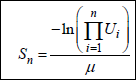
**Project 7: CTMC Bank Queue**

EE5011 Silvester – Bill Wang 9058118990

**Problem Statement:**

In this project, you are asked to simulate teller operations in a bank. The customer arrival process is Poisson (i.e. the interarrival times are iid exponentially distributed) with arrival rate lambda. 75% of the customers have a simple transaction to do; a simple transaction takes an amount of time that can be modeled by an Erlang-2 distribution with mean mu = 2 minutes. The other 25% of customers have complex transactions to do; complex transactions take an amount of time that can be modeled by an Erlang-5 distribution with mean mu = 6 minutes. Matlab has built-in capability to generate RV’s according to an Erlang distribution, but I would like you to generate the values using the technique explored in Project 6 part B, i.e.



Be sure to set n and mu appropriately to match the desired type of Erlang. You can assume that there are 3 tellers and an infinite amount of space is available to accommodate waiting customers. For strategy 1 and 2 assume that all tellers can handle any type of transaction. Consider the following 3 Queuing strategies.

1. There is a single queue. When a teller becomes free, the person at the head of the queue is served next.

2. Each teller has a separate queue. An arriving customer joins the shortest queue (and is not allowed to switch queues).

3. There are two queues. One for simple transactions and one for complex transactions. When a customer arrives, they join the appropriate queue. One teller (a relatively new hire) can handle only simple transactions, so is only able to accept customers from the simple transaction queue. The other two tellers are more experienced and can handle both simple and complex transactions. When an experienced teller becomes free and both queues are non-empty the teller randomly chooses to take the next customer from either queue. If only one of the queues has customers in it, then the teller accepts the next customer from that queue. If there are no customers in either queue, the teller becomes idle. When a customer arrives to find the queues empty and one or more tellers available, they randomly select a teller (subject to constraints on the type of service required, i.e. a customer with a complex transaction cannot be served by the new hire.)

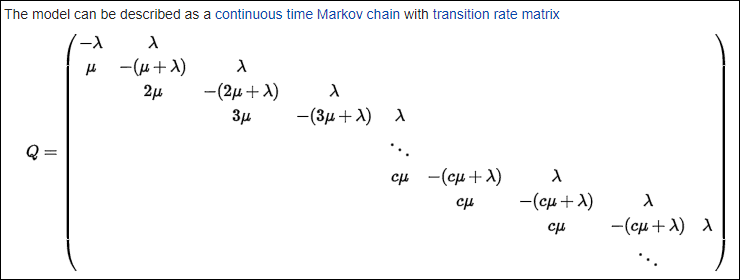
Simulate these systems and compare customer queue length distributions, waiting times (i.e. time spent waiting before being served), broken down by customer type for a variety of customer arrival rates. This should include estimates of the mean and variance of these statistics (and optionally estimate of the distribution of queue length, i.e. the pmf). Also, determine the fraction of time that the servers are idle (and any other statistics you find interesting.)

**Theoretical Analysis:**

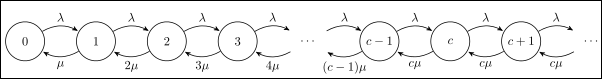
Scenario 1:

Scenario 1 is a standard M/M/C queue with C being the Erlang-C(mu) departure rate. Arrival rate is defined as a Poisson distribution with lambda arrival rate. The C and mu for the Erlang distribution is determined whether the arrival is a simple or complex transaction, calculated with a Bernoulli(0.75) RV. Lowercase c is defined as the number of servers working in parallel, in our case corresponding to the 3 bank tellers.

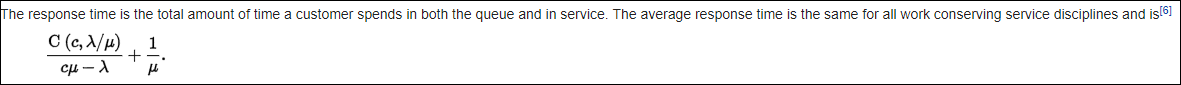
From Wikipedia:

The model can be described as a [continuous time Markov chain](https://en.wikipedia.org/wiki/Continuous_time_Markov_chain) with [transition rate matrix](https://en.wikipedia.org/wiki/Transition_rate_matrix) 

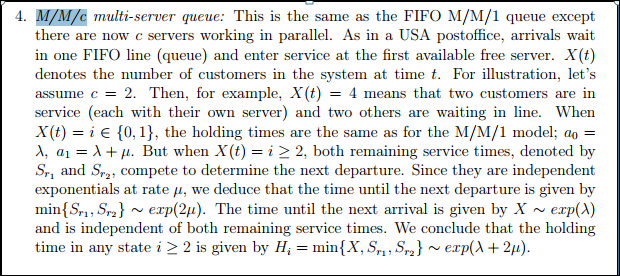
on the state space {0, 1, 2, 3, ...}. The model is a type of [birth–death process](https://en.wikipedia.org/wiki/Birth%E2%80%93death_process). We write *ρ* = *λ*/(*c μ*) for the server utilization and require *ρ* < 1 for the queue to be stable. *ρ* represents the average proportion of time which each of the servers is occupied (assuming jobs finding more than one vacant server choose their servers randomly).

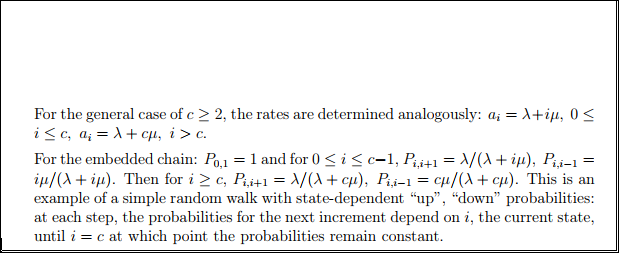
The [state space](https://en.wikipedia.org/wiki/State_space) diagram for this chain is as below. 

Hold time for customers is calculated:



Ross’s Simulation book also details this scenario with the following explanation, which reaches similar conclusions as the Wiki and provides some further insight to the simulation technique. Note that in Ross, the arrivals are and departures are distributed as exponentials, and so require some tweaking to fit our scenario.

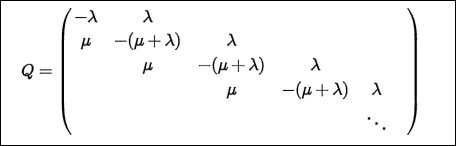




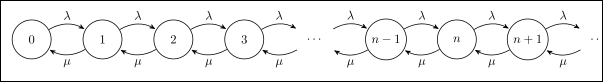
Scenario 2:

In the 2nd scenario, we have 3 separate queues. Because each teller is independent in this case (we determine the min buffer size prior to placing the arrivals in queue), there is no dependency on the other two queues once an arrival is placed. This independence allows us to assign three simultaneous M/M/1 queues to simulate the scenario. Similar to the first scenario, Wikipedia contains an explanation and analysis:

“The model can be described as a [continuous time Markov chain](https://en.wikipedia.org/wiki/Continuous_time_Markov_chain) with [transition rate matrix](https://en.wikipedia.org/wiki/Transition_rate_matrix)



on the state space {0,1,2,3,...}. This is the same continuous time Markov chain as in a [birth–death process](https://en.wikipedia.org/wiki/Birth%E2%80%93death_process). The [state space](https://en.wikipedia.org/wiki/State_space) diagram for this chain is as below:



The average response time or sojourn time (total time a customer spends in the system) does not depend on scheduling discipline and can be computed using [Little's law](https://en.wikipedia.org/wiki/Little%27s_law) as 1/(μ − λ). The average time spent waiting is 1/(μ − λ) − 1/μ = ρ/(μ − λ). The distribution of response times experienced does depend on scheduling discipline.”

At the end of our simulation, we can sum the results from the three queues, and divide by the total number of customer arrivals to obtain the net delay and other statistics.

Scenario 3:

In the 3rd scenario, there are two possible customer types (simple, complex) that wait in separate queues. For simulation, the queues can be further split into three scenarios.

The simple queue behaves as an M/M/3, similar to part 1. However, this can be further split due to the dedicated simple queue. Thus we consider one queue for simple only customers M/M/1. Another queue is defined as the total tellers serving simple customers. This is the M/M/3 queue. However, if there are complex customers, they can act as ‘blockers’ on the 2nd and 3rd tellers.

The complex customers can only be served as M/M/2, with ‘blockers’ being the simple customers also in the queue.

In both cases, the ‘buffer’ or ‘state’ is shared in that all arrivals cause it to go up by 1, and departures decrease by 1, regardless the transaction type.

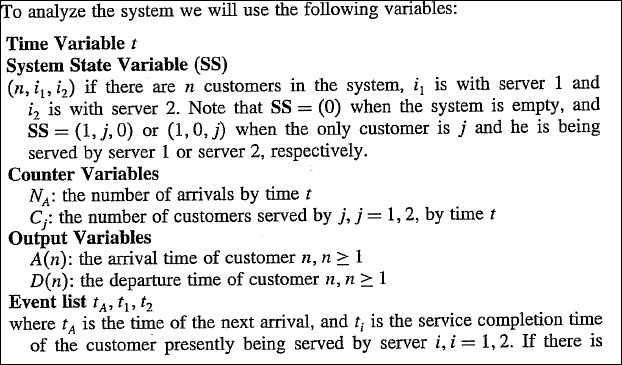
Further simulation steps outline found in the next section.

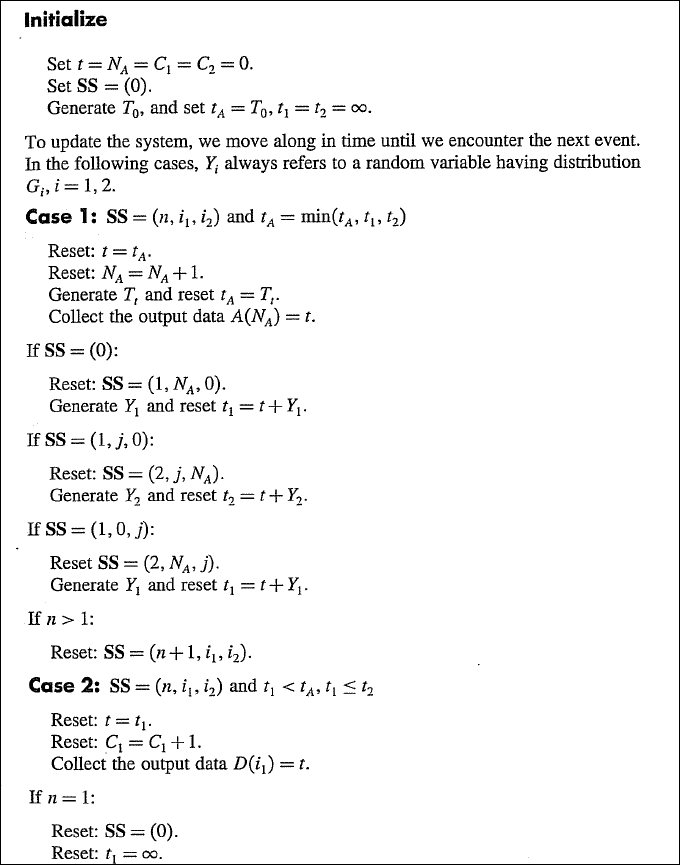
**Simulation Methodology:**

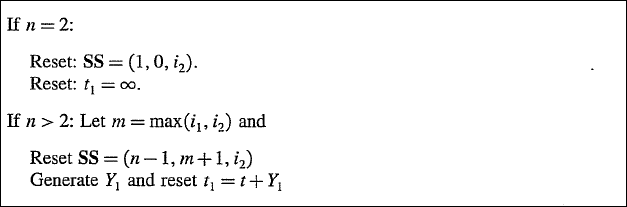
Scenario 1:

In the first scenario, there is a single queue for all customers. All three tellers service customers from this same queue in FIFO. This is considered an M/M/C queue where C is 2 or 5 conditional on whether the transaction is simple (Erlang-2) or complex (Erlang-5). We use the results from Lab 6 to generate multiplicative sums of Uniform RVs and find the inverse CDFs of Erlangs. This simulates our departure times.

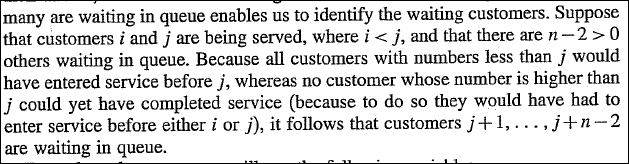
From Ross’s Simulation book, we can use the following strategy for 2 Servers, and extrapolate to the 3 Server case:







Difficulty here is keeping track of which departure is associated with which arrival. This is apparently resolved by having a system state variable SS(n, I, j, k) rather than just a counter of buffer size (see commented code for more information). Ross also gives the following hint which is used to select the next customer in queue.



We use the min and max of the system state variable, the count of customers in the system, and some algebra to determine this. This can then be used to populate the Customer Number variable for adding to the Output variables for arrival and departure times.

Delay and other statistics are calculated after the simulation is completed.

Scenario 2:

In the 2nd scenario, each teller has a separate queue. This is essentially 3 separate M/M/1 queues active at the same time. Thus, the simulation mirrors Professor’s example duplicated 3 times and run simultaneously, with a bit of code at the beginning for determining shortest queue (comparing system states of the 3 tellers). For this we generate three sets of all our variables to track the separate queues. In the end, we should have 3 different results of # of arrivals, (not necessarily the same size) where the sum of the arrivals should equal the runtime input of ‘Total Customers Served’. Then, we can calculate statistics on each individual queue to see how they perform in simulation, and also combine the results of each queue to get statistics on the net performance of the system.

Scenario 3:

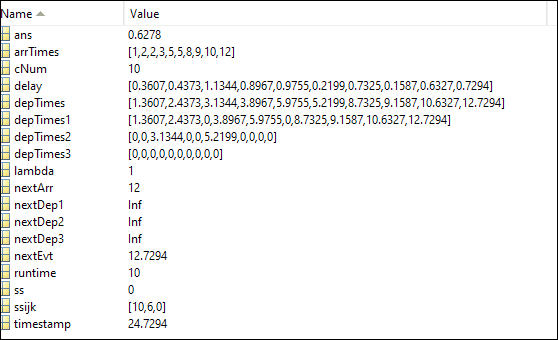
Outline of approach to Scenario 3 is detailed in the Theoretical Analysis. For simulation purposes, in terms of event driven simulation:

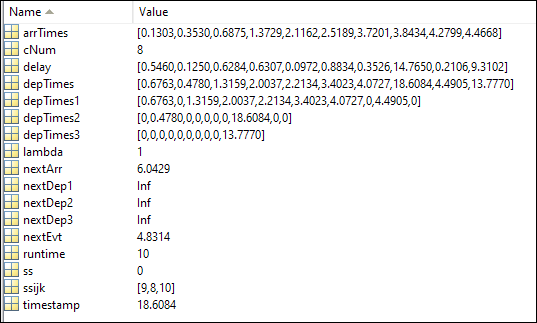
1. Customer arrival is generated with poisson (lambda)
2. Customer is categorized as simple or complex (bernoulli), departure time is calculated if agents are available OR
3. Added to appropriate buffer if no available agents
4. Longest waiting Customer arrival time **for each queue** is compared to departure times of all ongoing transactions
5. Next departure time is determined and agent type of that departure is used to push the next queued customer of appropriate type (simple, complex, rand logic) to newly available agent.
6. Repeat steps with while loop until # arrivals = max Customers to be served.

**Experiment:**

Part 1:

Experimental results are as follows:





See sample results in screenshots:

**Conclusion:**

Part 1: One issue with the results is that the poissrnd Matlab function used to calculate arrival times appears to return an integer value (which makes sense as Poisson is a discrete RV). Thus, with minutes as our timescale, it was possible to get arrival times truncated to 0. One way to correct for this would be to update all times by a factor of 60, to convert to seconds. While this would still truncate to integer seconds, in terms of the problem (customers), 1 second is a much smaller difference than 1 minute arrivals.

Part 2: Inconclusive

This lab was by far the most intensive of all, and also in my opinion the most valuable. Coding chops were developed as well as a deeper understanding of the event driven simulation methodologies. Extending the MM1, MMC queue code snippets to the scenarios mentioned in the problem were some brain gymnastics that I feel definitely allowed me to learn.

Would like hash out the remaining code in the project and to discuss in office hours if possible.

**Reference:**

Wikipedia: M/M/C Queue [https://en.wikipedia.org/wiki/M/M/c\_queue]

Wikipedia: M/M/1 Queue [https://en.wikipedia.org/wiki/M/M/1\_queue]

Wikipedia: Discrete Event Simulation [https://en.wikipedia.org/wiki/Discrete\_event\_simulation]

EE511: Lecture 12 - Discrete Event Simulation – Silvester

Columbia – CTMC [http//www.columbia.edu/~ks20/stochastic-I/stochastic-I-CTMC.pdf]

Simulation CH 6.4 Queuing System with 2 Parallel Servers -- Ross

**Commented Source Code:**

%{

Bill (Belal) Wang, 9058118990

EE511 Project 7: CTMC Bank Queue

%}

%-------------Scenario 1: Single Queue--------------

%-------------Initialization------------------------

%Simulation time (minutes), System State

%(buffer size, server1 customer #, server2 customer #, server3 customer #), Counters (# arrivals, total

%customers served by servers 1, 2, 3), Output vars: Arrival and Departure

%matrices, next events (arrival, departures 1, 2, 3)

timestamp=0; %tracks timestamps

ss=0; %systemstate(n customers in system, i customer in s1, j customer in s2, k customer in s3)

%ss=number Cs in system

ssijk=zeros(1,3);

%ssijk(1)=cNum at server 1

%ssijk(2)=cNum at server 2

%ssijk(3)=cNum at server 3

cNum=0; %customer number, similar to arrival#

%input number of arrivals/departure to simulate:

runtime=input('Max customer arrivals (simulation time): ');

lambda=input('Customer Arrival Rate (Poisson): ');

arrTimes=zeros(1,runtime);

depTimes1=zeros(1,runtime);

depTimes2=zeros(1,runtime);

depTimes3=zeros(1,runtime);

depTimes=zeros(1,runtime);

%initialize first arrival

nextArr=poisTime(lambda);

timestamp=nextArr;%next departures need to be greater for first run

nextDep1=inf;

nextDep2=inf;

nextDep3=inf;

%main loop

while cNum < runtime

nextEvt=min([nextArr,nextDep1,nextDep2,nextDep3])

%case 1, arrTime min -- arrival

if nextEvt==nextArr

%cNum=cNum+1; %count arrivals

%determine last position in system to add arrival

if (ss<4)

cNum=max(ssijk)+1

else

cNum=max(ssijk)+ss-3

end

%cNum=max(ss(1,:)) + ss(1,0) - 3 + 1;

%%%timestamp = timestamp+nextArr; %record arrival time

timestamp=nextArr;

%%%arrTimes(cNum)=timestamp; %set timestamp of event

arrTimes(cNum)=nextArr;

%actions based on system state ss

if ss<3 %open slots, fewer than 3 custs

%push arrival to next open slot

%if min(ssijk)==ssijk(1) %slot 1 has min cNum

if nextDep1==inf

ssijk(1)=cNum;

if bernTime(0.75)==1

%%%nextDep1=timestamp-erlangTime(2,5);

nextDep1=nextArr-erlangTime(2,5);

else

%%%nextDep1=timestamp-erlangTime(5,6);

nextDep1=nextArr-erlangTime(5,6);

end

%nextDep1=nextArr+nextDep1;

%elseif min(ssijk)==ssijk(2) %slot 2 has min cNum

elseif nextDep2==inf

ssijk(2)=cNum;

if bernTime(0.75)==1

%%%nextDep1=timestamp-erlangTime(2,5);

nextDep2=nextArr-erlangTime(2,5);

else

%%%nextDep1=timestamp-erlangTime(5,6);

nextDep2=nextArr-erlangTime(5,6);

end

%nextDep2=nextArr+nextDep2;

%elseif min(ssijk)==ssijk(3) %slot 3 has min cNum

elseif nextDep3==inf

ssijk(3)=cNum;

if bernTime(0.75)==1

%%%nextDep1=timestamp-erlangTime(2,5);

nextDep3=nextArr-erlangTime(2,5);

else

%%%nextDep1=timestamp-erlangTime(5,6);

nextDep3=nextArr-erlangTime(5,6);

end

%nextDep3=nextArr+nextDep3;

end

else %no slots open -- buffer arrival

%ss(1,0) = ss(1,0)+1; %increase n

end

%calculate time of next arrival

nextArr=nextArr+poisTime(lambda);

ss = ss + 1; %increase n system count

%cNum=cNum+1; %increase arrival count

timestamp

ss

ssijk

cNum

%case 2, depTime1 min -- departure at teller1

elseif nextEvt==nextDep1

cNum=ssijk(1); %%set customer # to customer in teller position

depTimes1(cNum)=nextDep1; %update output table

depTimes(cNum)=nextDep1;

ss=ss - 1; %remove Cs from the system

if ss < 3 %if no customers in queue (less than 3 in system would all be serviced)

nextDep1=inf; %set dep to high for min calculation in loop

else %customers waiting

ssijk(1)=max(ssijk) + 1; %pick next buffer - max customer in process + 1, set to first server

if bernTime(0.75)==1

nextDep1=nextDep1-erlangTime(2,5);

else

nextDep1=nextDep1-erlangTime(5,6);

end

end

timestamp

ss

ssijk

cNum

%case 3, depTime2 min -- departure at teller2

elseif nextEvt==nextDep2

timestamp=timestamp+nextDep2; %update timestamp

cNum=ssijk(2); %%set customer # to customer in teller position

depTimes2(cNum)=nextDep2; %update output table

depTimes(cNum)=nextDep2;

ss=ss - 1; %remove Cs from the system

if ss < 3 %if no customers in queue (less than 3 in system would all be serviced)

nextDep2=inf; %set dep to high for min calculation in loop

else %customers waiting

ssijk(2)=max(ssijk) + 1; %pick next buffer - max customer in process + 1, set to first server

if bernTime(0.75)==1

nextDep2=nextDep2-erlangTime(2,5);

else

nextDep2=nextDep2-erlangTime(5,6);

end

end

timestamp

ss

ssijk

cNum

%{

cNum=ss(1,2); %%set customer # to customer in teller position

ss(1,0)=ss(1,0)-1; %remove Cs from the system

if ss(1,0)==0 %if no customers in system

nextDep2=inf;

end

%}

%case 4, depTime3 min -- departure at teller3

elseif nextEvt==nextDep3

timestamp=timestamp+nextDep3; %update timestamp

cNum=ssijk(3); %%set customer # to customer in teller position

depTimes3(cNum)=nextDep3; %update output table

depTimes(cNum)=nextDep3;

ss = ss - 1; %remove Cs from the system

if ss < 3 %if no customers in queue (less than 3 in system would all be serviced)

nextDep3=inf; %set dep to high for min calculation in loop

else %customers waiting

ssijk(2)=max(ssijk) + 1; %pick next buffer - max customer in process + 1, set to first server

if bernTime(0.75)==1

nextDep3=nextDep3-erlangTime(2,5);

else

%nextDep3=timestamp-erlangTime(5,6);

nextDep3=nextDep3-erlangTime(5,6);

end

end

timestamp

ss

ssijk

cNum

end

end

%cleanup remaining customers

while (ss>0) %while customers are remaining in system

nextEvt=min([nextDep1,nextDep2,nextDep3])

%{

if (ss<4) %3 customers remaining, buffer is empty

cNum=max(ssijk)+1; %select customer at

else %buffer not empty

cNum=max(ssijk)+ss-3; %select customer at

end

%}

%case 1, depTime1 min -- departure at teller1

if nextEvt==nextDep1

timestamp=timestamp+nextDep1; %update timestamp

cNum=ssijk(1); %%set customer # to customer in teller position

depTimes1(1,cNum)=nextDep1; %update output table

depTimes(1,cNum)=nextDep1;

ss = ss - 1; %remove Cs from the system

if ss < 3 %if no customers in queue (less than 3 in system would all be serviced)

nextDep1=inf; %set dep to high for min calculation in loop

else %customers waiting

ssijk(1)=max(ssijk) + 1; %pick next buffer - max customer in process + 1, set to first server

if bernTime(0.75)==1

nextDep1=nextDep1-erlangTime(2,5);

else

nextDep1=nextDep1-erlangTime(5,6);

end

end

%case 2, depTime2 min -- departure at teller2

elseif nextEvt==nextDep2

timestamp=timestamp+nextDep2; %update timestamp

cNum=ssijk(2); %%set customer # to customer in teller position

depTimes2(1,cNum)=timestamp; %update output table

depTimes(1,cNum)=timestamp;

ss = ss - 1; %remove Cs from the system

if ss < 3 %if no customers in queue (less than 3 in system would all be serviced)

nextDep2=inf; %set dep to high for min calculation in loop

else %customers waiting

ssijk(2)=max(ssijk) + 1; %pick next buffer - max customer in process + 1, set to first server

if bernTime(0.75)==1

nextDep2=nextDep2-erlangTime(2,5);

else

nextDep2=nextDep2-erlangTime(5,6);

end

end

%case 3, depTime3 min -- departure at teller3

elseif nextEvt==nextDep3

timestamp=timestamp+nextDep3; %update timestamp

cNum=ssijk(3); %%set customer # to customer in teller position

depTimes3(1,cNum)=timestamp; %update output table

depTimes(1,cNum)=timestamp;

ss = ss - 1; %remove Cs from the system

if ss < 3 %if no customers in queue (less than 3 in system would all be serviced)

nextDep3=inf; %set dep to high for min calculation in loop

else %customers waiting

ssijk(2)=max(ssijk) + 1; %pick next buffer - max customer in process + 1, set to first server

if bernTime(0.75)==1

nextDep3=nextDep3-erlangTime(2,5);

else

nextDep3=nextDep3-erlangTime(5,6);

end

end

end

end

%calculate statistics

delay=0;

%--------Functions for inverse distributions------------

%determine transaction type (1=simple, 0=complex)

function b = bernTime(p)

rngU=rand();

if rngU<p

b=1;

else

b=0;

end

end

%determine next customer arrival time

function p =poisTime(lambda)

p=poissrnd(lambda);

end

%determine erlang transaction time

function e = erlangTime(k,mu)

multiSum=rand();

for iExpCount=1:k-1

multiSum=multiSum\*rand();

end

e=(-1/mu)\*(-log(multiSum));

end

Part 2:

%{

Bill (Belal) Wang, 9058118990

EE511 Project 7: CTMC Bank Queue

%}

%{

Scenario 2: Simulating Bank Queue with following parameters -

Given: All times in minutes

Given: 3 Tellers

-Customer arrival distributed as random Poisson(lambda) process

-Each customer has a random {simple, complex} transaction time

~Bernoulli(p), p=0.75 for simple

-{simple} - Erlang(x, 2, 2)

-{complex} - Erlang(x, 5, 6)

Scenarios:

1. Single queue, customer goes to first free teller

2. Separate queue for each teller (arriving customer joins shortest queue)

3. Queues for simple/complext transactions. 1 Teller for simple, 2 for

complex

%}

%------------3 Teller Separate Queue Scenario-------------

%initialization

buffer1=0;

buffer2=0;

buffer3=0; %number of customers in queue (including currently being served) -- SS

simTime1=0;

simTime2=0;

simTime3=0; %simulation time in minutes (can be fractional)

%number of events (arrival, departure) to simulate for

totalCustArrivals=input('Max customers to be serviced: ');

lambda=input('Customer Arrival Rate (Poisson): ');

%result containers

arrTime1=zeros(1, int8(totalCustArrivals/3)+1);

arrTime2=zeros(1, int8(totalCustArrivals/3)+1);

arrTime3=zeros(1, int8(totalCustArrivals/3)+1);

depTime1=zeros(1, int8(totalCustArrivals/3)+1);

depTime2=zeros(1, int8(totalCustArrivals/3)+1);

depTime3=zeros(1, int8(totalCustArrivals/3)+1);

%delay1=zeros(1, totalCustArrivals/3+1);

%delay2=zeros(1, totalCustArrivals/3+1);

%delay3=zeros(1, totalCustArrivals/3+1);

%Initialize event counters

totalArr=0;

arrNum1=0;

arrNum2=0;

arrNum3=0;

depNum1=0;

depNum2=0;

depNum3=0;

%Bernoulli p=0.75

p=0.75;

%Initialize arrival times 1-3

nextArrTime1 = expRV(lambda);

nextArrTime2 = expRV(lambda);

nextArrTime3 = expRV(lambda);

%artificial departure times

nextDepTime1 = nextArrTime1+1;

nextDepTime2 = nextArrTime2+1;

nextDepTime3 = nextArrTime3+1;

%main loop, runs until max number of arriving customers is reached

while totalArr < totalCustArrivals

if(min([buffer1, buffer2, buffer3])==buffer1) %teller 1 shortest

%teller 1 queue logic

%calc next arrival time

%nextArrTime1=poisPeriod(lambda);

%calc next departure time dependent on complexity

%{

if transactionType(p)==1 %simple transaction

nextDepTime1=erlangPeriod(2,2);

else %complex transaction

nextDepTime1=erlangPeriod(5,6);

end

%}

if(nextArrTime1<nextDepTime1) %arrival

arrNum1=arrNum1+1;

simTime1=nextArrTime1;

arrTime1(arrNum1)=simTime1;

if(buffer1==0)

if transactionType(p)==1 %simple transaction

nextDepTime1=simTime1-erlangTime(2,2);

else %complex transaction

nextDepTime1=simTime1-erlangTime(5,6);

end

%nextDepTime1=simTime1+nextDepTime1;

end

totalArr=totalArr+1;

buffer1=buffer1+1;

nextArrTime1=simTime1+expRV(lambda);

else %departure

depNum1=depNum1+1;

simTime1=nextDepTime1;

depTime1(depNum1)=simTime1;

buffer1=buffer1-1;

if buffer1>0

if transactionType(p)==1 %simple transaction

nextDepTime1=simTime1-erlangTime(2,2);

else %complex transaction

nextDepTime1=simTime1-erlangTime(5,6);

end

else

nextDepTime1=nextArrTime1+1; % artificial

end

%totalArr=totalArr+1;

end

elseif (min([buffer1, buffer2, buffer3])==buffer2) %teller 2 shortest

%teller 2 queue logic

%calc next arrival time

if(nextArrTime2<nextDepTime2) %arrival

arrNum2=arrNum2+1;

simTime2=nextArrTime2;

arrTime2(arrNum2)=simTime2;

if(buffer2==0)

if transactionType(p)==1 %simple transaction

nextDepTime2=simTime2-erlangTime(2,2);

else %complex transaction

nextDepTime2=simTime2-erlangTime(5,6);

end

end

totalArr=totalArr+1;

buffer2=buffer2+1;

nextArrTime2=simTime2+expRV(lambda);

else %departure

depNum2=depNum2+1;

sim\_time2=nextDepTime2;

depTime2(depNum2)=simTime2;

buffer2=buffer2-1;

if buffer2>0

if transactionType(p)==1 %simple transaction

nextDepTime2=simTime2-erlangTime(2,2);

else %complex transaction

nextDepTime2=simTime2-erlangTime(5,6);

end

else

nextDepTime2=nextArrTime2+2; % artificial

end

%totalArr=totalArr+1;

end

elseif (min([buffer1, buffer2, buffer3])==buffer3) %teller 3 shortest

%teller 3 queue logic

if(nextArrTime3<nextDepTime3) %arrival

arrNum3=arrNum3+1;

simTime3=nextArrTime3;

arrTime3(arrNum3)=simTime3;

if(buffer3==0)

if transactionType(p)==1 %simple transaction

nextDepTime3=simTime3-erlangTime(2,2);

else %complex transaction

nextDepTime3=simTime3-erlangTime(5,6);

end

end

totalArr=totalArr+1;

buffer3=buffer3+1;

nextArrTime3=simTime3+expRV(lambda);

else %departure

depNum3=depNum3+1;

sim\_time3=nextDepTime3;

depTime3(depNum3)=simTime3;

buffer3=buffer3-1;

if buffer3>0

if transactionType(p)==1 %simple transaction

nextDepTime3=simTime3-erlangTime(2,2);

else %complex transaction

nextDepTime3=simTime3-erlangTime(5,6);

end

else

nextDepTime3=nextArrTime3+2; % artificial

end

%totalArr=totalArr+1;

end

end

end

%cleanup buffer of remaining customers in line

%case1

while buffer1>0

depNum1=depNum1+1;

simTime1=nextDepTime1;

depTime1(depNum1)=simTime1;

buffer1=buffer1-1;

if buffer1>0

nextDepTime1=simTime1-nextDepTime1;

end

end

%case2

while buffer2>0

depNum2=depNum2+1;

simTime2=nextDepTime2;

depTime2(depNum2)=simTime2;

buffer2=buffer2-1;

if buffer2>0

nextDepTime2=simTime2-nextDepTime2;

end

end

%case3

while buffer3>0

depNum3=depNum3+1;

simTime3=nextDepTime3;

depTime3(depNum3)=simTime3;

buffer3=buffer3-1;

if buffer3>0

nextDepTime3=simTime3-nextDepTime3;

end

end

%output

delay1=depTime1-arrTime1;

delay2=depTime2-arrTime2;

delay3=depTime3-arrTime3;

sumDelay1=0;

sumDelay2=0;

sumDelay3=0;

for cNum1=1:depNum1%case1

sumDelay1=sumDelay1+delay1(cNum1);

end

meanDelay1=sumDelay1/depNum1;

for cNum2=1:depNum2%cas2

sumDelay2=sumDelay2+delay2(cNum2);

end

meanDelay2=sumDelay2/depNum2;

for cNum3=1:depNum3%case3

sumDelay3=sumDelay3+delay3(cNum3);

end

meanDelay3=sumDelay3/depNum3;

totalDelay=sum([meanDelay1, meanDelay2, meanDelay3])/3;

%--------Functions for inverse distributions------------

%determine transaction type (1=simple, 0=complex)

function b = transactionType(p)

rngU=rand();

if rngU<p

b=1;

else

b=0;

end

end

%determine next customer arrival time

function p =poisTime(lmbda)

%rngU=rand();

%p=poissinv(rngU, lambda);

p=poissrnd(lmbda)

end

function ex = expRV(lambda)

ex=exprnd(1/lambda);

end

%determine erlang transaction time

function e = erlangTime(k,mu)

multiSum=rand();

for iExpCount=1:k-1

multiSum=multiSum\*rand();

end

e=(-1/mu)\*(-log(multiSum));

end