Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Fourth Year
Spring 2013

CS433: Performance Evaluation
Assignment 2
Assigned: Wednesday, April 17th, 2013
Due: Wednesday, May 15th, 2013

Assignment 2

Name: Ali Hussein Ali

Number: 39

Name: Mohamed Hussein Ali

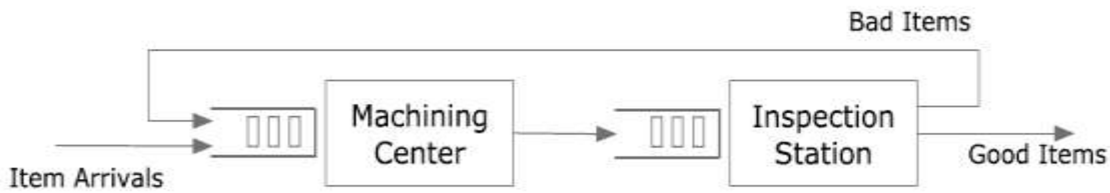
Number: 54

Factory Simulation

Objectives

Upon completion of this assignment, you will be able to apply basic simulation techniques to analyze and evaluate the performance of a certain model.

Model Description [1]



A small factory consists of a machining center and inspection station in series as shown in the above figure. Unfinished parts arrive at the factory with exponential interarrival times with a mean of 1 minute. Processing times at the machining center are uniform on the interval [0.65, 0.70] minute, and subsequent inspection times are uniformly distributed as [0.75, 0.80] minute.

Ten percent of the parts are bad and are sent back to the machine for rework (i.e. 90% of the inspected parts are good and are sent to shipping). You can assume infinite capacity for the queues of the above two modules.

The machining center is subject to randomly occurring breakdowns. In particular, a new (or a freshly-repaired) machine will break down after an exponential amount of time with a mean of 6 hours. Repair times are uniform on the interval [8, 12] minutes. If a part is being processed when the machine breaks down, then the machine continues where it left off upon the completion of repair. Assume the factory is initially empty and idle, and is working continuously without any breaks periods.

[1] Source: Averill M. Law. 1990. Design and analysis of simulation experiments for manufacturing applications (tutorial session). In *Proceedings of the 22nd conference on Winter Simulation* (WSC' 90),

Specifications
You are required to simulate the factory described earlier according to the following
specifications:
1- Implement an LCG random number generator, justifying the selection of the parameters
(Jain: section 26.2). Draw the histogram of 100,000 generated numbers. Test the
uniformity of your generator using chi-square test (Jain: section 27.1).
2- Implement an exponential random deviate using the inverse transformation technique
(Jain: section 28.1).
3- Simulate the above problem as discussed in class (Check Harry Perros, Chapter 1).
4- Your program should output the following:
a. Inter-arrival times at both queues.
b. Service times at both centers.
c. Total items' response times.
d. Queues lengths (see Jain: section 25.4).
e. Hourly throughput (Number of items sent for shipping per hour).
Plot queues lengths and hourly throughput with time.
5- Using a spreadsheet program, draw the histogram and calculate the average, the standard
deviation and 90% confidence interval of the above metrics.
Note: You are only required to calculate the average for the queues lengths.
6- Apply the initial data deletion and moving average of independent replications methods
for transient removal to the hourly throughput. (Jain: section 25.3)
Adjust the stopping criteria at step 3 (justify your selection).
7- Repeat the simulation 10 times (replications) with the stopping criteria adjusted at step 6.
8- Repeat the analysis of step 5 after removing the transient period and using the variance
estimation technique of independent replications (Jain: section 25.5).

Requirements

1- Implement an LCG random number generator, justifying the selection of the parameters

$$x_n = (2^{34} + 1)x_{n-1} + 1 \mod 2^{35}$$

$$x_n = (2^{18} + 1)x_{n-1} + 1 \mod 2^{35}$$

✐ Lower autocorrelations between successive numbers are preferable.

✐ Both generators have the same full period, but the first one has a correlation of 0.25 between *xn-1* and *xn*, whereas the second One has a negligible correlation of less than 2-18

```
int a =(int) Math.pow(2, 5)+1;

 int b = 1;

int m = Integer.MAX_VALUE;
```

justifying the selection of the parameters

- Integers m and b are relatively prime, that is, have no common factors other than 1.
- "Every prime number that is a factor of m is also a factor of a-1.
- "If integer m is a multiple of 4, a-1 should be a multiple of 4.
- Notice that all of these conditions are met if m=2^k, a = 4c + 1, and b is odd. Here, c, b, and k are positive integers.
- A generator that has the maximum possible period is called a
- Full-period generator.

Draw the histogram of 100,000 generated numbers. Test the Uniformity of your generator using chi-square test

Most commonly used test

✐ Can be used for any distribution

✐ Prepare a histogram of the observed data

✐ Compare observed frequencies with theoretical

k = Number of cells

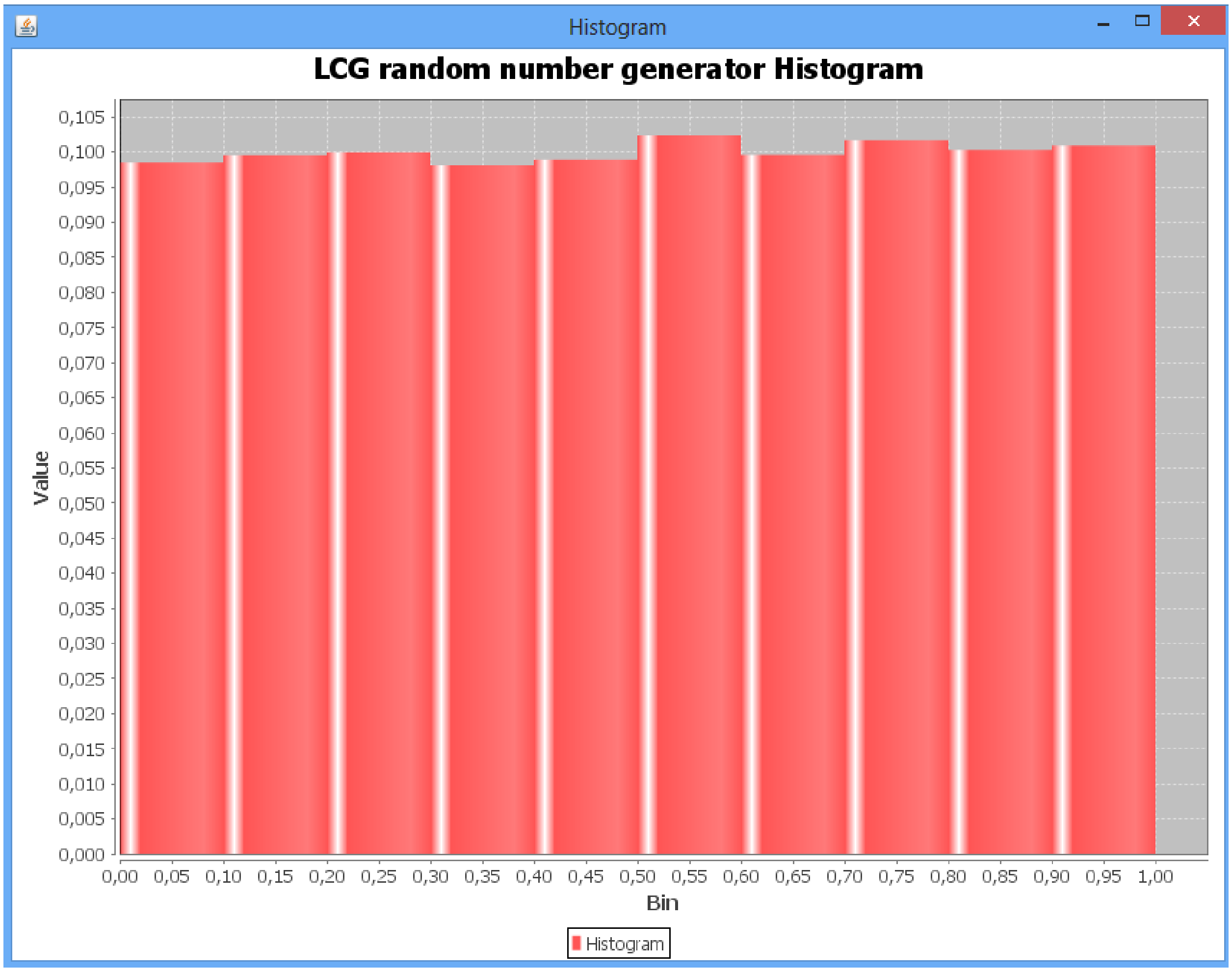
oi = Observed frequency for *i*th cell

ei = Expected frequency

$$D = \sum_{i=1}^{k} \frac{(o_i - e_i)^2}{e_i}$$

✐ *D*=0 ® Exact fit

✐ *D* has a chi-square distribution with *k*-1 degrees of freedom.



using chi-square test

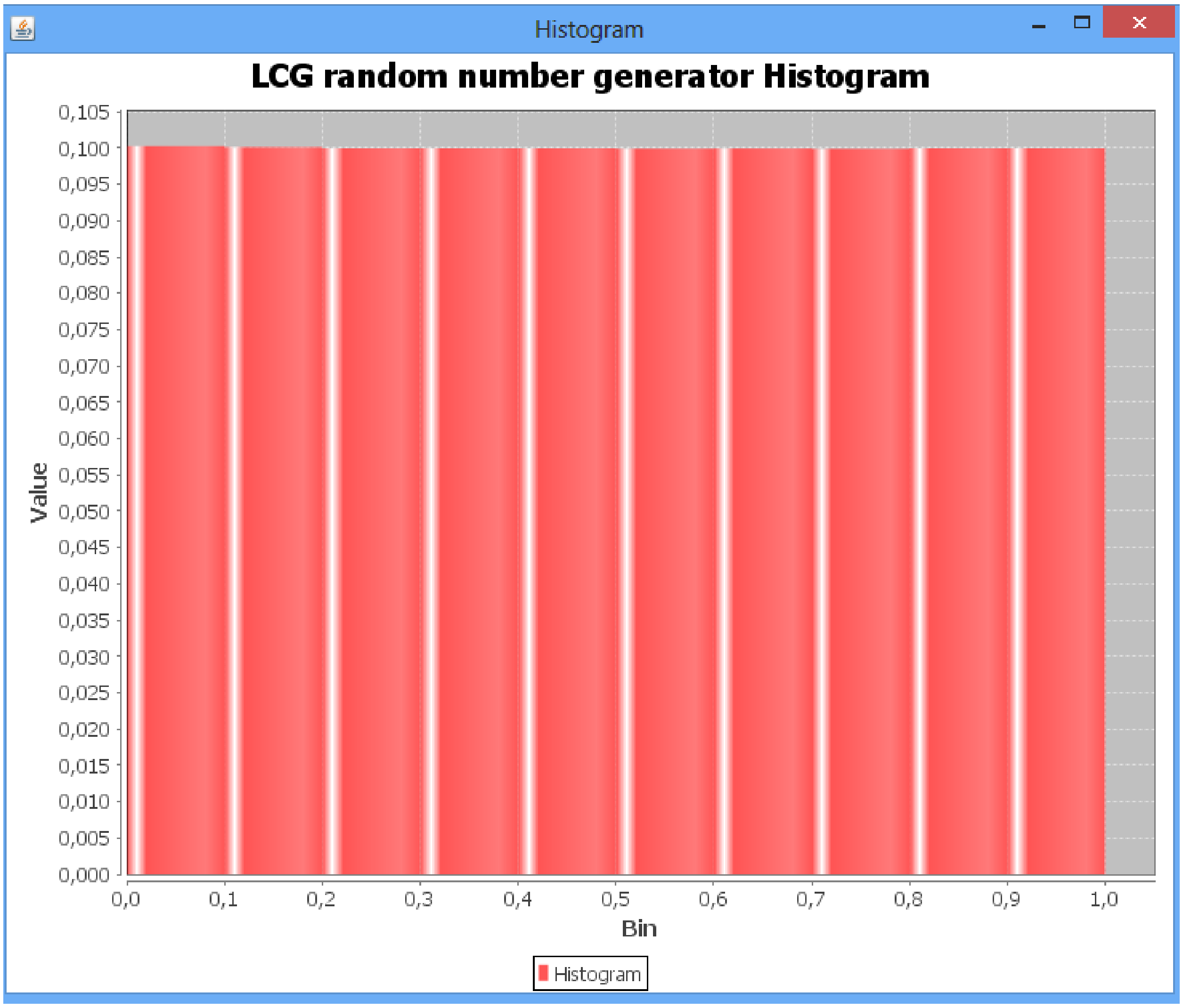
```
error =1.6842400000000035E-4
```

if we use these parameters

```
int a =(int) Math.pow(2, 18)+1;

int b = 1;

int m = Integer.MAX_VALUE;
```




```
The error decrease with these parameters
using chi-square test

Error =1.0280000000000158E-6
```

.

Multiplicative LCG: *b*=0

Two types:
m = 2*k*
m !=2*k*

when m!=2k

- Modulus m = prime number

With a proper multiplier a, period = m-1
Maximum possible period = m

- If and only if the multiplier a is a *primitive root* of the modulus m

- a is a primitive root of m if and only if an mod m $\bar{1}$ for n = 1,

when m =2k

m = 2k –> trivial division
Maximum possible period *2k-2*

- Period achieved if multiplier a is of the form 8i± 3, and the initial seed is an odd integer

- One-fourth the maximum possible may not be too small

- Low order bits of random numbers obtained using multiplicative LCG's with m=*2k* have a cyclic pattern

2- Implement an exponential random deviate using the inverse transformation technique

Using these parameters

int a =(int) Math.pow(2, 5)+1;

int b = 1;

int m = Integer.MAX_VALUE;

Used when F-1 can be determined either analytically or empirically.

- For exponential variates:

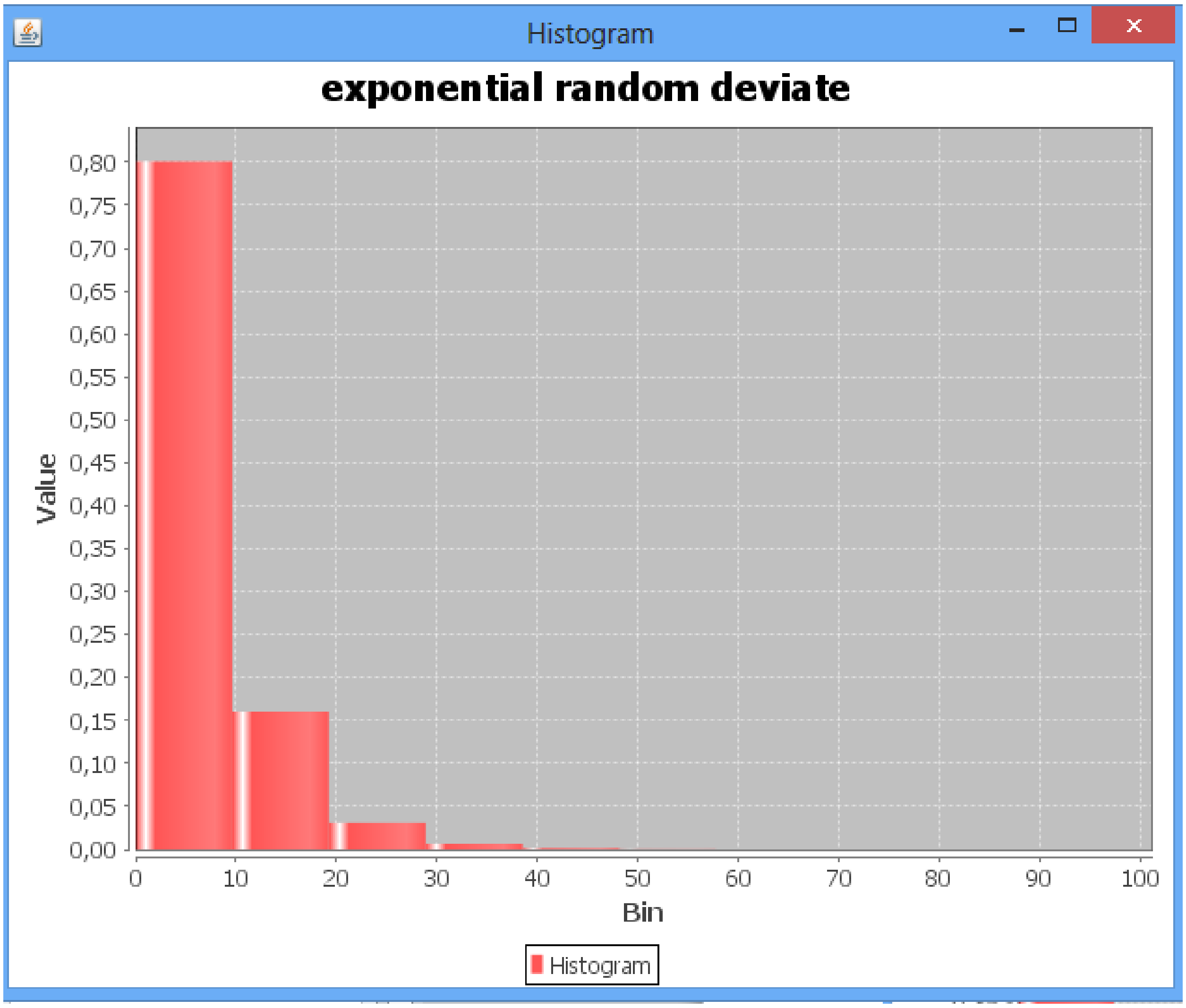
$$\text{The pdf } f(x) = \lambda e^{-\lambda x}$$
$$\text{The CDF } F(x) = 1 - e^{-\lambda x} = u \text{ or, } x = -\frac{1}{\lambda}\ln(1-u)$$

- If u is U(0,1), 1-u is also U(0,1)
- Thus, exponential variables can be generated by:

$$x = -\frac{1}{\lambda}\ln(u)$$

exponential random deviate function

```
double exp_rand_deviate(double lamda){
      double result = -(1/lamda)* Math.log(LCG());
      return result;
}
```



Error using using chi-square test = `5.682132185999998`

3-Simulate the above problem as discussed in class:

At Factory arrival Simulation:

Unfinished parts arrive at the factory with exponential interarrival times with a mean of 1 minute.

```
double exp_rand_deviate(double mean){
            double lamda = 1/mean;
            double result = -(1/lamda)* Math.log(LCG());
            return result;
      }
```

parts = 100
100 parts arrive at the factory with exponentional of 1 minute

```
  public void body() {
        for (int i=0; i < parts; i++) {
          // Send the processor a job
          sim_schedule(out, 0.0, 0);
          // Pause
          double delay = exp_rand_deviate(1)
          sim_pause(delay);
        }
   }
```

At the machining center Simulation: Processing times at the machining center are uniform on the interval [0.65,0.70] minute , The machining center is subject to randomly occurring breakdowns. In particular, a new (or a freshly-repaired) machine will break down after an exponential amount of time with a mean of 6 hours so we generate a random number with mean = 6 hours

Repair times are uniform on the interval [8, 12] minutes. If a part is being processed when the machine breaks down, then the machine continues where it left off upon the completion of repair.
min =0.67;
max =0.7;

```
public void body() {
// fail every 6 hours
int failure = (int) failureGenerator.exponential(360);
double count = 0;
while (Sim_system.running()) {

Sim_event e = new Sim_event();
sim_get_next(e);
```

```
double rand = LCG()*(max-min)+min
sim_process(rand);
sim_completed(e);
sim_schedule(out, 0.0, 1);
if (count++ <= failure) {
sim_pause(repairGenerator.uniformOver(8, 12));
failure = (int) failureGenerator.exponential(360);

}

}
}
```

At The inspection station Simulation : Processing times at The inspection station are uniformly distributed as [0.75, 0.80] minute.
Ten percent of the parts are bad and are sent back to the machine for rework (i.e. 90% of the Inspected parts are good and are sent to shipping).

```
public void IspectionProcess() {
while (Sim_system.running()) {
Sim_event e = new Sim_event();
sim_get_next(e);
double DelayTime = LCG()*(0.8-0.75)+0.75;
//processing Time
sim_process(DelayTime)
sim_completed(e);
double rand = LCG();

if rand < 0.10) {
// Ten percent of the parts are bad and are sent back to the machine for rework
Resumulate(0.1)
}
}
}
```

4- Your program should output the following:

a. Inter-arrival times at both queues.

nq = Number of jobs waiting

- *ns* = Number of jobs receiving service
- *r* = Response time or the time in the system
  = time waiting + time receiving service
- *w* = Waiting time
  = Time between arrival and beginning of service

For first 20 elements

Machining center	Inspection station
inter-arrival time	inter-arrival time
1.4000122071243881	1.4000244141789266
1.4000366212334652	1.400054931803632
1.4000732423737987	1.4000976564595935
1.4001220705453878	1.4001525881468098
1.4001831057482317	1.4002197268652816
0.9118687248701605	0.9119114495028393
0.4883876231121711	0.6502197267721499
0.9119541741355182	0.7502258302877785
0.7502197267721495	0.7502807619517107
0.7502746584360835	0.7503417971312736
0.7503356936156447	0.7504089358264618
0.7504028323108347	0.750482178037279
0.7504760745216519	0.7505615237637251
0.7505554202480962	0.7506469730057965
0.7506408694901694	0.7507385257634986
0.41175340033594665	0.6509338383101557
0.3389790219119231	0.6510437016147392
0.750830078521199	0.6511596684349499
0.7509338383101536	0.6512817387707912
0.7510437016147407	0.6514099126222597
0.7511596684349513	0.6515441899893553
0.7512817387707926	0.651684570872078
0.7514099126222575	0.699334716656864
0.7515441899893531	0.7516967779033337
0.7516845708720794	0.7518432623016871
0.17713806863136128	0.6523071295592509
0.574692986639068	0.6524780280201128

0.7519836431844098	0.6526550299966019
0.7521423346140139	0.6528381354887216
0.7523071295592487	0.6530273444964685
0.7524780280201142	0.6532226570198425
0.7526550299966033	0.6534240730588436
0.752838135488723	0.6977722164931563

b. Service times at both centers.

Machining center	Inspection station
Service-Time	Service-Time
0.650006103562194	0.7500061035621941
0.6500183106167325	0.7500183106167326
0.6500366211868993	0.7500366211868994
0.6500610352726941	0.7500610352726937
0.650091552874116	0.7500915528741157
0.650128173991166	0.7501281739911656
0.6501708986238448	0.7501708986238445
0.6502197267721499	0.7502197267721495
0.6502746584360839	0.7502746584360835
0.650335693615645	0.7503356936156447
0.6504028323108351	0.7504028323108347
0.6504760745216522	0.7504760745216519
0.6505554202480965	0.7505554202480962
0.6506408694901697	0.7506408694901694
0.6507324222478701	0.7507324222478697
0.6508300785211993	0.750830078521199
0.6509338383101557	0.7509338383101536
0.6510437016147392	0.7510437016147407
0.6511596684349499	0.7511596684349513
0.6512817387707912	0.7512817387707926
0.6514099126222597	0.7514099126222575
0.6515441899893553	0.7515441899893531
0.651684570872078	0.7516845708720794
0.6518310552704278	0.7518310552704293
0.6519836431844084	0.7519836431844098
0.652142334614016	0.7521423346140139
0.6523071295592509	0.7523071295592487
0.6524780280201128	0.7524780280201142
0.6526550299966019	0.7526550299966033
0.6528381354887216	0.752838135488723
0.6530273444964685	0.7530273444964664
0.6532226570198425	0.7532226570198404

a. Total items' response times.

Response-Time

12.628892217894025
7.263607240402306
27.73066408409768
36.3987339828015
22.177480019447106
4.4952159628459025
1.413195802976003
1.4137695334915863
1.414355471038425 2
1.414953615616519 7
1.415563967225870 2
1.416186525866476 3
1.41682129153833 8
1.417468264241455 3
1.418127443975828 2
1.418798830741457 2
1.419482424538341 7
1.4201782253664819
1.4208862332258776
1.421606448116529
1.4223388700384363
1.4230834989915992
1.4238403349760178
1.424609377991692

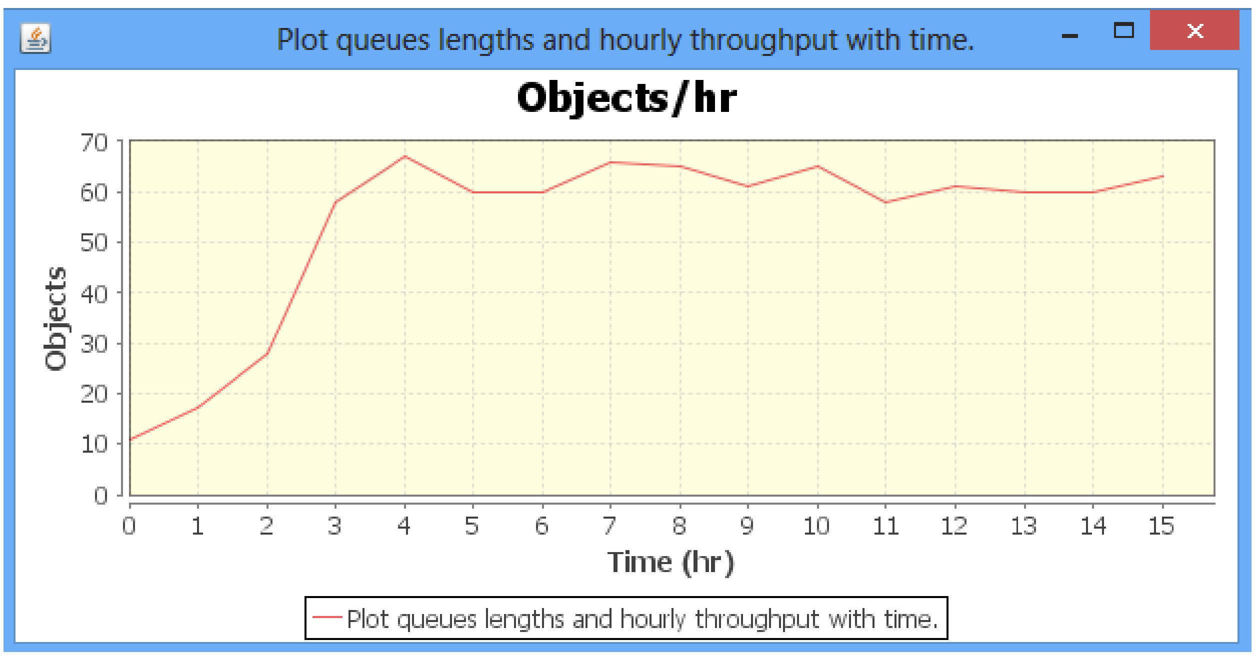
d. Queues lengths.

Machining center	Inspection station
Queues Length	Service-Time
0	0
1	1
0	0
1	1
0	0
1	1
0	0

1	1
0	0
1	1
0	0
1	1
0	0
1	1
0	0
1	1
0	0
1	1
0	0
1	1

e. Hourly throughput (Number of items sent for shipping per hour).

[11.0 17.0 28.0 58.0 67.0 60.0 60.0 66.0 65.0 61.0 65.0 58.0 61.0 60.0 60.0 63.0]

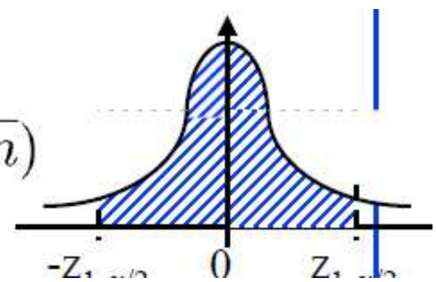


5- Using a spreadsheet program, draw the histogram and calculate the average, the standard deviation and 90% confidence interval of the above metrics. Note: You are only required to calculate the average for the queues lengths.

100(1-a)% confidence interval for μ:

$$\left(\bar{x} - z_{1-\alpha/2}s/\sqrt{n}, \bar{x} + z_{1-\alpha/2}s/\sqrt{n}\right)$$

$z_{1-\alpha/2}$ = (1-α/2)-quantile of N(0,1)



x = Mean

σ = Standard Deviation

α = 1 - (Confidence Level/100)

$Z_{\alpha/2}$ = Z-table value

$t_{\alpha/2}$ = t-table value

CI = Confidence Interval

a. Total items' response times.

Results:

Total Numbers:	893
Mean (Average):	3.74609
Standard deviation:	3.52027
Variance(Standard deviation):	12.39232
Population Standard deviation:	3.5183
Variance(Population Standard deviation):	12.37845

CI for Mean = 3.552 < μ < 3.940

e. Hourly throughput

Results:

Total Numbers:	16
Mean (Average):	53.75
Standard deviation:	17.89413
Variance(Standard deviation):	320.2
Population Standard deviation:	17.32592
Variance(Population Standard deviation):	300.1875

CI for Mean = 45.921 $< \mu <$ 61.579

<u>for the machining center</u>

b. Inter-arrival times

Results:

Total Numbers:	1020
Mean (Average):	0.97862
Standard deviation:	0.959
Variance(Standard deviation):	0.91969
Population Standard deviation:	0.95853
Variance(Population Standard deviation):	0.91879

CI for Mean = 0.929 $< \mu <$ 1.028

c. Service times

Results:

Total Numbers:	1020
Mean (Average):	0.67382
Standard deviation:	0.01474
Variance(Standard deviation):	0.00022

CI for Mean = 0.673 $< \mu <$ 0.675

d. Queues lengths

Results:	
Total Numbers:	2020
Mean (Average):	1.93564
Standard deviation:	3.11777
Variance(Standard deviation):	9.72047
Population Standard deviation:	3.117
Variance(Population Standard deviation):	9.71566

<u>For Inspection Station:</u>

Inter arrival time:

Results:	
Total Numbers:	1020
Mean (Average):	0.97864
Standard deviation:	0.62321
Variance(Standard deviation):	0.38839
Population Standard deviation:	0.6229
Variance(Population Standard deviation):	0.388

CI for Mean = 0.946 < μ < 1.011

Service time:

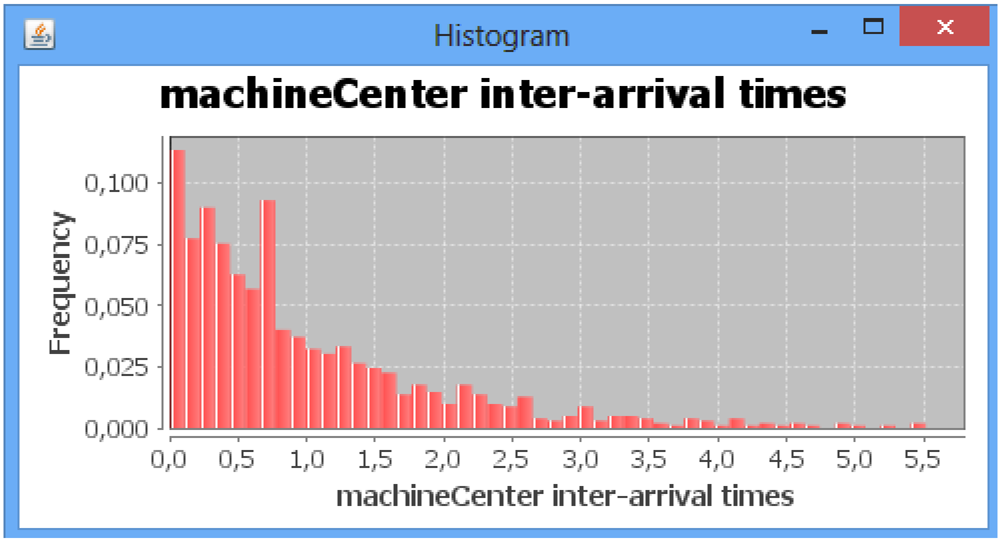
Mean (Average):	0.77381
Standard deviation:	0.01475
Variance(Standard deviation):	0.00022
Population Standard deviation:	0.01474
Variance(Population Standard deviation):	0.00022

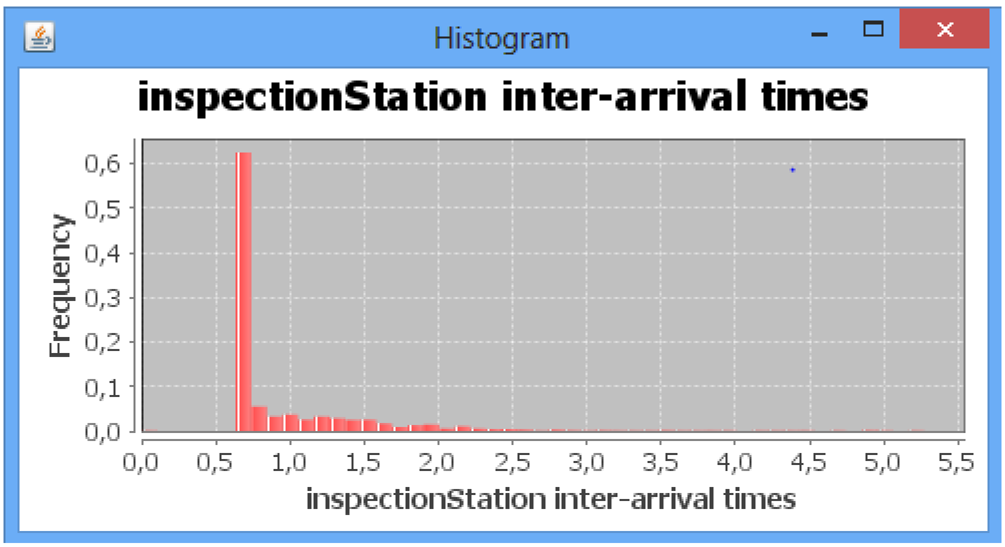
CI for Mean = 0.773 < μ < 0.775

Queue Length:

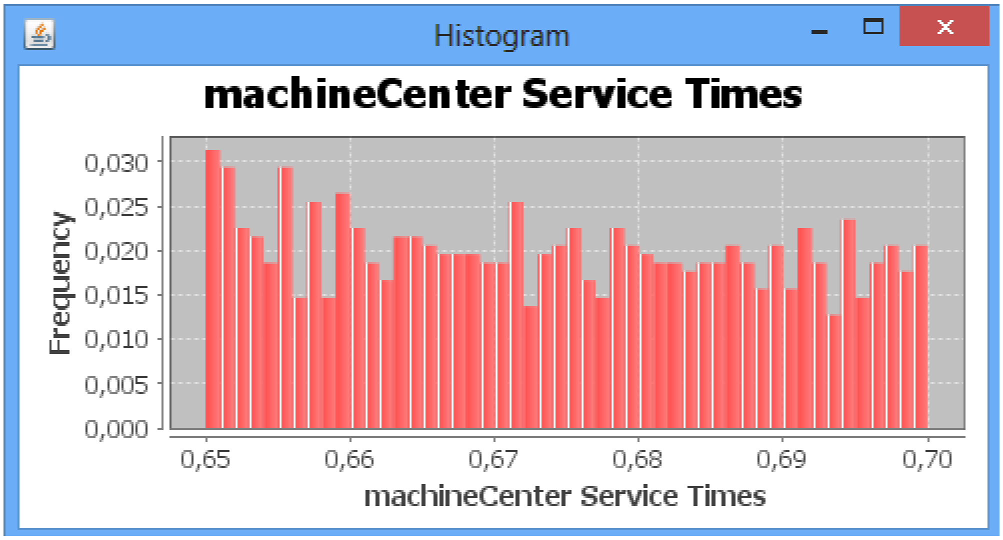
Results:	
Total Numbers:	2020
Mean (Average):	1.05941
Standard deviation:	1.13542
Variance(Standard deviation):	1.28919
Population Standard deviation:	1.13514
Variance(Population Standard deviation):	1.28855

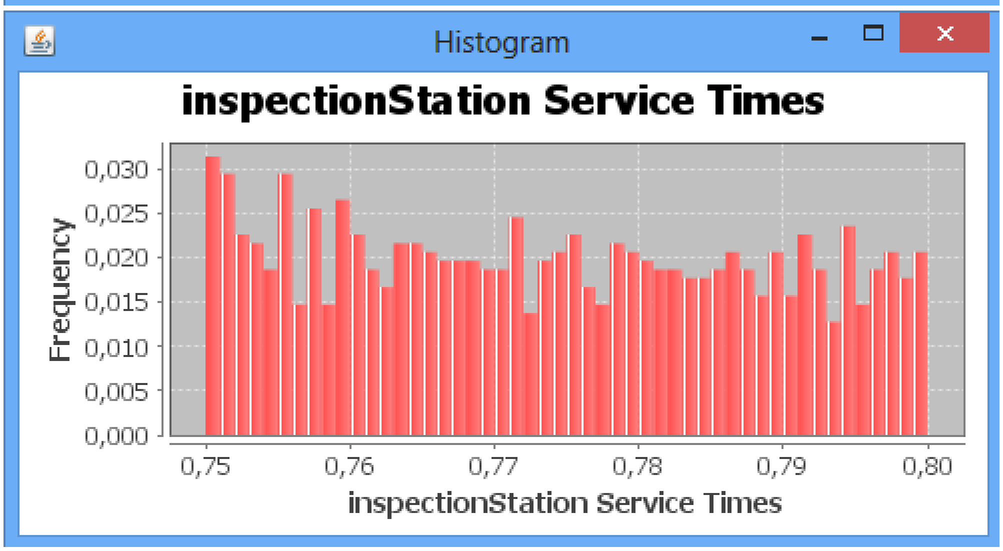
CI for Mean = 1.018 < μ < 1.101



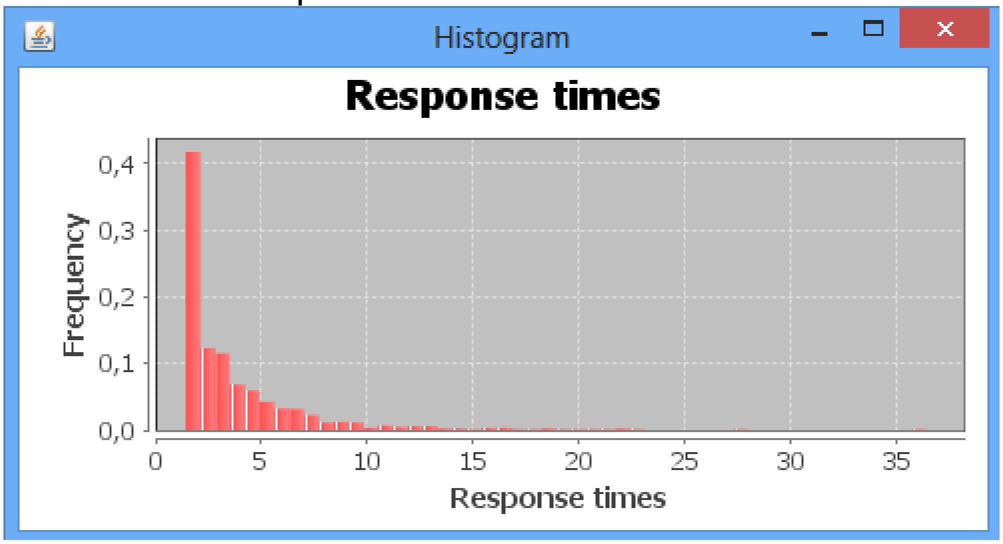


b. Service times at both centers.

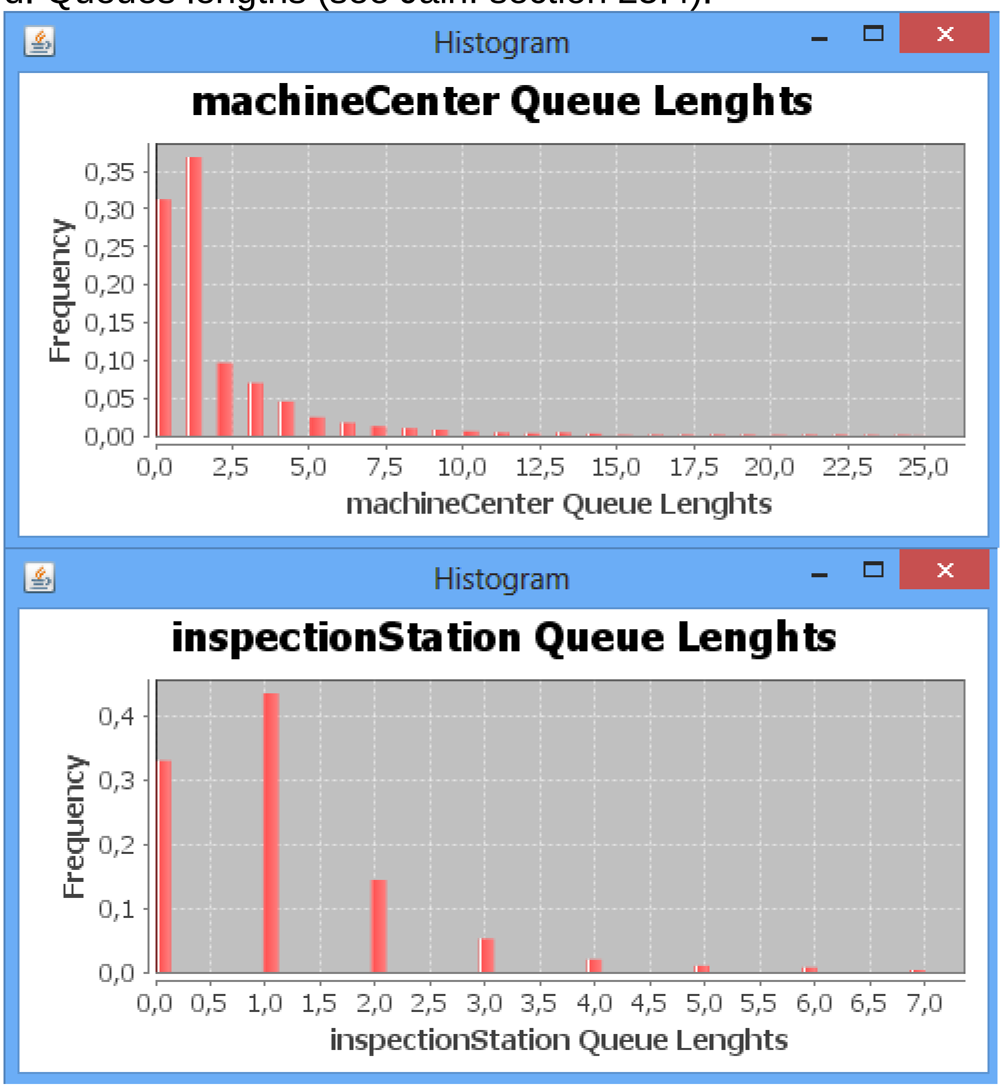

Histogram
machineCenter Service Times
Frequency
0,030
0,025
0,020
0,015
0,010
0,005
0,000
0,65
0,66
0,67
0,68
0,69
0,70
machineCenter Service Times


Histogram
inspectionStation Service Times
Frequency
0,030
0,025
0,020
0,015
0,010
0,005
0,000
0,75
0,76
0,77
0,78
0,79
0,80
inspectionStation Service Times

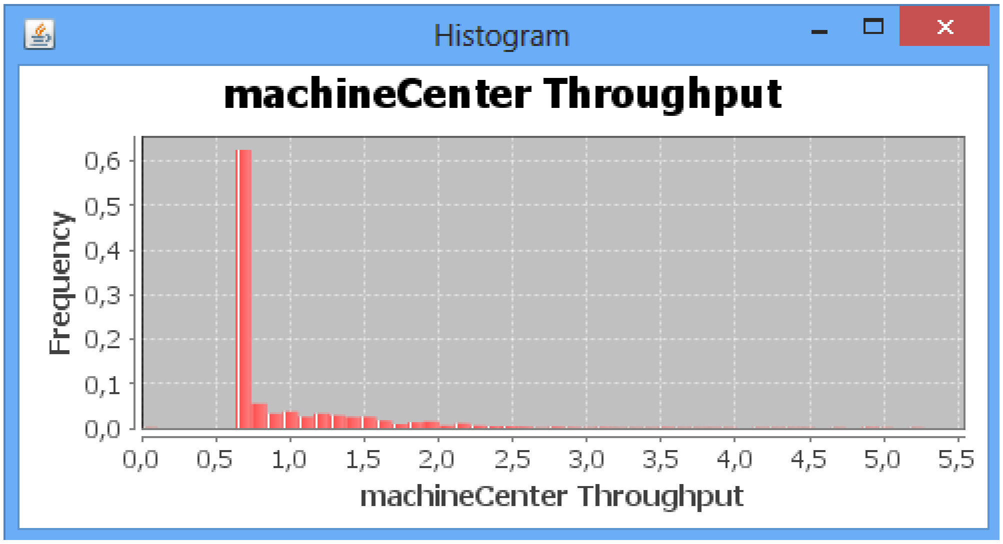
c. Total items' response times.

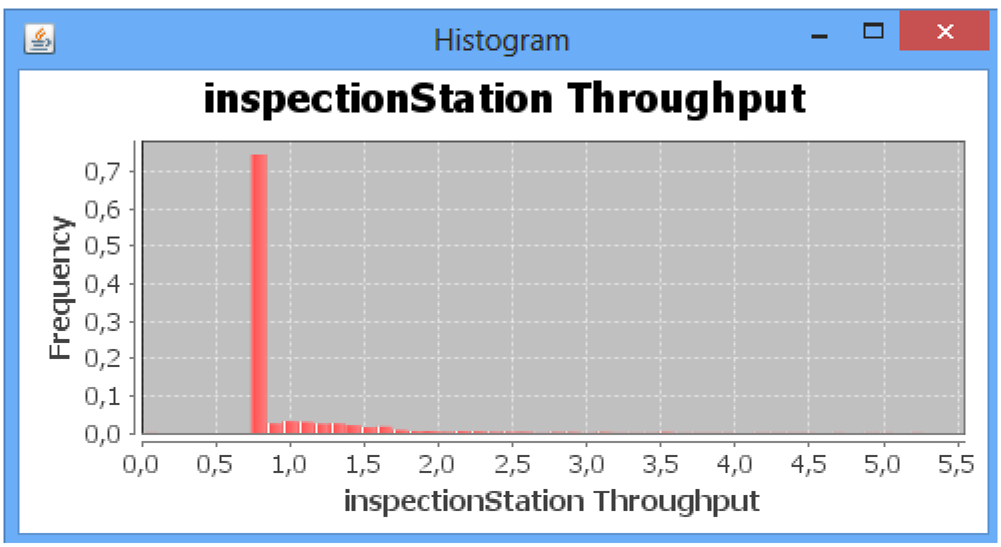


d. Queues lengths (see Jain: section 25.4).



e. Hourly throughput (Number of items sent for shipping per hour).





Repeat the simulation 10 times (replications) with the stopping criteria adjusted at step 6.

Run the simulation 10 times with different seeds then applying data deletion and moving average of independent replications methods for transient removal to hourly throughput

The hourly Throughput of repeating the simulation 10 times :

13.0 20.0 36.0 66.0 61.0 62.0 61.0 64.0 60.0 66.0 59.0 59.0 59.0 58.0 60.0 64.0

12.0 18.0 31.0 64.0 62.0 63.0 60.0 63.0 65.0 62.0 63.0 58.0 60.0 59.0 58.0 65.0

16.0 28.0 60.0 63.0 55.0 64.0 63.0 58.0 60.0 60.0 60.0 60.0 62.0 61.0 65.0 60.0

16.0 29.0 62.0 65.0 64.0 63.0 61.0 62.0 59.0 55.0 61.0 54.0 61.0 63.0 67.0 55.0

28.0 60.0 68.0 63.0 59.0 65.0 61.0 61.0 62.0 66.0 60.0 57.0 63.0 57.0 58.0 60.0

23.0 64.0 57.0 59.0 61.0 56.0 61.0 65.0 67.0 61.0 55.0 57.0 54.0 53.0 57.0 61.0

17.0 33.0 66.0 57.0 66.0 55.0 63.0 64.0 58.0 60.0 60.0 62.0 61.0 60.0 62.0 50.0

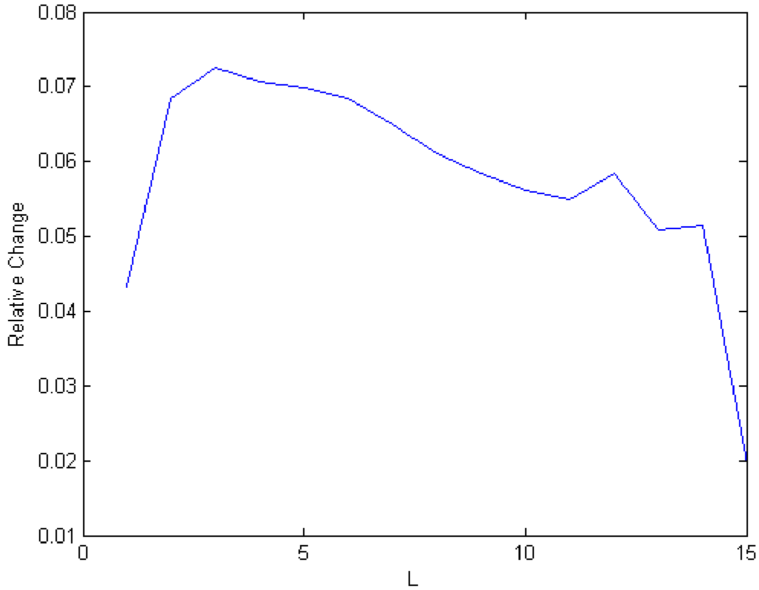
38.0 72.0 59.0 61.0 58.0 62.0 67.0 62.0 62.0 56.0 60.0 60.0 64.0 64.0 59.0 54.0

17.0 34.0 68.0 60.0 60.0 59.0 65.0 59.0 58.0 64.0 58.0 66.0 68.0 55.0 60.0 48.0

17.0 31.0 65.0 58.0 63.0 61.0 57.0 59.0 57.0 54.0 62.0 53.0 57.0 61.0 64.0 57.0

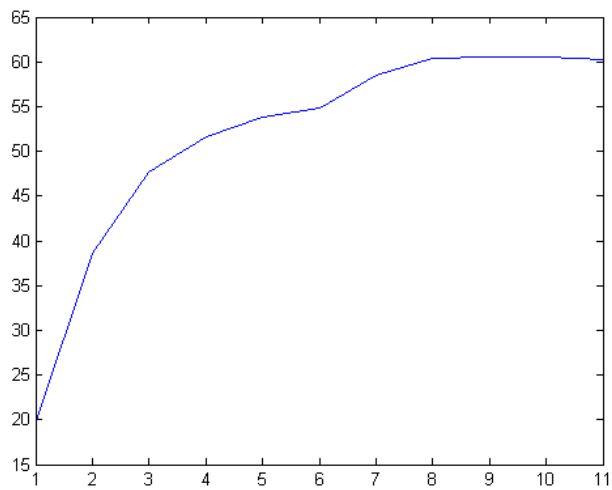
By applying initial data deletion

Result :



transient period = 11

By applying moving average
result :

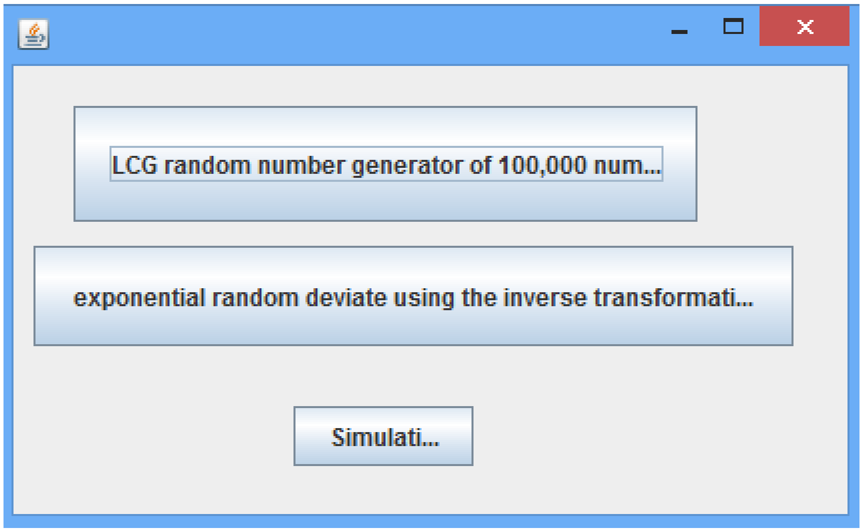


transient period = 8

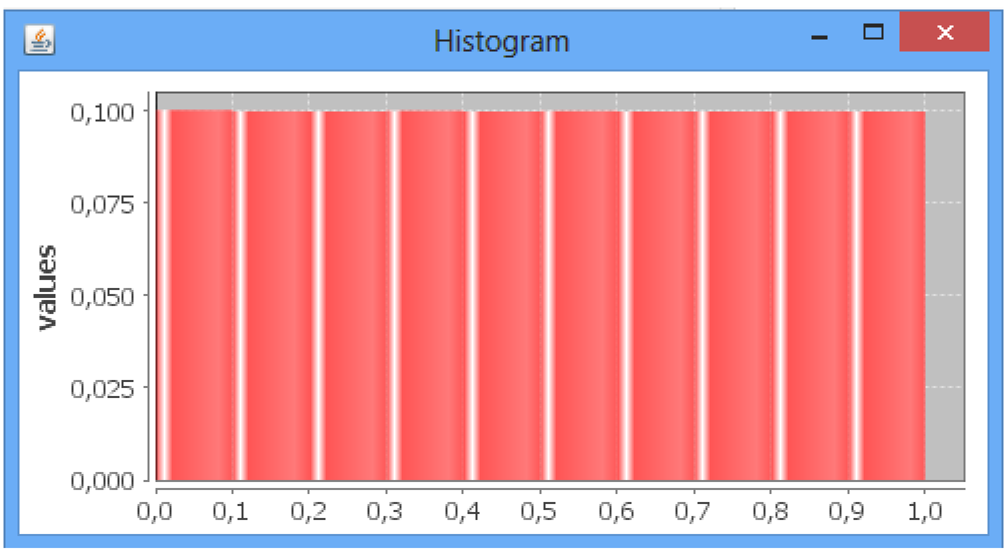
ScreenShoots :

Run the system

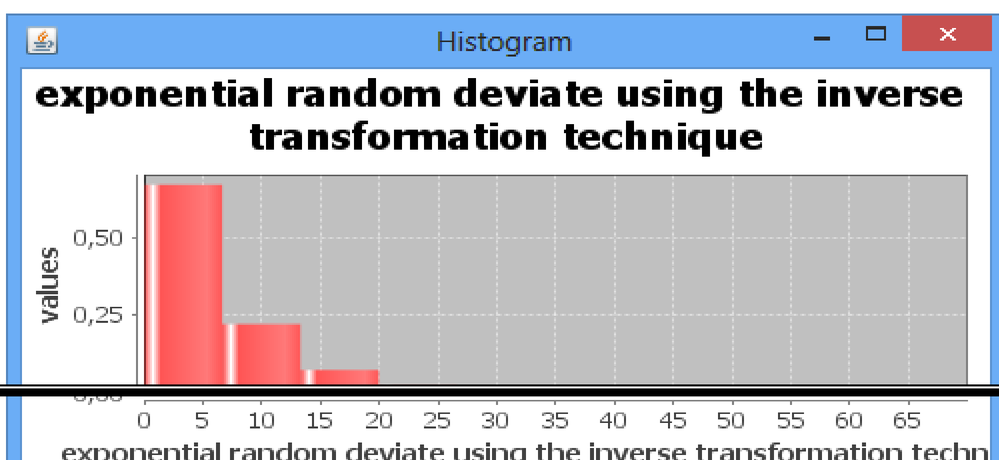
Parameters :a = 2^18 + 1   b = 1   m = Integer.MAX_VALUE;



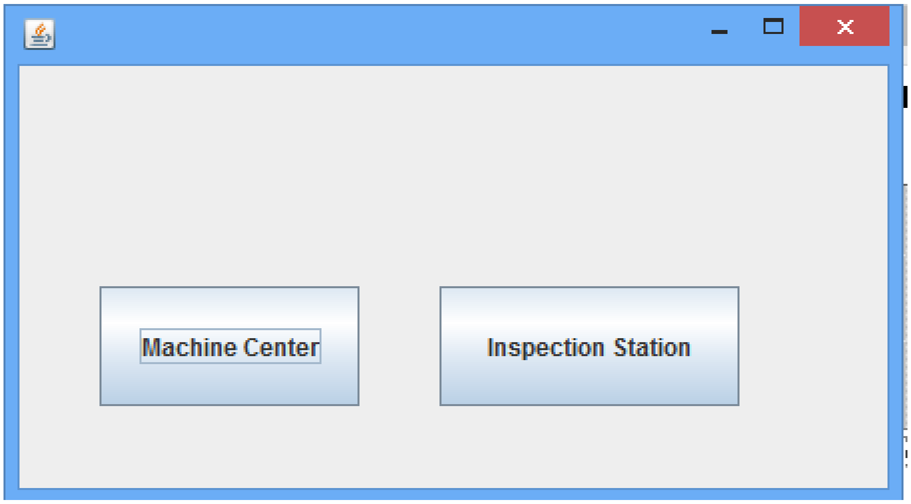
Click LCG random number generator of 100,000 numbers



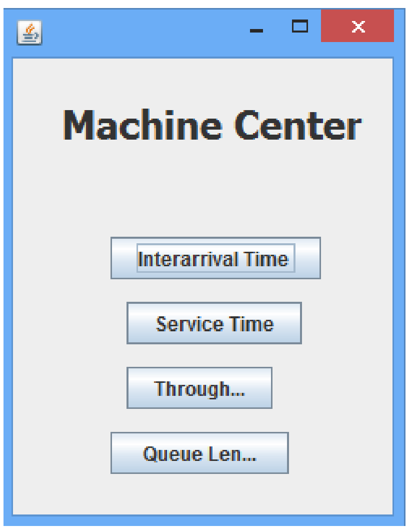
Click exponential random deviate usin inverse transformation



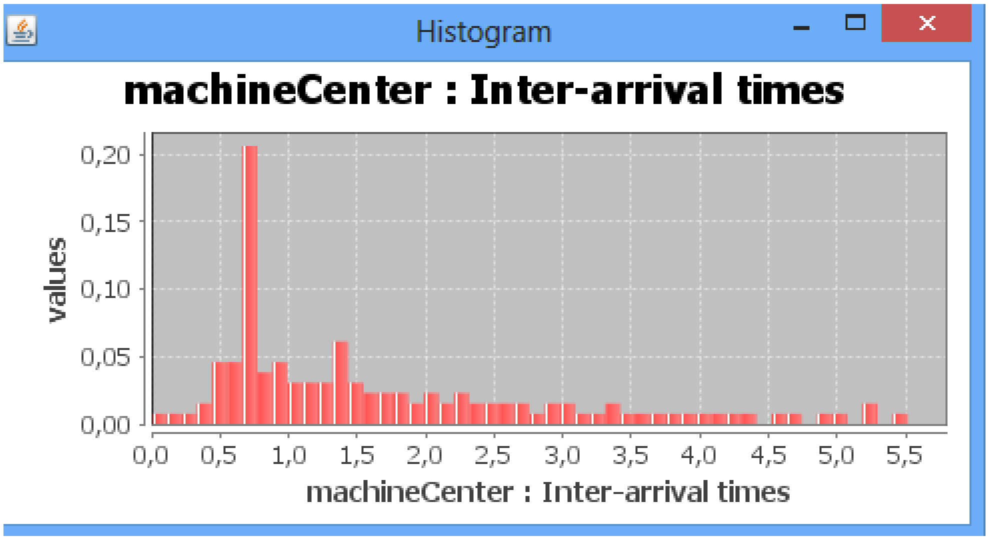
Click simulation

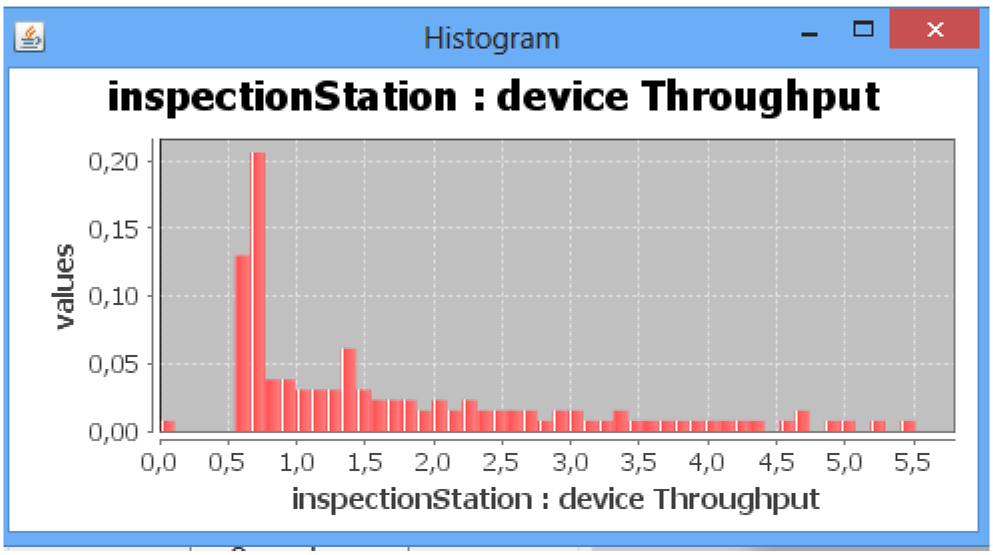


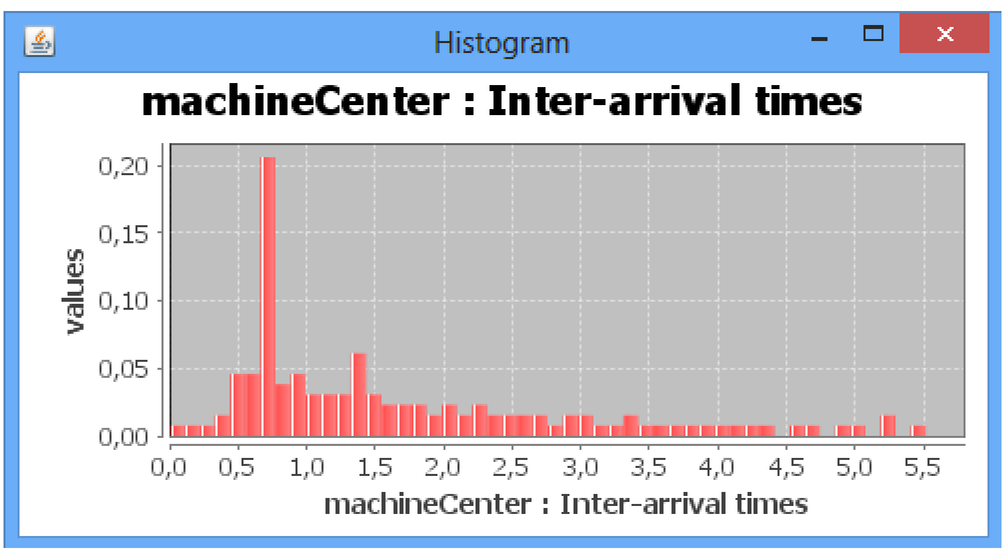
Click machine Center



Click Inspection Station


Histogram
machineCenter : Inter-arrival times
values
0,20
0,15
0,10
0,05
0,00
0,0 0,5 1,0 1,5 2,0 2,5 3,0 3,5 4,0 4,5 5,0 5,5
machineCenter : Inter-arrival times


Histogram
inspectionStation : device Throughput
values
0,20
0,15
0,10
0,05
0,00
0,0 0,5 1,0 1,5 2,0 2,5 3,0 3,5 4,0 4,5 5,0 5,5
inspectionStation : device Throughput


Histogram
machineCenter : Inter-arrival times
values
0,20
0,15
0,10
0,05
0,00
0,0 0,5 1,0 1,5 2,0 2,5 3,0 3,5 4,0 4,5 5,0 5,5
machineCenter : Inter-arrival times