Introduction to Discrete Event Simulation

Desempenho e Dimensionamento de Redes

Prof. Amaro de Sousa (asou@ua.pt)

DETI-UA, 2020/2021

Discrete event simulation

A discrete event simulation models the operation of a system whose state changes with events that happen in discrete time instants:

- each event might force a change of the system state;
- between consecutive events, the system remains in the same state;
- thus, the simulation can directly jump in time from one event to the next event.

Elements of a discrete event simulator:

(1) State variables: describe the state of the system at any time instant

(2) Statistical counters: variables that store the appropriate statistical data related with the performance of the system

(3) Simulation clock: variable indicating the current simulated time instant (simulated time $\neq$ computation time)

(4) Events: types of occurrences that change either the system state and/or the statistical counters

(5) Event list: list of future events, their time instants and associated parameters

Besides these elements, additional supporting variables might be required.

Basic structure of a discrete event simulator

A discrete event simulator is mainly composed by the following steps:

(1) Initialization of the state variables, the statistical counters and the event list with the first event(s);

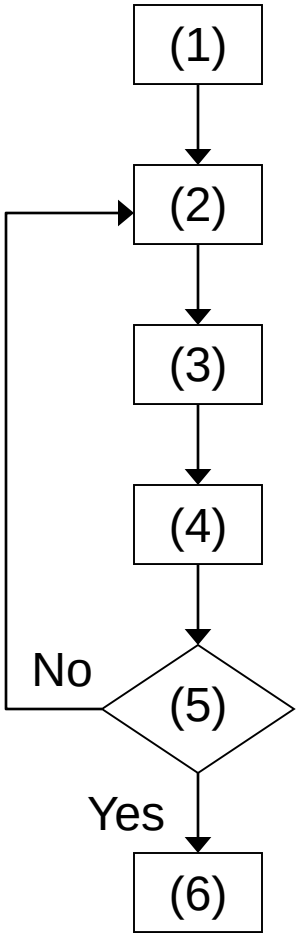
(2) Determination of the next event from the event list;

(3) Update of the simulation clock to the time instant of the event and removal of the event from the event list;

(4) Execution of all actions associated to the event (generation of new events, update of state variables and/or statistical counters);

(5) Check if the simulation must end; if not, return to Step (2);

(6) Update of the statistical counters and determination of the performance parameters.



Example: performance of a video-streaming service with a single server

Input parameters of simulation:

(1) Time between movie requests: exponential distributed variable with an average of 60 minutes

(2) Movie duration: exponential distributed variable with an average of 60 minutes

(3) Server capacity = *2* movies

Stopping criteria of simulation:

Time instant of the 4th movie request (this request counts for the statistical counters)

Performance parameters to be estimated by the simulation:

(1) Blocking probability: percentage of blocked movie requests (i.e., not provided due to server capacity)

(2) Average server utilization (in number of movies)

Example: performance of a video-streaming service with a single server

Events:

ARRIVAL: the request of a movie

DEPARTURE: the end of a movie transmission

State variables:

STATE: number of movies in transmission

Statistical counters:

OCUPATION: integral of the server occupation (in number of movies) from the beginning of the simulation until the current time instant

BLOCKED: number of blocked movie requests from the beginning of the simulation until the current time instant

REQUESTS: number of movie requests from the beginning of the simulation until the current time instant

Instant *t* = 0,0

Beginning of simulation

EVENT LIST
ARRIVAL - 63 min

Simulation Clock: 0

State Variable:

STATE
0

Statistical Counters:

OCUPATION
0

BLOCKED
0

REQUESTS
0

Instant *t* = 63 (First ARRIVAL)

EVENT LIST
ARRIVAL – 63 min
ARRIVAL - 117 min
DEPARTURE - 153 min

Simulation Clock: 63

State Variable:

STATE
1

$\leftarrow 0 + 1$

Statistical Counters:

OCUPATION
0

$\leftarrow 0 + 0 \times (63 - 0{,}0)$

BLOCKED
0

$\leftarrow 0 + 0$

REQUESTS
1

Instant t = 117 (Second ARRIVAL)

EVENT LIST
ARRIVAL - 63 min
ARRIVAL - 117 min
ARRIVAL - 150 min
DEPARTURE - 153 min
DEPARTURE - 207 min

Simulation Clock: 117

State Variable: STATE 2 $\leftarrow 1 + 1$

Statistical Counters:
OCUPATION 54 $\leftarrow 0 + 1 \times (117 - 63)$
BLOCKED 0 $\leftarrow 0 + 0$
REQUESTS 2

Instant *t* = 150 (Third ARRIVAL)

EVENT LIST
ARRIVAL - 63 min
ARRIVAL - 117 min
ARRIVAL - 150 min
DEPARTURE - 153 min
ARRIVAL - 204 min
DEPARTURE - 207 min

Simulation Clock: 150

State Variable: STATE 2 $\leftarrow 2 + 0$

Statistical Counters:
OCUPATION 120 $\leftarrow 54 + 2 \times (150 - 117)$
BLOCKED 1 $\leftarrow 0 + 1$
REQUESTS 3

Instant *t* = 153 (First DEPARTURE)

EVENT LIST
ARRIVAL - 63 min
ARRIVAL - 117 min
ARRIVAL - 150 min
DEPARTURE - 153 min
ARRIVAL - 204 min
DEPARTURE - 207 min

Simulation Clock: 153

State Variable: STATE 1 ← 2 – 1

Statistical Counters: OCUPATION 126 ← 120 + 2 × (153 – 150)

BLOCKED 1

REQUESTS 3

Instant *t* = 204 (Fourth ARRIVAL)

End of Simulation

Simulation Clock: 204

State Variable: STATE 2 ← 1 + 1

Statistical Counters:

OCUPATION 177 ← 126 + 1 × (204 – 153)

BLOCKED 1 ← 1 + 0

REQUESTS 4

EVENT LIST

ARRIVAL - 63 min

ARRIVAL - 117 min

ARRIVAL - 150 min

DEPARTURE - 153 min

ARRIVAL - 204 min

DEPARTURE - 207 min

Blocking probability:

BLOCKED / REQUESTS = 1/4 = 25%

Average server utilization:

OCUPATION / *t* = 177/204 = 0,87 movies

MATLAB simulation of a video-streaming service with a single server

```
function [b o]= simulator1(lambda,invmiu,C,M,R)
    invlambda=60/lambda;
    ARRIVAL= 0;
    DEPARTURE= 1;
    STATE= 0;
    OCUPATION= 0;
    REQUESTS= 0;
    BLOCKED= 0;
    Clock= 0;
    EventList= [ARRIVAL exprnd(invlambda)];
    while REQUESTS < R
        event= EventList(1,1);
        Previous_Clock= Clock;
        Clock= EventList(1,2);
        EventList(1,:)= [];
        OCUPATION= OCUPATION + STATE * (Clock - Previous_Clock);
        if event == ARRIVAL
            EventList= [EventList; ARRIVAL, Clock + exprnd(invlambda)];
            REQUESTS= REQUESTS + 1;
            if STATE + M <= C
                STATE= STATE + M;
                EventList= [EventList; DEPARTURE, Clock + exprnd(invmiu)];
            else
                BLOCKED= BLOCKED + 1;
            end
        else
            STATE= STATE - M;
        end
        EventList= sortrows(EventList,2);
    end
    b= BLOCKED/REQUESTS;
    o= OCUPATION/Clock;
end
```

```
INPUT PARAMETERS:
lambda: movie request rate (requests/hour)
invmiu: average movie duration (minutes)
C:      server capacity (in Mbps)
M:      throughput of each movie (in Mbps)
R:      no. of requests to stop the
        simulation
```

Generation of random numbers with a uniform distribution between 0 and 1

A Linear Congruential Generator (LCG) is an algorithm that yields a sequence of randomized numbers calculated with a linear equation.

The method represents one of the oldest and best-known pseudorandom number generator algorithms.

Generation method:

(1) Generate integer values Z_1, Z_2, … with the following recursive expression:

$$Z_i = (aZ_{i-1} + c) (\mathrm{mod}\, m)$$

where m, a, c and Z_0 are non-negative integer parameters;

(2) Compute $U_i = Z_i / m$.

The values U_i seem to be real values uniformly distributed on the interval [0,1]

Example

Example: $Z_i = (5Z_{i-1} + 3)(\text{mod } 16)$

$Z_0 = 7$

i	Z_i	U_i	i	Z_i	U_i
0	7	----	10	9	0.563
1	6	0.375	11	0	0.000
2	1	0.063	12	3	0.188
3	8	0.500	13	2	0.125
4	11	0.688	14	13	0.813
5	10	0.625	15	4	0.250
6	5	0.313	16	7	0.438
7	12	0.750	17	6	0.375
8	15	0.938	18	1	0.063
9	14	0.875	19	8	0.500

- In this example, m = 16 and the algorithm repeats the generated numbers after 16 iterations (we say the generator has a period of 16).
- The random generator of MATLAB has a period of 2^{31}-1.

Generation of random numbers with other distributions

Discrete variables:

Consider a random variable that can have the values X_1, X_2, …, X_n.

Consider the probability of value X_i as $P(X = X_i) = f_i$.

Method:

- Split the interval [0,1] in n intervals proportional to f_i, $i = 1 \ldots n$
- Generate a uniformly distributed random value U in [0,1]
- Return X_i is U falls into the i-th interval

For example, the Bernoulli variable X with $p(0) = 1/4$ and $p(1) = 3/4$ can be randomly generated as:

(1) Generate $U \sim U(0,1)$

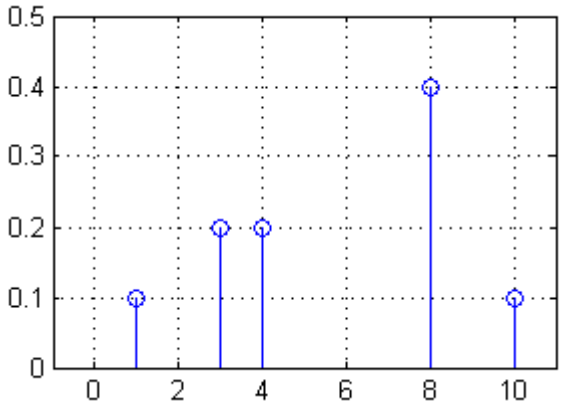
(2) If $U \leq 1/4$, return $X = 0$; otherwise, return $X = 1$

Generation of random numbers with other distributions

Discrete variables (MATLAB example)

```
x= [1 3 4 8 10];
f= [0.1 0.2 0.2 0.4 0.1];

figure(1)
stem(x,f)
axis([-1 11 0 0.5])
grid on
```



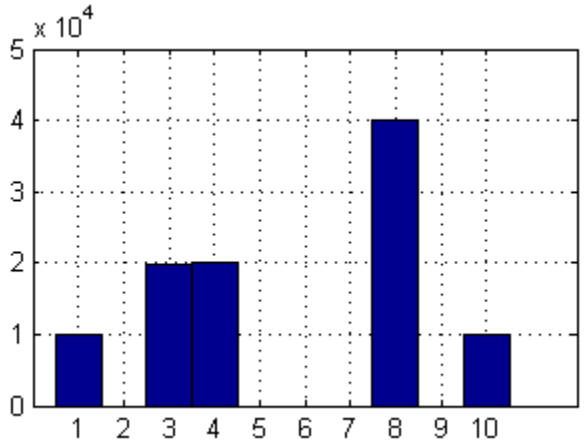

```
f_cum= [0 cumsum(f)]
```

```
f_cum =

    0.0   0.1   0.3   0.5   0.9   1.0
```

```
a= zeros(1,100000);
for it= 1:100000
    a(it)= x(sum(rand()>f_cum));
end

figure(2)
hist(a,1:10)
grid on
```



Generation of random numbers with other distributions

Continuous variables:

The most popular methods are based on the inverse of the cumulative distribution function (cdf).

Consider $F(X)$ as the cdf of a continuous random variable and $F^{-1}(U)$ as its inverse function.

Method:

(1) Generate $U \sim U(0,1)$

(2) Return $X = F^{-1}(U)$

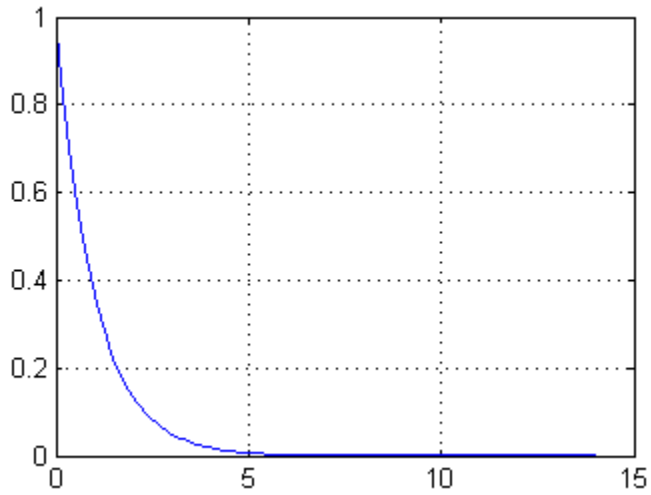
For example, an exponential distributed random variable with average $1/\lambda$:

$$F(x) = \begin{cases} 1 - e^{-\lambda x}, & x \geq 0 \\ 0, & x < 0 \end{cases} \qquad F^{-1}(U) = -\frac{1}{\lambda}\ln(U)$$

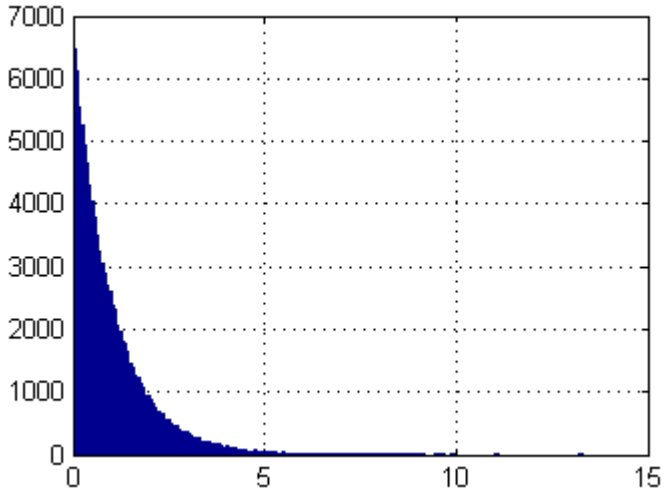
Generation of random numbers with other distributions

Exponential variable (MATLAB example)

```
x= 0:0.1:14;
f=exppdf(x,1)
figure(1)
plot(x,f)
grid on
```




```
a=exprnd(1,1,100000);
figure(2)
hist(a,200)
grid on
```



Analysis of the results of a simulation

Consider X_1, X_2, …, X_n as the observations of independent and identically distributed (IID) random variables with average μ and finite variance σ^2 (for example, the results of different simulations of a given system).

The sample mean defined by $\overline{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n}$

is an estimator for average μ.

The sample variance defined by $S^2(n) = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X}(n)\right)^2}{n-1}$

is an estimator for variance σ^2.

The analysis of the results of a simulation is, usually, based on the Central Limit Theorem.

Analysis of the results of a simulation

Consider Z_n as a random variable given by: $Z_n = \dfrac{\overline{X}(n) - \mu}{\sqrt{\sigma^2 / n}}$

Consider $F_n(z)$ the cumulative distribution function of Z_n for a sample of size n.

The *Central Limit Theorem* states that

$$\lim_{n \to +\infty} F_n(z) = \Phi(z)$$

where $\Phi(z)$ is the cumulative distribution function of a standard Gaussian random variable (i.e., a Gaussian distribution with mean 0 and variance 1).

Given that $\lim_{n \to +\infty} S^2(n) = \sigma^2$ than, the random variable $\dfrac{\overline{X}(n) - \mu}{\sqrt{S^2(n) / n}}$

has approximately a standard Gaussian distribution.

Analysis of the results of a simulation

For a sufficiently high value of n,

$$P\left(-z_{1-\alpha/2} \le \frac{\overline{X}(n)-\mu}{\sqrt{S^2(n)/n}} \le z_{1-\alpha/2} \right) =$$

$$P\left(\overline{X}(n)-z_{1-\alpha/2}\sqrt{S^2(n)/n} \le \mu \le \overline{X}(n)+z_{1-\alpha/2}\sqrt{S^2(n)/n} \right) \approx 1-\alpha$$

where $z_{1-\alpha/2}$ is the critical value of the standard Gaussian distribution ($z_{1-\alpha/2}$ is the value z such that $P(x \le z) = 1 - \alpha/2$ where x is a random variable with a standard Gaussian distribution).

Therefore, the approximate confidence interval of 100(1-α)% for the average μ is given as

$$\overline{X}(n) \pm z_{1-\alpha/2}\sqrt{S^2(n)/n}$$

Analysis of the results of a simulation

The approximate confidence interval of 100(1-α)% for the average μ is

$$\overline{X}(n) \pm z_{1-\alpha/2}\sqrt{S^2(n)/n}$$

In MATLAB:

```
N = 20; %number of simulations
results= zeros(1,N); %vector with N simulation results
for it= 1:N
       results(it)= simulator();
end

alfa= 0.1; %90% confidence interval%
media = mean(results);
term = norminv(1-alfa/2)*sqrt(var(results)/N);

fprintf('resultado = %.2e +- %.2e\n',media,term)
```

Analysis of the results of a simulation

The central limit theorem requires variables X_1, X_2, …, X_n to be independent and identically distributed (IID).

- One way to guarantee the independence between the different values is to run different simulations guaranteeing that the random values are different on the different runs.
- This is done by using different seeds on the random generators.

In general, the stochastic processes have initial transient states (which are dependent on the initial conditions) before reaching the stationary state.

- In order to guarantee that the performance estimations are correct, the simulation must first warm-up to let the transient states vanish.
- If the simulated time is much higher than the warm-up time, statistical counters can be initialized at the beginning of the simulation.
- Otherwise, the statistical counters must be initialized only after the warm-up time (this time must be estimated though).