HW2 - Simulation Report

Rajendran, Kirubakumaresh
A20328479
Bangalore
Individual Submission

1. Random Number Generator

1.1 Does your RNG generate random numbers?

The objective is to test if the sequence produced by the random number generator is random (pseudo-random). The approach is to generate 100000 random numbers and split them into 1000 buckets. The expectation is that the distribution should be uniform and each bucket should have approximately 1000 values. We can plot the CDF of the observed values and expected values to verify if the observed one follows uniform distribution. It is evident from the below plot that the RNG generates random numbers from uniform distribution.

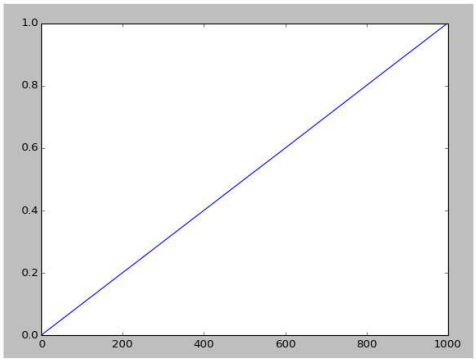


Fig a. CDF of Observed values

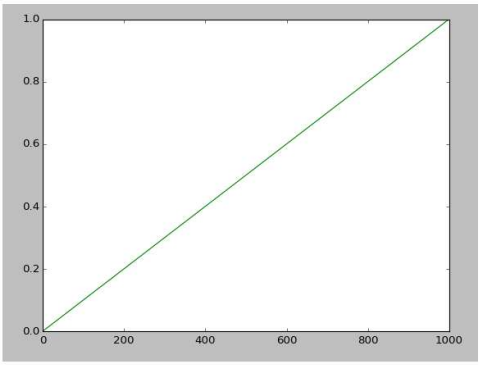


Fig b. CDF of expected values

1.2 How do you initialize the seed of your RNG?

The best way to initialize the seed is to use the time or any random number.

1.3 Generate two sequences of 1000000 number each, for every sequence use a different seed. Are these two sequences different? How do you know this?

This can be tested by generating two sequences of random numbers using different seed and then plotting them to understand their correlation. In Fig 1.3.1.a, we tried to plot random sequences generated from same seed. They will be very strong correlated

and it is visible in the plot whereas the ones generated from different seeds are random as evident from fig 1.3.1.b

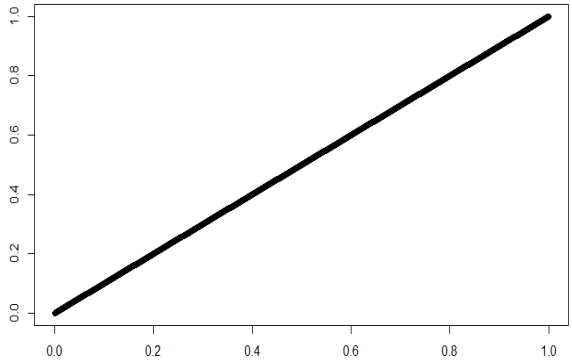


Fig 1.3.1.a Scatter plot of two random sequences generated with same seed

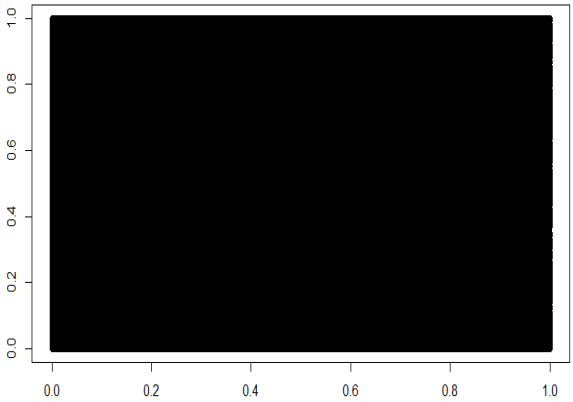


Fig 1.3.1.b Scatter plot of two random sequences generated with different seed

2. Warm-up period Elimination

We have a M/M/2/2+5 queuing system. We are going to study the following systems

a. System A : arrival rate = 2, service rate = 1
b. System B : arrival rate = 10, service rate = 1

In this section, we will focus on eliminating the warm up period for the above systems in four different scenarios

a. System A with the initial condition x(t=0)=0, i.e. the system is empty at t=0.
b. System A with the initial condition x(t=0)=7, i.e. the system is full at t=0.
c. System B with the initial condition x(t=0)=0, i.e. the system is empty at t=0.
d. System B with the initial condition x(t=0)=4, i.e. there are 4 customers in the system at t=0.

Pseudo code for warm-up period elimination

Repeat the below steps for 1000 simulations.

a. Initialize Seed with Different value.
b. Initialize the system with the given state
    - Set current buffer size and max buffer size
    - Set current server status and max servers in the system
    - Load the next events in the linked list.
    - Set the system counters to empty
    - Set clock to 0.
b. Repeat the below steps until Clock > threshold.
    - Read next event from linked list
    - Advance system clock
    - If arrival
        i. If server available
            1. Assign customers to server
            2. Add Departure Event to the linked list
        ii. If server not available
            1. If Queue available
                a. Add customers to the queue
            2. If Queue not available
                a. Update blocking counter
                b. Remove customer from linked list
        iii. Predict next arrival and add to linked list
    - If Departure
        i. Release the server
        ii. If queue not empty
            1. Assign next customer to server
            2. Add departure event to linked list
        iii. Release the customer from linked list
        iv. Capture Response time for the customer
    - Capture number of customers in the system at this time and add to the state information.

Once the simulation has been run 1000 times, we computed the number of customers at every deltaT (1t) across all the runs

First Run: Y1_0, Y1_1, Y1_2, Y1_3…Y1_M

Second Run: Y2_0,Y2_1,Y2_2,Y2_3….Y2_M

……..

1000^{th} Run: Y1000_0, Y1000_1,Y1000_2,..Y1000_M

We average across time to get the final vector and plotted it to visually identify the stationary period.

$\overline{Y0}, \overline{Y1}, \ldots\ldots. \overline{YM}$

2.1 System A with the initial condition x(t=0)=0, i.e. the system is empty at t=0.

Arrival Rate	2
Service Rate	1
Queue Length	5
Number Of Servers	2
Clock	0
Current Queue Length	0
Current Busy Servers	0
Initial Events	Arrival of First Customer
Stopping Condition	Clock > 100

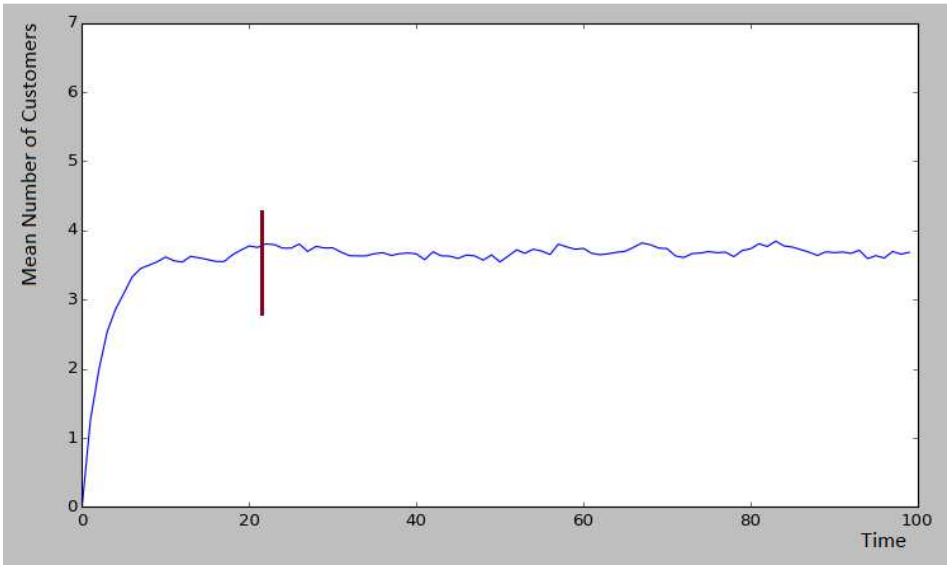
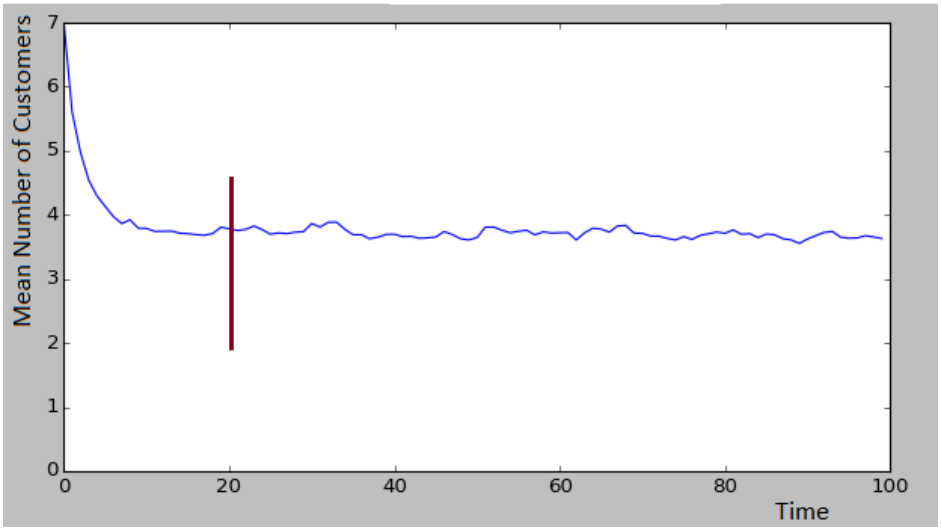


Figure 2.1: The system reaches stationary state around t20

It is clear from the plot that the system reaches stationary state around t20. The mean number of customers in the system does not oscillate much in the stationary. This particular system has same arrival rate and service rate. Hence the movement of customers in the systems is very frequent and is evident from the fact that the mean oscillates between 3.5 and 4.

2.2 System A with the initial condition x(t=0)=7, i.e. the system is full at t=0.

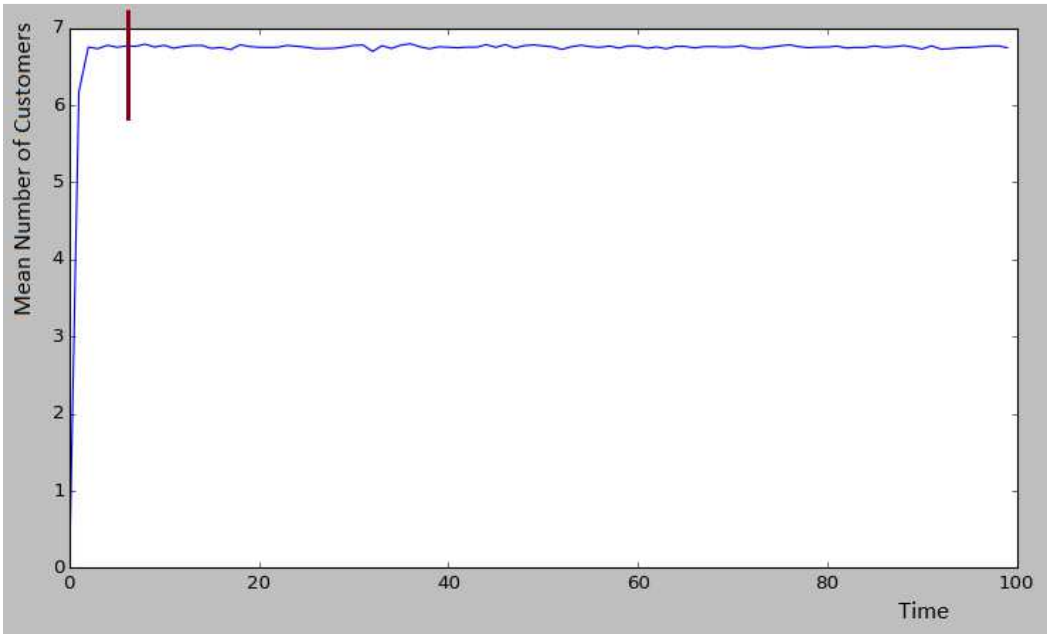
Arrival Rate	2
Service Rate	1
Queue Length	5
Number Of Servers	2
Clock	0
Current Queue Length	5
Current Busy Servers	2
Initial Events	Departure of 1st Customer Departure of 2nd Customer Arrival of 8th Customer
Stopping Condition	Clock > 100



In this case the system started with full state and hence it is different from the one that we got before. In this case, although it seems to get into stationary state around 15. It is safe to assume that the system reaches stationary state around t20.

2.3 System B with the initial condition x(t=0)=0, i.e. the system is empty at t=0.

Arrival Rate	10
Service Rate	1
Queue Length	5
Number Of Servers	2
Clock	0
Current Queue Length	0
Current Busy Servers	0
Initial Events	Arrival of First Customer
Stopping Condition	Clock > 100

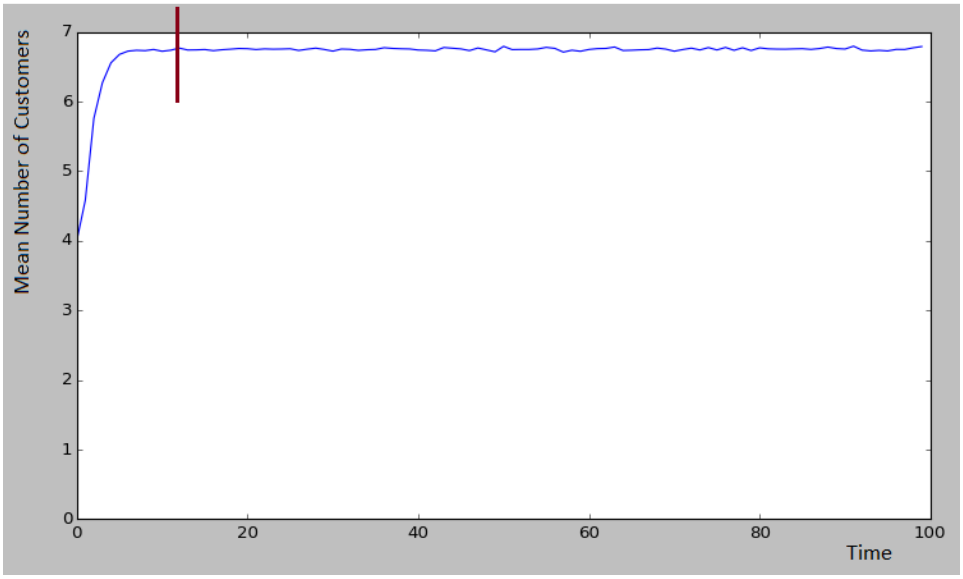


The system started with empty state. The incoming rate is much faster than the service rate. Hence the system quickly reaches the stable state. The stationary state appears to be around t7.

2.4 System B with the initial condition x(t=0)=4, i.e. there are 4 customers in the system at t=0.

Arrival Rate	10
Service Rate	1
Queue Length	5
Number Of Servers	2
Clock	0
Current Queue Length	2

Current Busy Servers	2
Initial Events	Departure of 1st Customer Departure of 2nd Customer Arrival of 5th Customer
Stopping Condition	Clock > 100



The system started with a non-empty state with 4 customers already in the system. In this case, the stationary period appears closer to t10.

3. Metrics at Stationary Region

We have found the warm up period by applying techniques defined in the previous section. In this section, we will capture some key metrics given below to understand the system's performance.

- Blocking Probability
- Mean customers in the system
- Mean time a customer spends in the system

Batch means technique is applied to simulate the system and capture the above metrics. Pseudo code is almost similar as the one given in section 2) except that the system counters will be reset once the batch threshold is reached and the system state will be captured at the end of the batch. One long simulation will be done and the cycle repeats for each batch.

Batch completion for System A is total customers > 2000 (exp time ~ 1000)

Batch completion for System B is total customers > 10000 (exp time ~ 1000)

Number of batches = 1000

3.1 Blocking Probability

Whenever a new customer arrives and the queue is full, we reject the customer and increment the blocking counter by 1. At the end of the batch, we compute the blocking probability as (*blocked_customers/total_customers)* . At the end of all the batches, we average the blocking probability to get the final score along with the confidence interval.

3.1.1. System A

For system A, the service rate and arrival rate are similar. Hence there are chances of customers being rejected as shown below but may not be very significant.

Blocking Probability for System A = **0.1326**

Confidence Interval at 90% confidence level = [0.1326-0.0008, 0.1326+0.008] = **[0.1318, 0.1334]**

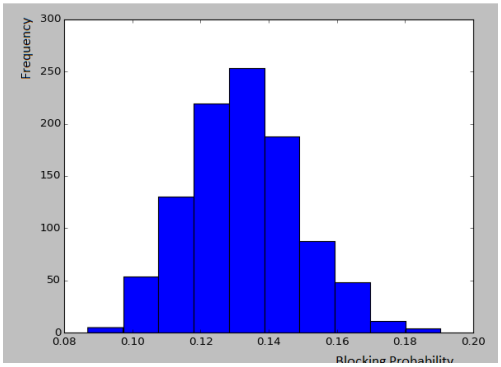


Fig 3.1.1 Distribution of blocking probability across all the batch runs for System A

3.1.2 System B

For system B, the arrival rate is much faster than service rate. Hence there are chances that most of the customers are rejected.

Blocking Probability for System B = **0.7999**

Confidence Interval at 90% confidence level = [0.7999-0.0003,0.7999+0.0003] = **[0.7996, 0.8002]**

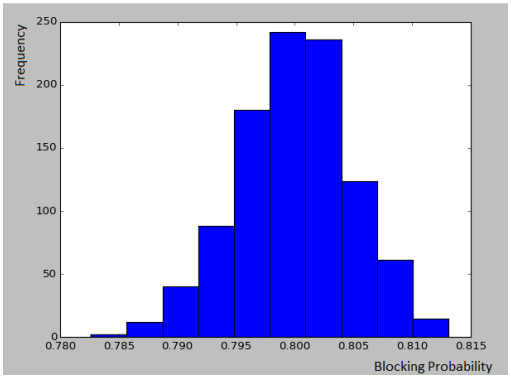


Fig 3.1.1 Distribution of blocking probability across all the batch runs for System B

3.2 Mean time a customer spends in the system

Whenever any customer leaves the system, We record the time that the customer spend in the system as (*departure_time – arrival_time)*. At the end of the batch, we compute the mean time a customer spends in the system by averaging out the time spent by each customer. At the end of all the batches, we average the mean spend time to get the final value along with the confidence interval.

3.2.1. System A

Mean Time Customer Spends in System A = **2.1516**

Confidence Interval at 90% confidence level = [2.1516-0.0054, 2.1516+0.0054] = **[2.1462, 2.157]**

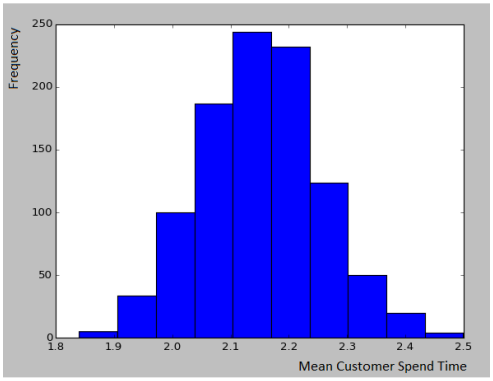


Fig 3.2.1 Distribution of Mean time customer spends in system for System A

3.2.2 System B

For system B, the queue would be long as arrival rate >> service rate. Hence the response time would be much greater than system A.

Mean Time Customer Spends in System B = **3.3766**

Confidence Interval at 90% confidence level = [3.3766-0.0041, 3.3766+0.0041] = **[3.3725, 3.3807]**

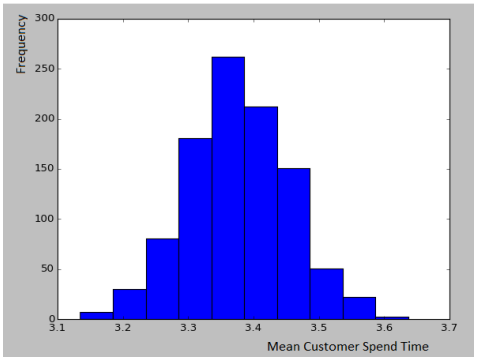


Fig 3.2.1 Distribution of Mean time customer spends in system for System B.

3.3 Mean number of customer in the system

We compute the mean number of customers in the system based on the system state and the duration of the system state. At the end of the batch, we compute the mean number of customers in the system by *summing up the time in each state * number of customers in that state / total time.*

3.3.1. System A

Mean Number of Customers in System A = **3.7282**

Confidence Interval at 90% confidence level = [3.7282-0.0085, 3.7282+0.0085] = [**3.7197, 3.7367]**

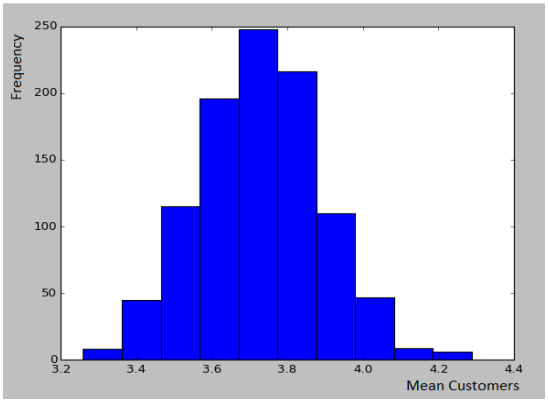


Fig 3.3.1 Distribution of mean customers across all batch runs for System A

3.3.2 System B

For system B, the queue would be almost full always as arrival rate >> service rate. Hence the mean number of customers is always close to 7 (i.e. system capacity).

Mean Number of Customers in System B = **6.7496**

Confidence Interval at 90% confidence level = [6.7496-0.0005, 6.7496+0.0005] **= [6.7491, 6.7501]**

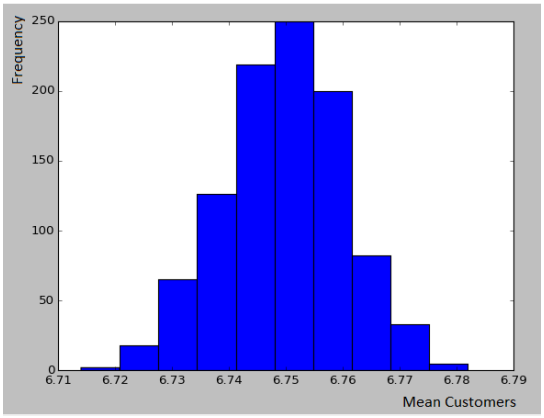


Fig 3.3.1 Distribution of mean customers across all batch runs for System B

4. Running the Simulator

4.1 Prerequisites

- Code written in python.
- Install Numpy and Matplotlib.

4.2 rngtest.py

Code located at \\HW2\code\

This is used to test the random number generator.

4.3 Simulator.py

Code located at \\HW2\code\

findWarmupPeriod – Simulates and Plots the graphical representation for the given queueing system.

findMetrics – Simulates and Prints the key metrics for the given queueing system.

4.4 Output Files

Output files are located at \\HW2\output\

2.1.out	Simulation output of HW 2.1. Each row represents single simulation run.
2.2.out	Simulation output of HW 2.2. Each row represents single simulation run.
2.3.out	Simulation output of HW 2.3. Each row represents single simulation run.
2.4.out	Simulation output of HW 2.4. Each row represents single simulation run.
3.1_SystemA_BlockingProbability.out	Simulation output of blocking Probability for system A
3.2_SystemA_MeanCustomerSpendTime.out	Simulation output of Mean customer spend time for system A
3.3_SystemA_MeanCustomers.out	Simulation output of Mean customers for system A
3.1_SystemB_BlockingProbability.out	Simulation output of blocking Probability for system B
3.2_SystemB_MeanCustomerSpendTime.out	Simulation output of Mean customer spend time for system B
3.3_SystemB_MeanCustomers.out	Simulation output of Mean customers for system B

----------END OF REPORT----------