$$E(X) = \sum_{i \in I} x_i p_i, \quad E(X) = \int_{\mathbb{R}} x f(x) dx \ ' \quad (1.13)$$

The *variance (dispersion)* measures the spread in data, how much the values of a random variable vary from its mean value and so does its square root, the *standard deviation*.

$$V(X) = E\left(\left(X - E(X)\right)^2\right) = E(X^2) - \left(E(X)\right)^2, \ \sigma(X) = \sqrt{V(X)}. \quad (1.14)$$

Normal approximation of a Binomial distribution

For values of p that are not too extreme, say $p \in [0.05, 0.95]$ and for large values of $n \in N$, the Normal distribution can be used to approximate the Binomial distribution:

$$B(n,p) \approx N\left(\mu = np, \sigma = \sqrt{np(1-p)}\right). \quad (1.15)$$

This formula is especially useful in Statistics, when *quantiles* (inverses of the cdf) are needed.

Next, we recall one important case of two-dimensional random vectors, which will be used further on.

Uniformly distributed random vector (X,Y) over a region $D \subseteq \mathbb{R}^2$, is vector whose joint density is a constant over that region (and 0 everywhere else). Since the total (double) integral of that density $\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x;\, y)\, dxdy$ must be 1 (representing the probability of the sure event), the joint pdf of the vector is given by

$$f(x,y) = \frac{1}{\text{area}(D)}, (x,y) \in D\ . \quad (1.16)$$

From the joint pdf of a vector, $f_{(X;Y)}$, one can find the marginal densities (the pdf's of its components) by

$$f_X(x) = \int_{\mathbb{R}} f_{(X,Y)}(x,y)dy, \quad f_Y(y) = \int_{\mathbb{R}} f_{(X,Y)}(x,y)dx. \quad (1.17)$$

2. SIMULATIONS OF RANDOM VARIABLES AND MONTE CARLO METHODS

2.1. Inverse Transform Method

This is a method used when we want to generate a random variable whose cdf F does not have a very complicated form. It is based on the following result:

Theorem 2.1. *Let X be a continuous random variable with cdf* $F : \mathbb{R} \to \mathbb{R}$. *Then* $U = F(X) \in \mathcal{U}(0,1)$.

Proof. We will show that U has the $\mathcal{U}(0,1)$ pdf.

First off, let us notice that, being a cdf, $F(x) \in [0,1]$, for all $x \in \mathbb{R}$ and, thus, all the values of U are in $[0,1]$.

Secondly, X being a continuous random variable, there exists an interval $(a,b) \subseteq \mathbb{R}$ such that $F\colon (a,b) \to [0,1]$ is strictly increasing (therefore one-to-one), $F(x) = 0, \forall x \le a$ and $F(x) = 1, \forall x \ge b$.

Hence, its inverse $F^{-1}\colon [0,1] \to (a,b)$ exists.

Now, let us consider the cdf, F_U. Let $x \in \mathbb{R}$.

If $x < 0$, then $F_U(x) = P(U \le x) = P(\text{impossible event}) = 0$. Hence, $f_U(x) = F_U'(x) = 0$.

If $x > 1$, then $F_U(x) = P(U \le x) = P(\text{sure event}) = 1$ and thus, $f_U(x) = F_U'(x) = 0$.

For $x \in [0,1]$, we have

$$F_U(x) = P(U \le x) = P(F(X) \le x) = P\left(X \le F^{-1}(x)\right) = F\left(F^{-1}(x)\right) = x.$$

Then $f_U(x) = F_U'(x) = 1$ and $U \in \mathcal{U}(0,1)$.□

As a consequence, to generate a continuous random variable with given $cdf\ F$, we generate a variable $U \in \mathcal{U}(0,1)$ and let

$$X = F^{-1}(U). \tag{2.1}$$

Indeed, then the cdf of X is

$$F_X(x) = P(X \leq x) = P(F^{-1}(U) \leq x) = P\big(U \leq F(x)\big) = F_U\big(F(x)\big) = F(x).$$

For all $x \in \mathbb{R}$, the last assertion following from (1.9) and the fact that $F(x) \in [0,1]$. Thus X has the desired cdf F.

Remark 2.2. *The inverse transform method can be adjusted for discrete random variables as well, if we consider the generalized inverse of the cdf, i.e. let*

$$X = \min\{x \mid F(x) \geq U\}. \tag{2.2}$$

Algorithm 1.

1. Generate $\in \mathcal{U}(0,1)$.
2. if X is continuous, then let $X = F^{-1}(U)$.
3. if X is discrete, then let $X = \min\{x \in S \mid F(x) \geq U\}$, where S is a set of possible values of X.

Example 2.3. *Use the inverse transform method to generate a variable* $X \in Exp(\lambda)$, $\lambda > 0$.

By (1.12), we find the inverse $F^{-1}(x) = -\frac{1}{\lambda}\ln(1-x)$. Then, for $U \in \mathcal{U}(0,1)$ we generate

$$X_1 = -\frac{1}{\lambda}\ln(1-U)\ . \tag{2.3}$$

Now, since $U \in \mathcal{U}(0,1) \Leftrightarrow 1 - U \in \mathcal{U}(0,1)$ we can also use

$$X_2 = -\frac{1}{\lambda}\ln(U)\ . \tag{2.4}$$

Notice that since $U, 1-U \in \mathcal{U}(0,1)$, we have that both $\ln(U)\,, \ln(1-U) < 0$ and, thus, $X_1, X_2 > 0$, as they should be.□

2.2. Rejection Method

The previous method is inconvenient when the cdf F has a complicated expression and/or its inverse is difficult to find. We present next a method that uses the pdf f instead.

Theorem 2.4. *Let* $f: \mathbb{R} \to \mathbb{R}$ *be a pdf. Let the vector* (X, Y) *be Uniformly distributed over the region*

$$D = \{(x,y) \in \mathbb{R}^2 | 0 \leq y \leq f(x)\}\ . \tag{2.5}$$

Then X has pdf f, i.e. $f_X = f$.

Proof. First, let us determine the joint pdf of the vector (X, Y). By (1.16), it is

$$f_{(X,Y)}(x,y) = \frac{1}{area(D)}, \text{ for } (x,y) \in D$$

and 0 everywhere else. But, since f is a pdf, that area is $\int_{\mathbb{R}} f(x)dx = 1$.

So, the joint pdf of (X,Y) is

$$f_{(X,Y)}(x,y) = \begin{cases} 1, & (x,y) \in D \\ 0, & (x,y) \notin D. \end{cases}$$

Then, by (1.17), the (marginal) pdf of its first component is

$$f_X(x) = \int_{\mathbb{R}} f_{(X,Y)}(x,y)dy = \int_D dy = \int_0^{f(x)} dy = f(x). \quad (2.6)$$

Thus, X indeed has the function f as its pdf. □

To generate a variable with given pdf f, we generate points (X,Y) that are Uniformly distributed in D.

Algorithm 2.

1. Find numbers $a,b \in \mathbb{R}, c \in \mathbb{R}_+$ such that $f(x) \in [0,c]$ for $x \in [a,b]$ (this is always possible, since D is a bounded in $\mathbb{R}^2$, having an area of 1. The rectangle $[a,b] \times [0,c]$ is called a *bounding box*.
2. Generate $U,V \in \mathcal{U}(0,1)$.
3. Let $X = a + (b-a)U$ and $Y = cV$. Then $X \in \mathcal{U}(\mathrm{a},\mathrm{b})$, $Y \in \mathcal{U}(0,\mathrm{c})$ and $(X,Y) \in \mathcal{U}([\mathrm{a},\mathrm{b}] \times [0,\mathrm{c}])$.
4. If $Y > f(X)$, reject the point and return to step 2. If $Y \leq f(X)$, then X has the desired pdf, f.

The idea of the rejection method is displayed graphically in Figure 1.

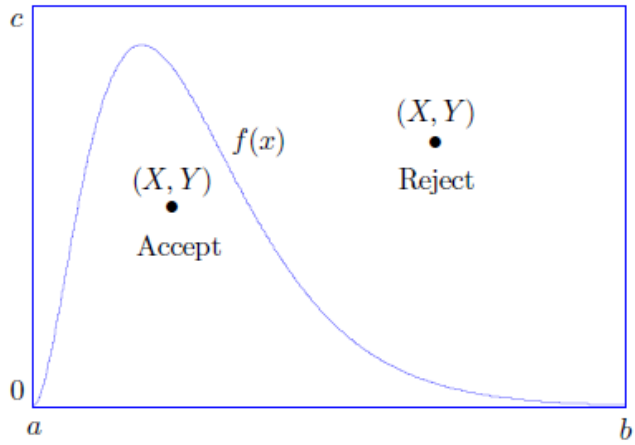

Figure 1. Rejection Method

2.3. Special Methods

These methods use specific properties of certain distributions. They are a good alternative of simulation, when the more general methods presented so far, are too complicated to implement.

There is a large number of such methods, both for discrete (see e.g. [4]) and continuous random variables. We only present a few.

Generation of a Poisson random variable, $P(\lambda), \lambda > 0$, whose pdf is given by (1.7).

We use the fact that such a variable counts the number of "rare" events that occur during one unit of time and the fact that the time elapsed between any two such events has Exponential distribution, which can be generated by (2.4), using the inverse transform method. So, each such time is generated by $T_i = -\frac{1}{\lambda}\ln(U_i)$, for $U_i \in \mathcal{U}(0,1)$ and then we count the number of events that occurred in one unit of time:

$$X = \max\{n \mid T_1 + \cdots + T_n \leq 1\}, \text{ i.e. } X = \max\{n \mid U_1 \cdot \ldots \cdot U_n \geq e^{-\lambda}\}. \quad (2.7)$$

Algorithm 3.

1. Generate $U_1, U_2, \ldots \in \mathcal{U}(0,1)$
2. Let $X = \max\{n \mid U_1 \cdot U_2 \cdot \ldots \cdot U_n \geq e^{-\lambda}\}$.

Generation of a Normal random variable, $N(\mu, \sigma)$, $\mu \in \mathbb{R}$, $\sigma > 0$, whose pdf is given by (1.10). We present an algorithm for generating Normal Variables that uses two-dimensional vectors, but omit the details, as they are too technical.

Box-Muller transform

Algorithm 4.

1. Generate $U, V \in \mathcal{U}(0,1)$.
2. Let

$$\begin{cases} Z_1 = \sqrt{-2\ln(U)}\cos(2\pi V), \\ Z_2 = \sqrt{-2\ln(U)}\sin(2\pi V). \end{cases}$$

Then Z_1, Z_2 are independent $N(0,1)$ random variables.

3. Let $X = \sigma Z + \mu$ (for either Z from above). Then $X \in N(\mu, \sigma)$.

2.4. Accuracy of a Monte Carlo Study

Now, using the methods of simulation presented so far, we perform a Monte Carlo study, meaning that we put the chosen algorithm in a loop and simulate a "long run", i.e. generate a number of such variables, $X_1, \ldots, X_N$.

Recall from Statistics that when a parameter is approximated by an estimator (a function of sample variables) $\bar{\theta}$, a desired quality of that estimator is to be *unbiased*, i.e. that

$$E(\bar{\theta}) = \theta, \tag{2.8}$$

so that, in the long run, we know its values will stabilize at the right point. We also want that its variance $V(\bar{\theta})$ be small, approaching 0, as the simple size $N \to \infty$.

Estimating probabilities, means and variances

We estimate probabilities by long run relative frequencies. For a random variable X, we generate variables $X_1, \dots, X_N$ with the same distribution and approximate $p = P(X \in A)$ by

$$\bar{p} = \frac{\text{number of } X_1, \dots, X_N \in A}{N}. \tag{2.9}$$

The mean value $E(X) = \mu$, the variance $V(X) = \sigma^2$ and the standard deviation $\sigma = \sqrt{V(X)}$ of a random variable X estimated by

$$\bar{X} = \frac{X_1 + \cdots + X_N}{N}$$
$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2, \quad s = \sqrt{s^2}, \tag{2.10}$$

respectively. Since the simulations are independent, the number at the numerator in (2.9) has Binomial $B(N,p)$ distribution and, hence, expected value Np and variance $Np(1-p)$. Then, we have

$$E(\bar{p}) = \frac{1}{N}Np = p,$$
$$V(\bar{p}) = \frac{1}{N^2}Np(1-p) = \frac{p(1-p)}{N} \tag{2.11}$$

Thus, $\bar{p}$ is an unbiased estimator for p and its standard deviation $\sigma(\bar{p}) = \sqrt{\frac{p(1-p)}{N}}$ decreases with N at the rate of $1/\sqrt{N}$.

The same is true for the estimators in (2.10), but we omit the details.

Accuracy of a Monte Carlo study

When we conduct a Monte Carlo study, the question arises about its size. What would be a suitable size in order to get a certain accuracy? Given a tolerable error $\varepsilon > 0$ and a significance level (probability of error) $\alpha \in (0,1)$, we want to determine the size N so that

$$P(|\bar{p} - p| > \varepsilon) \le \alpha. \tag{2.12}$$

By (1.15), for moderate values of p, we have that $\frac{N\bar{p} - E(N\bar{p})}{\sqrt{V(N\bar{p})}} = \frac{\bar{p}-p}{\sqrt{\frac{p(1-p)}{N}}} \approx N(0,1)$. Then

$$P(|\bar{p} - p| > \varepsilon) = P\left(\frac{|\bar{p}-p|}{\sqrt{\frac{p(1-p)}{N}}} > \frac{\varepsilon}{\sqrt{\frac{p(1-p)}{N}}}\right) = 2\Phi\left(-\frac{\varepsilon\sqrt{N}}{\sqrt{p(1-p)}}\right),$$

where Φ is Laplace's function (the cdf of a $N(0,1)$ variable) described in (1.11).

Still, this contains the unknown value p. We can manage that using the fact that for any $p \in (0,1)$, $p(1-p) \leq \frac{1}{4}$ and, thus, $\Phi\left(-\frac{\varepsilon\sqrt{N}}{\sqrt{p(1-p)}}\right) \leq \Phi(-2\varepsilon\sqrt{N})$. Then to ensure (2.12), we take $\Phi\left(-2\varepsilon\sqrt{N}\right) \leq \alpha/2$, i.e.

$$N \geq \frac{1}{4}\left(\frac{z_{\alpha/2}}{\varepsilon}\right)^2, \tag{2.13}$$

where $z_{\alpha/2}$ is the quantile (inverse of the cdf Φ) of order $\alpha/2$ for the $N(0,1)$ distribution.

3. APPLICATIONS

Let us start by implementing in Matlab some of the examples discussed earlier.

Example 3.1. *A Poisson variable $P(\lambda), \lambda > 0$, using Algorithm 3.*

The implementation of Algorithm 3, is given below

```
lambda = input ('lambda ( > 0) = '); % the parameter
err = input ('error = ' ); % maximum error
alpha = input ('alpha (level of significance) = ');
N =  ceil (0.25*(norminv (alpha/2,0,1)/err) ^2); % MC size
for j = 1 : N
 U = rand; % generated U(0,1) variable
 X(j)  = 0;  % initial value
 while U >= exp (- lambda)
  U = U * rand; % go further to n + 1 (i.e. X + 1)
  X(j) = X(j) + 1; % the Poisson variable
 end
end
clf % Compare it to the Poisson distribution, graphically.
k = 0 : 25; % values for the Poisson distr.
p_k = poisspdf (k, lambda); % probabilities of a Poiss distr.
UX = unique (x); % the values of X listed ONLY ONCE
n_X = hist (X,length(UX)); % the freq. of each value in UX
plot(UX,n_X/N,'*',k,p_k, 'ro','Markersize',7,'LineWidth',2)
legend ('simulation', 'Poisson distr',0)
```

A graphical comparison of the pdf's is shown in Figure 2., for $\lambda = 5, \varepsilon = 1e-3$ and $\alpha = 0.05$.

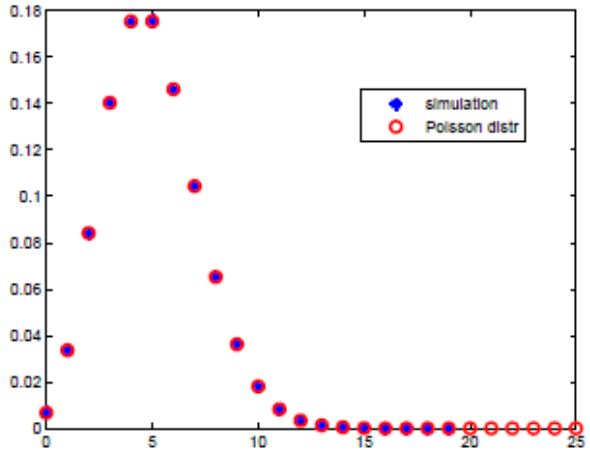

Figure 2. Poisson distribution

Example 3.2. *An Exponential variable* $Exp(\lambda), \lambda > 0$, *using the inverse transform method.*

An implementation of the algorithm in Example 2.3 is given below.

```
lambda = input('lambda ( > 0) = '); % the parameter
err = input ('error ( < lambda) = ');% maximum error
alpha = input ('alpha (level of significance) = ');
N = ceil(0.25*(norminv(alpha/2,0,1)/err)^2); % MC size
for j = 1 : N
 X(j) = -1/lambda*log(rand); % the Exp variables
end
clf
% Compare it to the Exp(1/lambda) distr. (from Matlab), graphically.
x = -0.1 : 0.01 : 1/lambda* log(lambda/err);
cdfx = expcdf (x, 1/lambda); % the cdf of an Exp distr.
for i = 1: length(x)
 mycdf(i) = mean(X < x(i)); % cdf of the simulation
end
plot (x,cdfx, x, mycdf, 'r: ', 'LineWidth', 2)
legend('cdf of Exp distr', 'cdf of simulation',0)
```

A graphical comparison of the cdf's is shown in Figure 3., for $\lambda = 4$, $\varepsilon = 1e - 3$ and $\alpha = 0.05$.

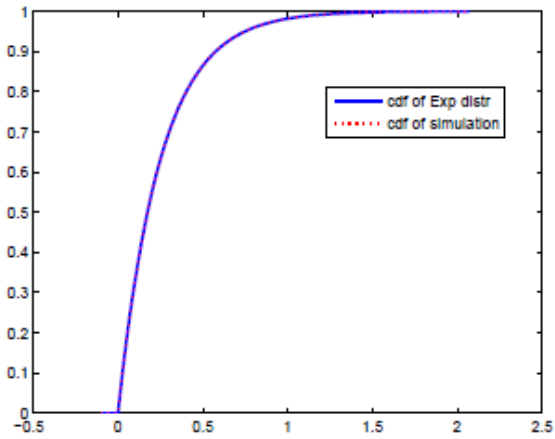

Figure 3. Exponential distribution, inverse transform method

Example 3.3. Aplication: Forecasting for new software release

An IT company is testing a new software to be released. Every day, software engineers find a random number of errors and correct them. On each day t, the number of errors found, X_t, has a Poisson (λ_t) distribution, where the parameter λ_t is the lowest number of errors found during the previous k days,

$$\lambda_t = \min\{X_{t-1}, X_{t-2}, \dots, X_{t-k}\}.$$

If some errors are still undetected after tmax days (i.e. if not all errors are found in tmax days), the software is withdrawn and goes back to development. Generate a Monte Carlo study to estimate

a) *the time it will take to find all errors;*
b) *the total number of errors found in this new release;*
c) *the probability that the software will be sent back to development.*

This problem does not have a simple analytic solution, therefore we use Monte Carlo methods to solve it. Below is the algorithm that gives the desired estimates.

```
% Forecasting errors in new software release.
err = input ('error = '); % maximum error
alpha = input('alpha (level of significance) = '); % level of sign.
N = ceil(0.25*(norminv(alpha/2,0,1)/err)^2); % MC size
fprintf('Nr. of simulations N = %d \n', N)
k = input('number of previous days considered = ');
inlastk = input('numbers of errors in the last k days ...
 (vector of length k) = '); % initial. nr of errors
tmax = input ('max time (in days) = ');
% Ttotal is the time it takes to find all the errors (in days)
% Ntotalerr is the total number of errors that are detected
for j = 1 : N;
% T is time from now (in days), X is nr. of errors on day T
% nrerr is the number of errors detected so far
T = 0;
X = inlastk (k);
```

```
nrerr = sum(inlastk);
lastk = inlastk; % number of errors in the last k days
i=0;
while X>0; % while loop until no errors are found
 lambda = min(last); % par. for var X, Poisson
% Simulate the nr of errors on day T, Poisson (lambda)      U = rand; %
generated U(0,1) variable
X = 0; % initial value
while U >= exp(- lambda);
 U = U * rand;
 X = X +1; % the Poisson variable
end;
T = T +1; % next day
nrerr = nrerr + X; % new nr. of errors
last = [last(2:k), X]; % new nrs of errors last k
end;
% the while loop ends when X = 0 on day T, that means that all errors
were found on previous day, T - 1
Ttotal (j) = T - 1; % the day all errors were found
 Ntotalerr(j) = nrerr;
end
disp([mean(Ttotal), mean (Ntotalerr), mean (Ttotal > tmax)])
```

Several runs of this algorithm for $\varepsilon = 5e-3$, $\alpha = 0.01$, $k = 4, [X_{t-1}, X_{t-2}, X_{t-3}, X_{t-4}] = [10, 5, 7, 6]$ and $tmax = 10$, give the time to find all errors approximately 7 days, the total number of errors around 53 and the probability that the software will be sent back to development about 0.18.

REFERENCES

[1] C. Andrieu, A. Doucet, R. Holenstein, *Particle Markov chain Monte Carlo methods*, J. Royal. Statist. Soc. B. Vol. 72(3), 2010, 269-342.

[2] M. Baron, *Probability and Statistics for Computer Scientists*, 2nd Edition, CRC Press, Taylor & Francis, Boca Raton, FL, USA, 214.

[3] S. Micula, *Probability and Statistics for Computational Sciences*, Cluj University Press, 2009.

[4] S. Micula, *Statistical Computer Simulations and Monte Carlo Methods*, J. of Information Systems and Operations Management, Vol. 9(2), 2015, 384-394.

[5] J.S. Milton, J. C. Arnold, *Instruction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences*, 3rd Edition. McGraw-Hill, New York, 1995.

[6] http://www.mathworks.com/help/matlab/,2015.