Project 3

in Stochastic processes and simulation 1

Mona Hazeem & Simon Jorstedt

2021-03-06

Opening

We shall imagine ourselves at the institution of experimental economics. The institution has a computer that solves difficult computational tasks. Tasks arrive to the computer according to a Poisson process with intensity λ. The time it takes for the computer to perform a task is exponentially distributed with intensity μ. If the computer is working on a problem, a maximum of two other tasks can be queued at the same time.

In order to avoid bottlenecks, the board of the institution want the queue to be full no more than 5% of the time. There are three suggestions on who should have access to the computer. Each suggestion leads to different intensity of tasks arriving at the computer.

Suggestion 1: Only professors may use the computer. In this case we have $\lambda = 2$ and $\mu = 10$.

Suggestion 2: Professors and students may use the computer. In this case we get $\lambda = 6$ and $\mu = 10$.

Suggestion 3: Everyone, including the general public, is allowed to use the computer. Then $\lambda = 10$ and $\mu = 10$.

This report will seek to investigate which proposal(s) fulfil the 5% requirement.

Problem 1

The queue process to the computer can be seen as an $M|M|1$-queue. An $M|M|1$-queue is a type of birth-death process with one "server", in this case the computer performing tasks. We note that tasks are added according to an exponential distribution with intensity λ, unless there is a maximum number of tasks in the system (in this case 3). We also note that tasks are performed according to an exponential distribution with intensity μ, as long as there are tasks left in the system (state 0).

The time spent in each of the states 1 and 2 is exponentially distributed with intensity $\mu + \lambda$. In state 0 the time spent is exponentially distributed with intensity λ, and in state 3 the time spent is exponentially distributed with intensity μ.

The probability that the number of tasks in the queue increases is 1 if $i = 0$ (i.e. there are no elements in the queue, so an increase is the only option here).

In the case of $i = 1$ or $i = 2$, the probability of the amount increasing is described by $P_{i,i+1} = \frac{\lambda}{\lambda+\mu}$.

In the case of $i = 3$, the probability of an increase is 0, since the system has reached its maximum capacity. The probability of decrease in state 3 is 1, since no elements can be added. The probability of decrease in conditions 1 and 2 is described by $P_{i,i-1} = \frac{\mu}{\lambda+\mu}$.

According to suggestion 1, we have $\mu = 10$ and $\lambda = 2$, except in state 3, when $\lambda = 0$, and in state 0 when $\mu = 0$. This is because when we are in state 3, the queue cannot increase, and then no elements can be added;

and when we are in state 0, the amount cannot decrease. This means that each visit in state 0 will last $\frac{1}{2}$ hours on average. The same way, each visit to either state 1 or 2 will last $\frac{1}{12}$ hours on average, and each visit in state 3 will last $\frac{1}{10}$ on average. The increase probability is $P_{0,1} = 1$, meanwhile $P_{1,2} = P_{2,3} = \frac{2}{12} = \frac{1}{6}$ The decrease probability is $P_{3,2} = 1$, $P_{2,1} = P_{1,0} = \frac{10}{12} = \frac{5}{6}$.

Problem 2

Bellow we have the required code:

```
bd_process <- function(lambda, mu, initial_state = 0, steps = 100) {

time_now <- 0
state_now <- initial_state

#These vectors will be developed in the following loop
time <- 0
state <- initial_state

for (i in 1:steps) {

  # In state 3, the queue is full. We set lambda = 0.
  if (state_now == 3) {
    lambda_now <- 0
  } else {
    lambda_now <- lambda
  }


  # In state 0, the system is empty. Once a task arrives it
  #directly starts with no delay or queuing. We set mu = 0.
  if (state_now == 0) {
    mu_now <- 0
  } else {
    mu_now <- mu
  }

  # We generate a uniformly distributed random variable in[0,1].
  uniform <- runif(1)

  # Time_to_transition describes the time taken to the next
  #change of state. Either an increse or a decrease.
  time_to_transition <- rexp(1,rate=lambda_now + mu_now)

  # If the uniformly distributed variable is smaller than
  #mu /(lambda+mu), the queue decrease. Else it increases.
  if (uniform < (mu_now/(mu_now + lambda_now))) {
    state_now <- state_now - 1
  } else {
    state_now <- state_now + 1
  }
```

```
  #time_now is the current time, when the state changes.
  time_now <- time_now + time_to_transition

  #the vector "time" contains all times the system has changed.
  #This argument is similar to "append" in Python.
  time <- c(time, time_now)


  #the vector "state" contains all states in the respective
  #order the system entered them.
  state <- c(state, state_now)
}

#Combined list of the vectors time and state.
list(time = time, state = state)
}
```

Problem 3

Below are the required diagrams that show which states are adopted at which time for each of the three different suggestions.

```
#Simulating suggestion 1 (100 steps)
set.seed(19900924)
suggetion1 <- bd_process(lambda = 2, mu = 10)
time1 <- suggetion1$time
state1 <- suggetion1$state

#Simulating suggestion 2 (100 steps)
set.seed(19900924)
suggestion2 <- bd_process(lambda = 6, mu = 10)
time2 <- suggestion2$time
state2 <- suggestion2$state

#Simulating suggestion 3 (100 steps)
set.seed(19900924)
suggestion3 <-bd_process(lambda = 10, mu = 10)
time3 <- suggestion3$time
state3 <- suggestion3$state


#Plotting results of suggestion 1 (100 steps)
plot(stepfun(time1[-1], state1),
     do.points = FALSE,
     xlab = "Time (hours)",
     ylab = "State",
     main = "Figure 1.1: Suggetion 1, 100 steps",
     yaxt = "n")
axis(2, at =c(0, 1, 2, 3), las = 2)

#Plotting results of suggestion 2 (100 steps)
plot(stepfun(time2[-1], state2),
```

```
     do.points = FALSE,
     xlab = "Time (hours)",
     ylab = "State",
     main = "Figure 1.2: Suggetion 2, 100 steps",
     yaxt = "n")
axis(2, at =c(0, 1, 2, 3), las = 2)

#Plotting results of suggestion 3 (100 steps)
plot(stepfun(time3[-1], state3),
     do.points = FALSE,
     xlab = "Time (hours)",
     ylab = "State",
     main = "Figure 1.3: Suggetion 3, 100 steps",
     yaxt = "n")
axis(2, at =c(0, 1, 2, 3), las = 2)
```

Figure 1.1: Suggetion 1, 100 steps

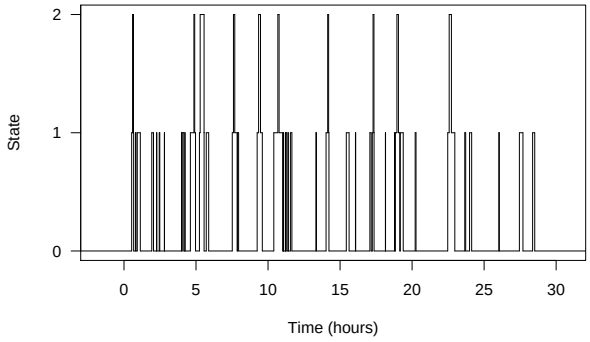


Figure 1.2: Suggetion 2, 100 steps

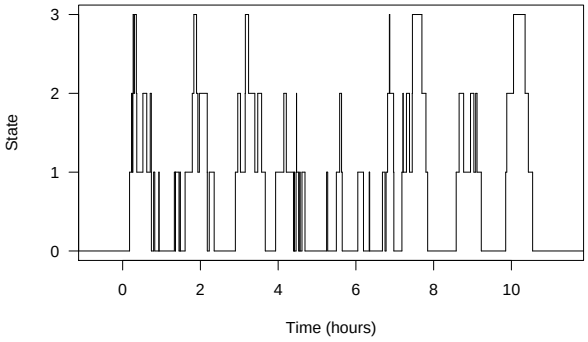
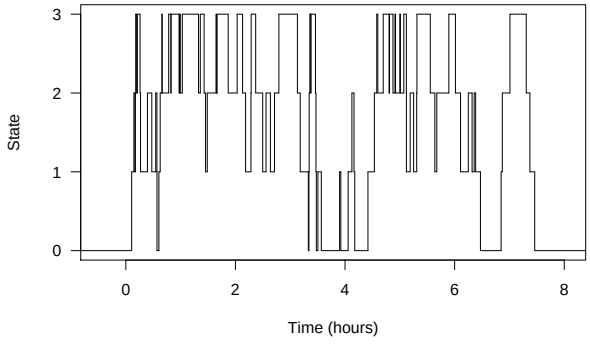


Figure 1.3: Suggetion 3, 100 steps



We notice in figure 1 that suggestion 1 is lasting for the longest time while suggestion 3 is lasting for the shortest time. This is not strange because the waiting time between states becomes shorter the more elements that line up. More specifically, in an exponential distribution, the times between events become shorter the higher the intensity is, and the intensity for changing the state is $\lambda+\mu$, so that when λ increases, the intensity increases. This shortens the time in between the events.

Another difference is that in suggestion 1 the chain is more often in states 0 and 1, in comparison with suggestion 2 and suggestion 3. We note in figure 1.3 that with suggestion 3, the system is more often in statess 2 and 3 than in the other suggestions. This is natural, because if queuing elements become more frequent, the queue will also tend to be longer.

Problem 4

```
#Simulating suggestion 1 (500 steps)
set.seed(19900924)
suggetion1_v2 <- bd_process(lambda = 2, mu = 10,steps=500)
time1_v2 <- suggetion1_v2$time
state1_v2 <- suggetion1_v2$state

#Simulating suggestion 2 (500 steps)
set.seed(19900924)
suggetion2_v2 <- bd_process(lambda = 6, mu = 10,steps=500)
time2_v2 <- suggetion2_v2$time
state2_v2 <- suggetion2_v2$state

#Simulating suggestion 3 (500 steps)
set.seed(19900924)
suggetion3_v2 <- bd_process(lambda = 10, mu = 10,steps=500)
time3_v2 <- suggetion3_v2$time
state3_v2 <- suggetion3_v2$state




df <- data.frame("Suggetion" = "Time (hours)")
df <- cbind(df, cbind(time1_v2[500], time2_v2[500], time3_v2[500]))
names(df)[-1] <- c(1,2,3)


knitr::kable(df,digits=0, caption = "Time when we reach 500 states", label=NA)
```

Table 1: Time when we reach 500 states

Suggetion	1	2	3
Time (hours)	126	46	33

Problem 5

5.1

Now we are going to write a function that goes through a process and calculates the proportion of time that the process spends in different states. The code for this function follows below.

```
proportion_in_state <- function(s, bdp) {
  #We will use "tot_time" to count time spent in
  #the given state s. Starting at 0.
  tot_time <- 0

  #We go through all states the simulation passes.
  #Not including the last state.
```

```
  for (i in 1:(length(bdp$state)-1)) {

    #If we are in state s
    if(bdp$state[i]==s){

      #We keep track of how long we've been in state s,
      #by taking the difference of the time we
      #arrived at the next state and the time we
      #arrived at state s.
      tot_time <- tot_time + bdp$time[i+1]-bdp$time[i]
  }
  }

#we divide the time when the chain has been in the 'right' state
#with the total time the chain has been active. We multiply with
#100 to get the percentage.
return(100*(tot_time/bdp$time[length(bdp$time)]))
}
```

5.2

Now we'll let the process go 1000 steps forward, for all the three suggestions. Then, using the function created in the previous exercise, we will calculate the proportion of time that the chains spend in the various states. The results are presented in a table below.

```
#Simulating suggestion 1 (1000 steps)
set.seed(19900924)
suggestion1_v3 <-bd_process(lambda = 2, mu = 10,steps=1000)
time1_v3 <- suggestion1_v3$time
state1_v3 <- suggestion1_v3$state

#Simulating suggestion 2 (1000 steps)
set.seed(19900924)
suggestion2_v3 <-bd_process(lambda = 6, mu = 10,steps=1000)
time2_v3 <- suggestion2_v3$time
state2_v3 <- suggestion2_v3$state

#Simulating suggestion 3 (1000 steps)
set.seed(19900924)
suggestion3_v3 <- bd_process(lambda = 10, mu = 10, steps=1000)
time3_v3 <- suggestion3_v3$time
state3_v3 <- suggestion3_v3$state
```

```
#Vector of results from simulating suggestion 1 (1000 steps)
sgstn1 <- c(proportion_in_state(0,suggestion1_v3),
            proportion_in_state(1,suggestion1_v3),
            proportion_in_state(2,suggestion1_v3),
            proportion_in_state(3,suggestion1_v3))

#Vector of results from simulating suggestion 2 (1000 steps)
sgstn2 <- c(proportion_in_state(0,suggestion2_v3),
            proportion_in_state(1,suggestion2_v3),
```

```
            proportion_in_state(2,suggestion2_v3),
            proportion_in_state(3,suggestion2_v3))

#Vector of results from simulating suggestion 3 (1000 steps)
sgstn3 <- c(proportion_in_state(0,suggestion3_v3),
            proportion_in_state(1,suggestion3_v3),
            proportion_in_state(2,suggestion3_v3),
            proportion_in_state(3,suggestion3_v3))

#Combinating immediately preceding results
proportions <- cbind("State" = 0:3,
                     "Suggestion 1" = sgstn1,
                     "Suggestion 2" = sgstn2,
                     "Suggestion 3" = sgstn3)

#Plotting a table of the results
knitr::kable(proportions,
             digits = 3,
             caption = "The distribution of time (in percentages) spent in each state for each suggestio
```

Table 2: The distribution of time (in percentages) spent in each state for each suggestion.

State	Suggestion 1	Suggestion 2	Suggestion 3
0	80.064	45.706	22.962
1	16.013	28.118	24.755
2	3.237	16.819	26.272
3	0.686	9.356	26.011

In table 2, we see that only suggestion 1 satisfies the institutions requirement that the queue may be full for a maximum of 5% of the time. However, we also see that with suggestion 1, the computer will sit idly - not doing anything - for an estimated 80% of the time.

Problem 6

We shall now look at the stationary distribution of this system and what the long term theoretical probabilities are. Previously in the course lectures we arrived at the following results for the stationary probabilities P_0 and P_n. They represent the probability of being in some state $n = 0, 1, 2, 3$ in the arbitrarily distant future, or equivalently, the portion of time spent in each state on average. The factor $\rho = \frac{\lambda}{\mu}$ occurs often and is thus compacted with ρ.

$$P_0 = \frac{1-\rho}{1-\rho^{N+1}}$$

$$P_n = \rho^n \cdot \frac{1-\rho}{1-\rho^{N+1}}$$

$$n = 1, 2, 3$$

This method usually works fine for calculating the long term distribution of time spent in respective states. However, this result is derived under the assumption that $\lambda \neq \mu$. Therefore, we can apply these formulas for suggestion 1 and 2, but not for suggestion 3, since $\lambda = 10 = \mu$.

However, a birth-death-process with $\lambda = \mu$ acts like a random walk, where $(p = 0.5)$. Because of this, it is clear that the time spent in each state is uniformally distributed. In other words, 25% of all time is spent in each respective state.

Below, we construct the function P_theoretical which formalizes the formulas for P_0 and P_n when $\lambda \neq \mu$. After that, we construct table 3, similarly to table 2, to contain the results. The value(s) for suggestion 3 are manually entered. In order to simplify comparison between tables 2 and 3, the portions of time will be converted to percentage points.

```r
#n = 0,1,2,3
P_theoretical <- function(n, lambda, mu, N=3) {
  #Lambda over mu
  lamb_mu <- lambda / mu

  #fraction is formally equal to P_0
  fraction <- (1-lamb_mu)/(1-lamb_mu^(N+1))

  return(lamb_mu^n * fraction)
}
```

```r
#Vector of theoretical distribution for suggestion 1 (converted to percentages)
sgstn1 <- c(P_theoretical(0, 2, 10)*100,
            P_theoretical(1, 2, 10)*100,
            P_theoretical(2, 2, 10)*100,
            P_theoretical(3, 2, 10)*100)

#Vector of theoretical distribution for suggestion 2 (converted to percentages)
sgstn2 <- c(P_theoretical(0, 6, 10)*100,
            P_theoretical(1, 6, 10)*100,
            P_theoretical(2, 6, 10)*100,
            P_theoretical(3, 6, 10)*100)

#Vector of theoretical distribution for suggestion 3 (converted to percentages)
sgstn3 <- c(25,
            25,
            25,
            25)

#Combinating immediately preceding results
proportions <- cbind("State" = 0:3,
                     "Suggestion 1" = sgstn1,
                     "Suggestion 2" = sgstn2,
                     "Suggestion 3" = sgstn3)

#Plotting a table of the results
knitr::kable(proportions,
             digits = 3,
             caption = "The stationary distribution of time (in percentages) spent in each state for ea
```

Table 3: The stationary distribution of time (in percentages) spent in each state for each suggestion.

State	Suggestion 1	Suggestion 2	Suggestion 3
0	80.128	45.956	25
1	16.026	27.574	25
2	3.205	16.544	25
3	0.641	9.926	25

In table 3, we see results that are very similiar to the corresponding results in table 2. This strengthens the reliability of the simulations carried out throughout this report. Regarding the requirements of the institution that the queue be full at most 5% of the time, table 3 indicates that this requirement is only met by suggestion 1. This is the same result observed in table 2.

Just like we observed in table 2, the computer is expected to sit idly for about 80% of the time with suggestion 1. This does fulfil the institutions requirement, but is a waste of a computational resource. One way to improve this, without changing the suggestions, would be to extend the computers queueing-capabilities. If the number of states in the system would be extended, the distributions in table 2 and 3 would be "drawn" out. For a large enough number of states $N+1$, suggestion 2 and 3 would also eventually fulfil the institutions requirement. As shown below, this altercation does not even require a relatively large N. For suggestion 2, we would need $N = 5$, meaning states $0, 1, 2, 3, 4, 5$ make up the system. For suggestion 3, we would need $N = 19$, because the probability is uniformly distributed over the states.

```
#Suggestion 2
P_theoretical(5,6,10,5)*100
```

```
## [1] 3.262621
```

```
#Suggestion 3
100/(19+1)
```

```
## [1] 5
```