HOMEWORK 3
OR647: QUEUEING THEORY, SPRING 2021

DAVID PRENTISS

1. Problem 1.17

Table 1 gives observations regarding customers at a single-server FCFS queue.

a. Compute the average time in the queue and the average time in the system.

Date: March 3, 2021.

Table 1. Customer Data and Results for Problem 1.17

	Data		Results				
Customer	$T^{(n)}$	$S^{(n)}$	$A^{(n)}$	$U^{(n)}$	$D^{(n)}$	$W_q^{(n)}$	$W^{(n)}$
1	1	3	0	0	3	0	3
2	9	7	1	3	10	2	9
3	6	9	10	10	19	0	9
4	4	9	16	19	28	3	12
5	7	10	20	28	38	8	18
6	9	4	27	38	42	11	15
7	5	8	36	42	50	6	14
8	8	5	41	50	55	9	14
9	4	5	49	55	60	6	11
10	10	3	53	60	63	7	10
11	6	6	63	63	69	0	6
12	12	3	69	69	72	0	3
13	6	5	81	81	86	0	5
14	8	4	87	87	91	0	4
15	9	9	95	95	104	0	9
16	5	9	104	104	113	0	9
17	7	8	109	113	121	4	12
18	8	6	116	121	127	5	11
19	8	8	124	127	135	3	11
20	7	3	132	135	138	3	6
Sum		124				67	191
Mean		6.2				3.35	9.55

Solution. Let $A^{(1)} = U^{(1)} = 0$. By iteratively applying the FCFS relationships to the data, we may generate the results in Table 1. The average time in the queue and average time in the system are the averages of their respective columns over the total number of arrivals. That is

$$W_q = \frac{1}{20} \sum_{i=1}^{20} W_q^{(i)} = 3.35$$

and

$$W = \frac{1}{20} \sum_{i=1}^{20} W^{(i)} = 9.55$$

b. Calculate the average system waiting time of those customers who had to wait for service (i.e., exclude those who were immediately taken into service). Calculate the average length of the queue, the average number in the system, and the fraction of idle time of the server. What is the maximum queue length?

Solution. From $T^{(20)}$ and $A^{(20)}$ we know that customer 21 arrives after customer 20 has left the system. Thus the system starts and ends in an empty state over the time horizon $[A^{(0)}, D^{(20)}]$ and we may apply Little's Law to find L_q and L. For the sample arrival rate, $\lambda = 20/D^{(20)} = 20/138$ we have

$$L_q = \lambda W_q = \frac{20}{138} \cdot \frac{67}{20} \approx 0.486 \text{ customers}$$

and

$$L = \lambda W = \frac{20}{138} \cdot \frac{191}{20} \approx 1.384 \text{ customers}$$

The in-service time fraction is the sum of service times divided by the length of the time horizon. So the idle time fraction is one minus the in-service time fraction or

$$1 - \frac{1}{D^{(20)}} \sum_{i=0}^{20} S^{(i)} = 1 - \frac{124}{138} \approx 0.101$$

Finally, to find the maximum length of the queue, first note that the number of customers in the queue can only change as a result of arrival or service-start events. Then calculate the number of customers in the queue $N_q(t)$ at all times $\{t\}$ such that t is an arrival or service-start time. That is, for

$$\begin{aligned}\{t\} &= \{0, 1, 3, 10, 16, 19, 20, 27, 28, 36, 38, 41, 42, 49, 50, 53, 55, 60,\\ &\qquad 63, 69, 81, 87, 95, 104, 109, 113, 116, 121, 124, 127, 132, 135\},\\ \{N_q\} &= \{0, 1, 0, 0, 1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0,\\ &\qquad 0, 1, 0, 1, 0, 1, 0, 1, 0\}.\end{aligned}$$

From which, we can see the maximum is two (2).

2. Problem 1.18

Items arrive at an initially unoccupied inspection station at a uniform rate of one every 5 min. With the time of the first arrival set equal to 5, the chronological times for inspection completion of the first 10 items were observed to be 7, 17, 23, 29, 35, 38, 39, 44, 46, and 60, respectively. By manual simulation of the operation for 60 min, using these data, develop sample results for the mean number in system and the percentage idle time experienced.

TABLE 2. Customer Data and Results for Problem 1.19

Data		Results	
$A^{(n)}$	$S^{(n)}$	$U^{(n)}$	$D^{(n)}$
1	2.22	1.00	3.22
2	1.76	3.22	4.98
3	2.13	4.98	7.11
4	0.14	7.11	7.25
5	0.76	7.25	8.01
6	0.70	8.01	8.71
7	0.47	8.71	9.18
8	0.22	9.18	9.40
9	0.18	9.40	9.58
10	2.41	10.00	12.41
11	0.41	12.41	12.82
12	0.46	12.82	13.28
13	1.37	13.28	14.65
14	0.27	14.65	14.92
15	0.27	15.00	15.27

Solution. From the sequences of arrival and departure times calculate the sequence of customers in the system $\{N\}$ as before

$$\{t\} = \{0, 5, 7, 10, 15, 17, 20, 23, 25, 29, 30, 35, 38, 39, 40, 44, 45, 46, 50, 55, 60\}$$
$$\{N\} = \{0, 1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 2, 1, 0, 1, 0, 1, 0, 1, 2, 2\}.$$

Then, calculate the time interval between arrival and departure events as

$$\Delta t = \{t_{n+1} - t_n\} = \{5, 2, 3, 5, 2, 3, 3, 2, 4, 1, 5, 3, 1, 1, 4, 1, 1, 4, 5, 5\}$$

To find the mean number in the system, L, we integrate $N(t)$ and divide by the length of the time horizon. That is

$$L = \frac{1}{60}\int_{\{t\}} N(t) = \frac{1}{60}\sum_{i=0}^{\|\{\Delta t\}\|} N(t_i)\Delta t_i = \frac{68}{60} \approx 1.133$$

Finally, to find the percentage idle time, we tally the length of intervals where $N(t) = 0$. That is

$$\frac{1}{60}(5 + 3 + 1 + 1 + 4) = \frac{14}{60} \approx 0.233 = \%23.3$$

3. PROBLEM 1.19

Table 2 lists the arrival times and service durations for customers in a FCFS single-server queue. From this data, compute L_q (the time-average number in queue) and $L_q^{(A)}$ (the average number in queue as seen by arriving customers). For L_q, use a time horizon of [0, 15.27], where 15.27 is the time that the last customer exits the system. Assume the system is empty at $t = 0$.

TABLE 3. Customer Data and Results for Problem 4 (a)

Data		Results		
$A^{(n)}$	$S^{(n)}$	$U^{(n)}$	$D^{(n)}$	$W_q^{(n)}$
0	6	0	6	0
6	4	6	10	0
9	6	10	16	1
10	1	16	17	6
15	2	17	19	2
17	1	19	20	2
19	3	20	23	1
23	5	23	28	0
29	8	29	37	0
35	6	37	43	2

Solution. From the sequences of arrival and departure times calculate the sequence of customers in the queue $\{N_q\}$ as before

$$\begin{aligned}\{t\} = \{&1, 2, 3, 3.22, 4, 4.98, 5, 6, 7, 7.11, 7.25, 8, 8.01, 8.71, 9,\\ &9.18, 9.4, 10, 11, 12, 12.41, 12.82, 13, 13.28, 14, 14.65, 15\}\\ \{N_q\} = \{&0, 1, 2, 1, 2, 1, 2, 3, 4, 3, 2, 3, 2, 1, 2, 1, 0, 0, 1, 2, 1, 0, 1, 0, 1, 0, 0\}\end{aligned}$$

Then, calculate the time interval between arrival and service-start events as

$$\begin{aligned}\Delta t = \{t_{n+1} - t_n\} = \{&1, 1, 0.22, 0.78, 0.98, 0.02, 1, 1, 0.11, 0.14, 0.75, 0.01, 0.7,\\ &0.29, 0.18, 0.22, 0.6, 1, 1, 0.41, 0.41, 0.18, 0.28, 0.72, 0.65, 0.35\}\end{aligned}$$

To find the mean number in the queue, L_q, we integrate $N_q(t)$ and divide by the length of the time horizon. That is

$$L_q = \frac{1}{15.27}\int_{\{t\}} N_q(t) = \frac{1}{15.57}\sum_{i=0}^{\|\{\Delta t\}\|} N_q(t_i)\Delta t_i = \frac{17.02}{15.57} \approx 1.115$$

To find $L_q^{(A)}$, we must first identify all customers that experienced queue delays ($W_q(n) > 0$). The queue length as seen by such a customer arriving at time t is $N_q(t) - 1$. The average of these values is

$$L_q^{(A)} = \frac{1}{8}(1 + 1 + 1 + 2 + 3 + 2 + 1 + 1) = \frac{12}{8} = 1.5$$

4

Table 3 gives the arrival and service times of a sequence of customers to an airline ticket counter. The counter has one line and customers are served in a first-come-first-served manner. Note: Use the same time horizon for parts (a) and (b) in computing the average number in queue.

a. Assuming one server, compute the average wait in queue and the average number in queue.

TABLE 4. Customer Data and Results for Problem 4 (b)

Data		Results		
$A^{(n)}$	$S^{(n)}$	$U^{(n)}$	$D^{(n)}$	$W_q^{(n)}$
0	12	0	12	0
6	8	6	14	0
9	12	12	24	3
10	2	14	16	4
15	4	16	20	1
17	2	20	22	3
19	6	22	28	3
23	10	24	34	1
29	16	29	45	0
35	12	35	47	0

Solution. See Table 3. For the sample arrival rate, $\lambda = 10/D^{(10)} = 10/43$ we have

$$W_q = \frac{1}{10}\sum_{i=1}^{10} W_q^{(i)} = \frac{14}{10} = 1.4$$

and

$$L_q = \lambda W_q = \frac{10}{43}\cdot\frac{14}{10} \approx 0.326$$

b. Assuming there are two servers who are working at half speed (i.e., double the service times), compute the average wait in queue and the average number in queue. Table 4

Solution. See Table 4. For the sample arrival rate, $\lambda = 10/D^{(10)} = 10/43$ we have

$$W_q = \frac{1}{10}\sum_{i=1}^{10} W_q^{(i)} = \frac{15}{10} = 1.5$$

and

$$L_q = \lambda W_q = \frac{10}{43}\cdot\frac{15}{10} \approx 0.349$$

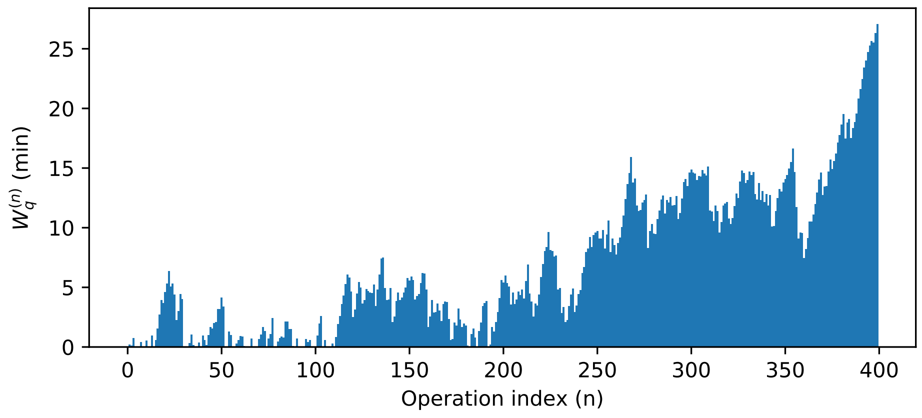
c. Compare the answers in (a) and (b) and discuss why there is a difference.

Solution. I could not discern a significant difference between (a) and (b). I suspect this is because in both cases, $\rho = 42/43 \approx 1$. At lower demand, we might expect to see larger differences in performance.

5

Reconstruct the Excel spreadsheet given in class for airplane arrivals and departures to/from a single runway. Specifically, create columns containing the following elements: Operation #, inter-arrival time (min), actual arrival time (min), arrival or departure (A or D), runway hold time, queue waiting time, departure time, arrival count (1 for arrivals 0 for departures), departure count (0 for arrivals 1 for departures). The spreadsheet should also contain the following global parameters: % arrivals (0.5), arrival runway hold time (1 min), departure runway hold time (1.5 min), operation arrival rate (40 / hr), average service time, rho.

FIGURE 1. Sample queue-waiting times for a typical scenario. Proportion of arrival operations = 0.5, number of operations = 400.



a. Graph the queue-waiting time for a given sequence of 400 simulated operations.

Solution. See Figure 1.

b. Vary λ from 5, 10, . . . , 45, 50. Calculate the average queue-waiting time W_q for each of these cases (based on simulation of 400 operations).

Solution. See part (c).

c. Calculate the average queue-waiting time for each of these cases for an $M/M/1$ queue with the same λ and μ. Plot both values of W_q as a function of λ and compare the results. How good is the $M/M/1$ queue as an approximation you're your simulation?

Solution. See Figures 2 and 3. Applying Little's Law to the $M/M/1$ consistently overestimates the queue-waiting times. Relative error of the estimate grows with ρ and is impractically large as $\rho \to 1$. Error improves with larger numbers of operations. The source of the error is largely the fact that service times are not exponentially distributed, which contradicts the assumptions of the $M/M/1$ model. Additional simulations with exponentially distributed service times with same mean confirmed this with relative errors less than %5 for $\rho \leq 0.8$

d. Repeat part (b) when 75% of operations are departures.

Solution. See Figures 4 and 5. No significant difference was noted for scenarios with a higher proportion of departure events. This may be because arrival rates have an even lower variance than the case with equal numbers of arrival and departure events. non-exponentially distributed arrival times was confirmed as the most significant source of error for this case as well.

FIGURE 2. Sample average queue-waiting times by arrival rate. Boxplot shows summary statistics for simulations of 10000 seeds. Blue line indicates theoretical values from $M/M/1$ model. Proportion of arrival operations = 0.5, number of operations = 400.

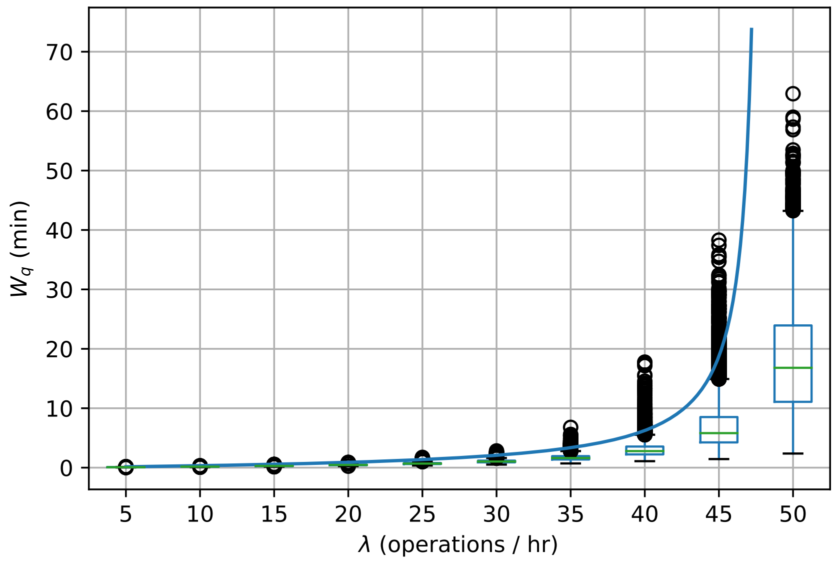


FIGURE 3. Relative error by arrival rate. Proportion of arrival operations = 0.5, number of operations = 400.

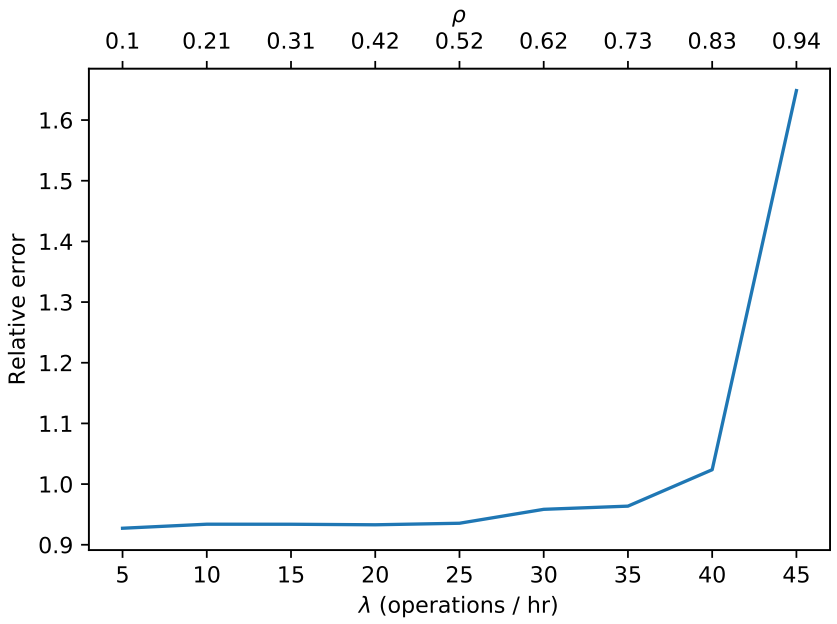
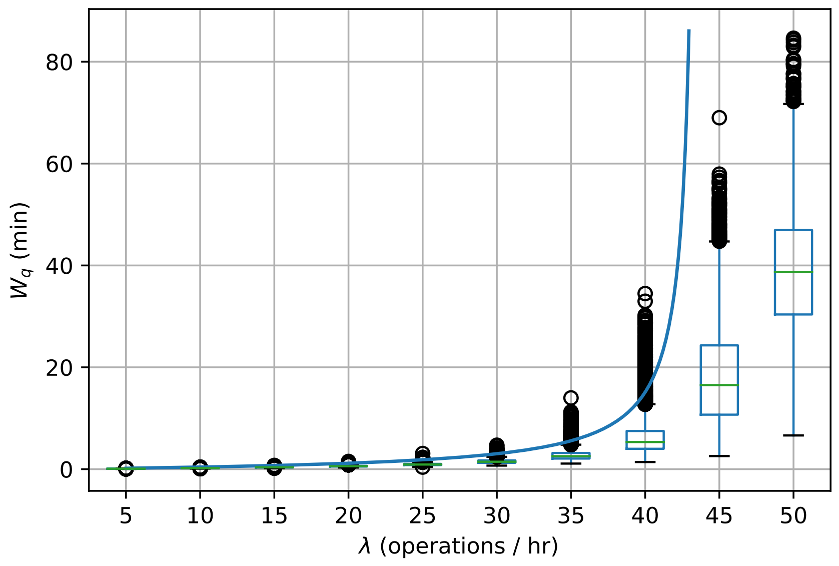


FIGURE 4. Sample average queue-waiting times by arrival rate. Boxplot shows summary statistics for simulations of 10000 seeds. Blue line indicates theoretical values from $M/M/1$ model. Proportion of arrival operations = 0.25, number of operations = 400.



6

A hardware store has a "merchandise pickup" window where customers arrive to load previously purchased items into their car. Arrivals to the pickup window follow a Poisson process with rate 45 per hour. The time for a store employee to load merchandise into a customer's car is exponentially distributed with a mean of 3 minutes. There are 3 employees. What is the average time it takes for a customer to get his/her merchandise loaded into the car (wait in queue plus service time)?

Solution. Let $\lambda = \frac{45}{60} = 0.75$, $\mu = \frac{1}{3}$, and $c = 3$. Then $r = \frac{\lambda}{\mu} = \frac{9}{4} = 2.25$ and $\rho = \frac{r}{c} = \frac{2.25}{3} = 0.75$. For an $M/M/c$

$$p_0 = \left(\frac{r^c}{c!(1-\rho)} + \sum_{n=0}^{c-1} \frac{r^n}{n!} \right)^{-1} \approx 0.075$$

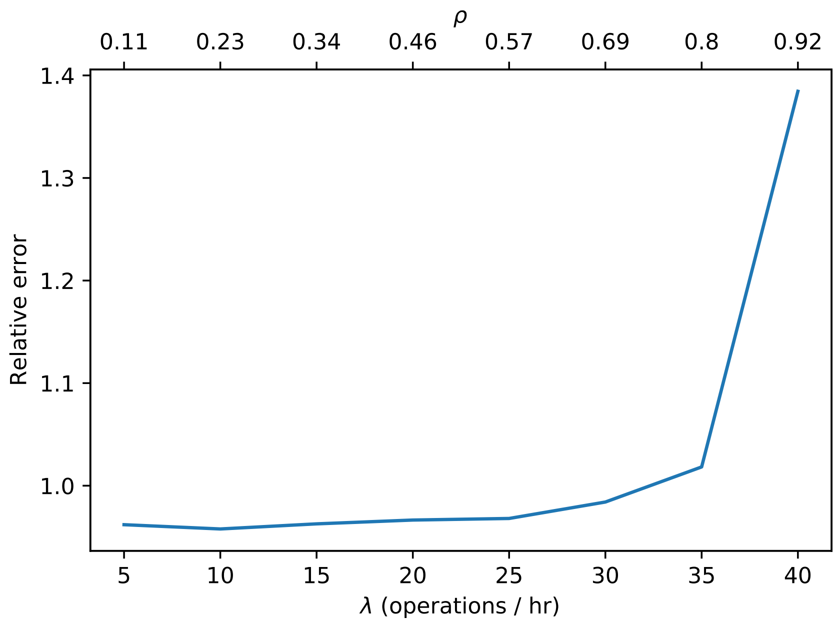
So

$$L_q = \left(\frac{r^c \rho}{c!(1-\rho)^2} \right) p_0 = \left(\frac{(2.25)^5 (0.75)}{3!(1-(0.75))^2} \right) 0.075 \approx 1.703 \text{ customers}$$

and then

$$W = \frac{1}{\mu} + \frac{L_q}{\lambda} = \frac{1}{1/3} + \frac{1.703}{0.75} \approx 5.271 \text{ minutes}$$

FIGURE 5. Relative error by arrival rate. Proportion of arrival operations = 0.25, number of operations = 400.



7

A local polling location has 5 voting machines. People arrive to cast their votes according to a Poisson process with rate $\lambda = 75$ per hour. The time to cast a ballot is exponential with a mean of 2 minutes.

a. What is the fraction of time that a given voting machine is being utilized?

Solution. Let $\lambda = \frac{75}{60} = 1.25$, $\mu = \frac{1}{2} = 0.5$, and $c = 5$. Then $r = \frac{\lambda}{\mu} = \frac{1.25}{0.5} = 2.5$ The long-term average fraction of occupied server time for an $M/M/c$ is $\rho = \frac{r}{c} = \frac{2.5}{5} = 0.5$.

b. What is the fraction of time that 3 or fewer machines are in use?

Solution. The fraction of time that 3 or fewer machines are in use is the sum of the probabilities p_n that the system is in state n where $n \leq 3$ or

$$\Pr\{n \leq 3\} = \sum_{n=0}^{3} p_n = \sum_{n=0}^{3} \frac{\lambda^n}{n!\mu^n} p_0$$

For the $M/M/c$ model

$$p_0 = \left(\frac{r^c}{c!(1-\rho)} + \sum_{n=0}^{c-1} \frac{r^n}{n!} \right)^{-1} \approx 0.080$$

and

$$\Pr\{n \leq 3\} = 0.080 \cdot \sum_{n=0}^{3} \frac{(1.25)^n}{n!(0.5)^n} \approx 0.739$$

c. What is the average line length?

Solution.

$$L_q = \left(\frac{r^c\rho}{c!(1-\rho)^2}\right)p_0 = \left(\frac{(2.5)^5(0.5)}{5!(1-(0.5))^2}\right)0.080 \approx 0.130 \text{ customers}$$

8

Arrivals to a queueing system follow a Poisson process with rate 18 per hour. Service times are exponential with rate 20 per hour. Each server is paid \$40 per hour while working and \$10 per hour while idle. The company determines that it loses \$2 per customer per hour spent waiting in the queue, due to "ill-will" or lost customer loyalty.

a. What is the hourly cost to the company if 1 server is employed?

Solution. Let $\lambda = 18$, $\mu = 20$, and $c = 1$. Then $r = \frac{\lambda}{\mu} = \frac{18}{20} = 0.9$ and $\rho = \frac{r}{c} = \frac{0.9}{1} = 0.9$. If there is a single server, we may adopt the $M/M/1$ model and $L_q = \frac{\rho^2}{(1-\rho)} = \frac{0.9^2}{1-0.9} = 8.1$. The cost of the system then is

$$\$40 \cdot r + \$10(c-r) + \$2 \cdot L_q = \$40 \cdot 0.9 + \$10(1-0.9) + \$2 \cdot 8.1 = \$53.20 \text{ per hour}$$

b. What is the hourly cost to the company if 2 servers are employed?

Solution. Let $\lambda = 18$ and $\mu = 20$ as before, and $c = 2$. Then $r = \frac{\lambda}{\mu} = \frac{18}{20} = 0.9$ as before and $\rho = \frac{r}{c} = \frac{0.9}{2} = 0.45$. Since there are two servers, we chose the $M/M/c$ model where

$$p_0 = \left(\frac{r^c}{c!(1-\rho)} + \sum_{n=0}^{c-1}\frac{r^n}{n!}\right)^{-1} \approx 0.379$$

and then

$$L_q = \left(\frac{r^c\rho}{c!(1-\rho)^2}\right)p_0 \approx 0.229$$

The cost then is

$$\$40 \cdot r + \$10(c-r) + \$2 \cdot L_q = \$40 \cdot 0.9 + \$10(2-0.9) + \$2 \cdot 0.012 \approx \$47.46 \text{ per hour}$$

c. If the company adds a 3rd server, would the cost go up or down compared with 2 servers? (Give a one- or two-sentence explanation without working the numbers out fully).

Solution. Adding a third server would likely increase the cost as compared with two servers since, in the later case, the cost of the queue is only \$0.46 per hour, whereas the cost of an additional server is at least \$10 per hour. We know that such a break-even point for additional servers exists since the cost of "ill will" is bound below at zero, but the cost of idle employees is positive and unbound above.

9. Problem 3.32

You are managing a call center where the arrival rate is 500 calls per hour and the service time is exponential with a mean of 2 minutes.

a. What is the approximate number of servers needed so that the probability that a customer has a non-zero wait in queue is 10%?

Solution. Let $\lambda = \frac{500}{60} = \frac{25}{3}$, $\mu = \frac{1}{2} = 0.5$, and $\alpha = 0.10$. Then $r = \frac{\lambda}{\mu} = \frac{50}{3}$, $\beta \approx 1.420$ and

$$c \approx r + \beta\sqrt{r} = 22.465 \approx 23$$

b. If the arrival rate increases by 60%, what is the approximate number of servers needed to maintain the same level of service?

Solution. Let $\lambda = 1.6 \cdot \frac{500}{60} = \frac{40}{3}$, $\mu = \frac{1}{2} = 0.5$, and $\beta \approx 1.420$. Then $r = \frac{\lambda}{\mu} = \frac{80}{3}$ and

$$c \approx r + \beta\sqrt{r} = 34.000 \approx 34$$

c. If the average time to complete a call increases by 1 minute (assuming the original arrival rate of 500 calls per hour), what is the approximate number of servers needed to maintain the same level of service?

Solution. Let $\lambda = \frac{500}{60} = \frac{25}{3}$, $\mu = \frac{1}{3}$ and $\beta \approx 1.420$. Then $r = \frac{\lambda}{\mu} = 25$ and

$$c \approx r + \beta\sqrt{r} = 32.101 \approx 33$$

d. For parts (a) and (b), compute the exact minimum number of servers needed to achieve no more than 10% probability of non-zero wait in queue. Compute the exact expected wait in queue for the two scenarios. Are they the same?

Solution. For part (a) Let $r = \frac{\lambda}{\mu} = \frac{50}{3}$. Then $\rho = r/c = \frac{50}{3c}$. The probability of nonzero wait time is given by

$$C(c,r) = \frac{r^c}{c!(1-\rho)}\left(\frac{r^c}{c!(1-\rho)} + \sum_{n=0}^{c-1}\frac{r^n}{n!}\right)^{-1}$$

Starting with our estimate, $c \approx 23$, we have

$$C(23, \frac{50}{3}) = 0.101 > 0.10$$
$$C(24, \frac{50}{3}) = 0.064 < 0.10 \checkmark$$

For part (b) Let $r = \frac{\lambda}{\mu} = \frac{80}{3}$ with $C(c,r)$ as before. Starting with our estimate, $c \approx 34$, we have

$$C(34, \frac{80}{3}) = 0.122 > 0.10$$
$$C(35, \frac{80}{3}) = 0.085 < 0.10 \checkmark$$

10. PROBLEM 3.37

You are managing a call center where arrivals follow a Poisson process with rate of 300 calls per hour and the service time is exponential with a mean of 2 minutes.

a. What is the approximate number of servers needed so that the probability that a customer has a nonzero wait in queue is 5%?

Solution. Let $\lambda = \frac{300}{60} = 5$, $\mu = \frac{1}{2} = 0.5$, and $\alpha = 0.05$. Then $r = \frac{\lambda}{\mu} = 10$, $\beta \approx 1.740$ and

$$c \approx r + \beta\sqrt{r} = 15.502 \approx 16$$

b. If the arrival rate doubles, what is the approximate number of servers needed to maintain the same level of service?

Solution. Let $\lambda = 2 \cdot \frac{300}{60} = 10$, $\mu = \frac{1}{2} = 0.5$, and $\beta \approx 1.740$. Then $r = \frac{\lambda}{\mu} = 20$ and

$$c \approx r + \beta\sqrt{r} = 27.781 \approx 28$$

c. Suppose that you are given the steady-state probabilities $p_0, p_1, \ldots, p_n$. Give a formula or procedure for computing the expected number of customers in queue and the variance of the number of customers in queue using these values.

Solution. Given a sequence of $n + 1$ steady-state probabilities with n large enough that $p_n \approx 0$, the average number in the queue, L_q, is approximated by the sums of the length of the queue in each state weighted by the probability of its respective probability or

$$L_q = \sum_{i=c+1}^{n} (i - c)p_i$$

where c is the number of servers. Here, $i - c$ serves as the size of the queue in each state n and, we may ignore $i \leq c$ since the length of the queue in those states is 0. The variance then is

$$\sigma^2_{L_q} = \sum_{i=c+1}^{n} (i - c - L_q)^2 p_i$$

Care must be taken with an estimate such as this since the distribution of queue lengths may be heavy-tailed. As such, n may need to be very large to avoid a significant estimation error.

Email address: dprentis@gmu.edu

hw3

March 3, 2021

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     from scipy.stats import norm
     from scipy.optimize import newton
     from math import factorial
     from math import sqrt
```

```
[2]: %pprint
```

```
Pretty printing has been turned OFF
```

1 Problem 1 (1.17)

1.1 1 (a)

```
[3]: T = [1,9,6,4,7,9,5,8,4,10,6,12,6,8,9,5,7,8,8,7]
     print(T)
```

```
[1, 9, 6, 4, 7, 9, 5, 8, 4, 10, 6, 12, 6, 8, 9, 5, 7, 8, 8, 7]
```

```
[4]: S = [3,7,9,9,10,4,8,5,5,3,6,3,5,4,9,9,8,6,8,3]
     print(S)
```

```
[3, 7, 9, 9, 10, 4, 8, 5, 5, 3, 6, 3, 5, 4, 9, 9, 8, 6, 8, 3]
```

```
[5]: A = [0] + list(np.cumsum(T)[:-1])
     print(A)
```

```
[0, 1, 10, 16, 20, 27, 36, 41, 49, 53, 63, 69, 81, 87, 95, 104, 109, 116, 124,
132]
```

```
[6]: U = [0 for i in range(len(A))]
     print(U)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
[7]: D = U.copy()
     print(D)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
[8]: for i in range(len(A)):
         D[i] = U[i] + S[i]
         if i < len(A)-1:
             U[i+1] = max(D[i], A[i+1])
     print(U)
```

```
[0, 3, 10, 19, 28, 38, 42, 50, 55, 60, 63, 69, 81, 87, 95, 104, 113, 121, 127,
135]
```

```
[9]: print(D)
```

```
[3, 10, 19, 28, 38, 42, 50, 55, 60, 63, 69, 72, 86, 91, 104, 113, 121, 127, 135,
138]
```

```
[10]: W_q = [u-a for u,a in zip(U,A)]
      print(W_q)
```

```
[0, 2, 0, 3, 8, 11, 6, 9, 6, 7, 0, 0, 0, 0, 0, 0, 4, 5, 3, 3]
```

```
[11]: W = [w+s for w,s in zip(W_q,S)]
      print(W)
```

```
[3, 9, 9, 12, 18, 15, 14, 14, 11, 10, 6, 3, 5, 4, 9, 9, 12, 11, 11, 6]
```

```
[12]: W_ave = sum(W)/len(W)
      print(W_ave)
```

```
9.55
```

```
[13]: W_q_ave = sum(W_q)/len(W_q)
      print(W_q_ave)
```

```
3.35
```

```
[14]: S_ave = sum(S)/len(S)
      print(S_ave)
```

```
6.2
```

1.2 1 (b)

[15]:
```
rate = len(D)/D[-1]
print(len(D), D[-1], rate)
```

```
20 138 0.14492753623188406
```

[16]:
```
L_q = rate*W_q_ave
print(L_q)
```

```
0.48550724637681164
```

[17]:
```
L = rate*W_ave
print(L)
```

```
1.384057971014493
```

[18]:
```
W_q_ave_wait = sum(W_q)/sum([w>0 for w in W_q])
print(W_q_ave_wait)
```

```
5.583333333333333
```

[19]:
```
print(sum(S), D[-1], sum(S)/D[-1], 1-sum(S)/D[-1])
```

```
124 138 0.8985507246376812 0.10144927536231885
```

[20]:
```
print(A)
print(U)
```

```
[0, 1, 10, 16, 20, 27, 36, 41, 49, 53, 63, 69, 81, 87, 95, 104, 109, 116, 124,
132]
[0, 3, 10, 19, 28, 38, 42, 50, 55, 60, 63, 69, 81, 87, 95, 104, 113, 121, 127,
135]
```

[21]:
```
Ti = list(set(A+U))
Ti.sort()
print(Ti)
```

```
[0, 1, 3, 10, 16, 19, 20, 27, 28, 36, 38, 41, 42, 49, 50, 53, 55, 60, 63, 69,
81, 87, 95, 104, 109, 113, 116, 121, 124, 127, 132, 135]
```

[22]:
```
N_q=[sum([a <= t for a in A])-sum([u <= t for u in U]) for t in Ti]
print(N_q)
```

```
[0, 1, 0, 0, 1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 1, 0, 1, 0]
```

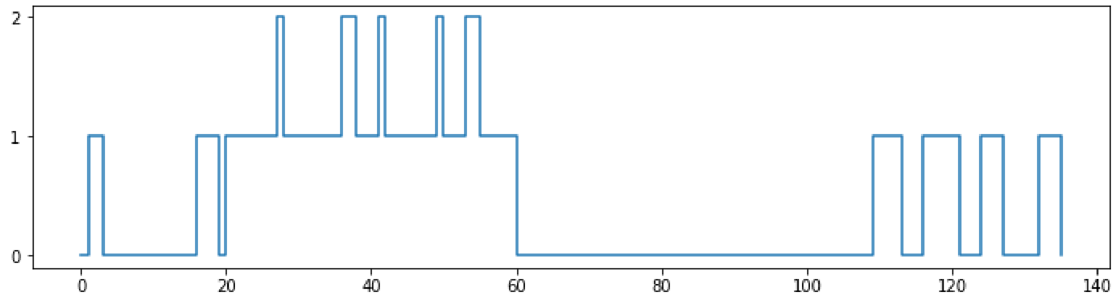
[23]:
```
N=[sum([a <= t for a in A])-sum([d <= t for d in D]) for t in Ti]
print(N)
```

```
[1, 2, 1, 1, 2, 1, 2, 3, 2, 3, 2, 3, 2, 3, 2, 3, 2, 1, 1, 1, 1, 1, 1, 1, 2, 1,
2, 1, 2, 1, 2, 1]
```

[24]:
```
max(N_q)
```

[24]:
```
2
```

[25]:
```
plt.figure(figsize=[12,3])
plt.plot(Ti,N_q,drawstyle='steps-post')
plt.yticks(range(max(N_q)+1))
plt.show()
```



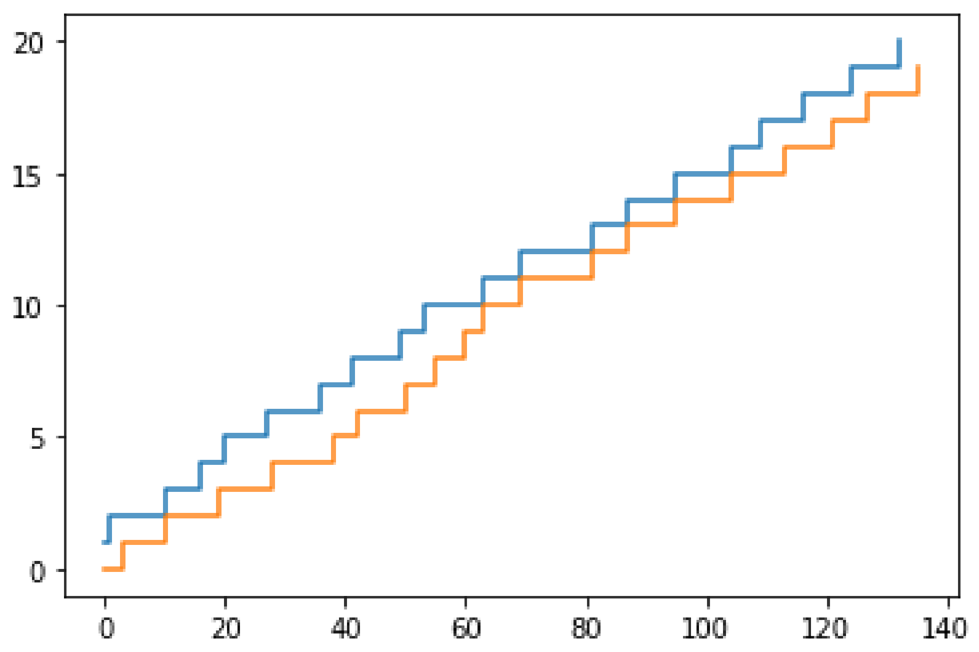
[26]:
```
sum([n*t for n,t in zip(N_q,np.diff(Ti))])/D[-1]
```

[26]:
```
0.4855072463768116
```

[27]:
```
L_q
```

[27]:
```
0.48550724637681164
```

[28]:
```
plt.plot(A,[i for i in range(1,len(A)+1)],drawstyle='steps-post')
plt.plot(U,range(len(U)),drawstyle='steps-post')
#plt.plot(D,range(len(D)),drawstyle='steps-post')
#plt.plot(Ti,N,drawstyle='steps-post')
#plt.plot(Ti,N_q,drawstyle='steps-post')
plt.yticks(list(set([i*(i%5==0) for i in range(len(A)+1)])))
plt.show()
```



```
[29]: print(sum(S), sum(W_q), sum(W))
```

```
124 67 191
```

```
[30]: print(sum(S)/20, sum(W_q)/20, sum(W)/20)
```

```
6.2 3.35 9.55
```

```
[31]: df = pd.DataFrame(dict(Customer=[i for i in␣
       ↪range(1,21)],T=T,S=S,A=A,U=U,D=D,W_q=W_q,W=W))
      print(df.to_latex(index=False))
```

```
\begin{tabular}{rrrrrrrr}
\toprule
 Customer &   T &   S &    A &    U &    D &  W\_q &   W \\
\midrule
        1 &   1 &   3 &    0 &    0 &    3 &    0 &   3 \\
        2 &   9 &   7 &    1 &    3 &   10 &    2 &   9 \\
        3 &   6 &   9 &   10 &   10 &   19 &    0 &   9 \\
        4 &   4 &   9 &   16 &   19 &   28 &    3 &  12 \\
        5 &   7 &  10 &   20 &   28 &   38 &    8 &  18 \\
        6 &   9 &   4 &   27 &   38 &   42 &   11 &  15 \\
        7 &   5 &   8 &   36 &   42 &   50 &    6 &  14 \\
        8 &   8 &   5 &   41 &   50 &   55 &    9 &  14 \\
        9 &   4 &   5 &   49 &   55 &   60 &    6 &  11 \\
       10 &  10 &   3 &   53 &   60 &   63 &    7 &  10 \\
       11 &    6 &    6 &    63 &    63 &    69 &     0 &    6 \\
       12 &   12 &    3 &    69 &    69 &    72 &     0 &    3 \\
       13 &    6 &    5 &    81 &    81 &    86 &     0 &    5 \\
       14 &    8 &    4 &    87 &    87 &    91 &     0 &    4 \\
       15 &    9 &    9 &    95 &    95 &   104 &     0 &    9 \\
       16 &    5 &    9 &   104 &   104 &   113 &     0 &    9 \\
       17 &    7 &    8 &   109 &   113 &   121 &     4 &   12 \\
       18 &    8 &    6 &   116 &   121 &   127 &     5 &   11 \\
       19 &    8 &    8 &   124 &   127 &   135 &     3 &   11 \\
       20 &    7 &    3 &   132 &   135 &   138 &     3 &    6 \\
\bottomrule
\end{tabular}
```

2 Problem 2 (1.18)

[32]:
```
A = [5*i for i in range(1,60//5+1)]
print(A)
```

```
[5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
```

[33]:
```
D = [7, 17, 23, 29, 35, 38, 39, 44, 46, 60]
print(D)
```

```
[7, 17, 23, 29, 35, 38, 39, 44, 46, 60]
```

[34]:
```
Ti = list(set(A+D))
Ti.sort()
Ti = [0] + Ti
print(Ti)
```

```
[0, 5, 7, 10, 15, 17, 20, 23, 25, 29, 30, 35, 38, 39, 40, 44, 45, 46, 50, 55,
60]
```

[35]:
```
N=[sum([a <= t for a in A])-sum([d <= t for d in D]) for t in Ti]
print(N)
```

```
[0, 1, 0, 1, 2, 1, 2, 1, 2, 1, 2, 2, 1, 0, 1, 0, 1, 0, 1, 2, 2]
```

[36]:
```
L = sum([n*t for n,t in zip(N[:-1],np.diff(Ti))])/D[-1]
print(list(np.diff(Ti)))
print(sum([n*t for n,t in zip(N[:-1],np.diff(Ti))]))
print(D[-1])
print(L)
```

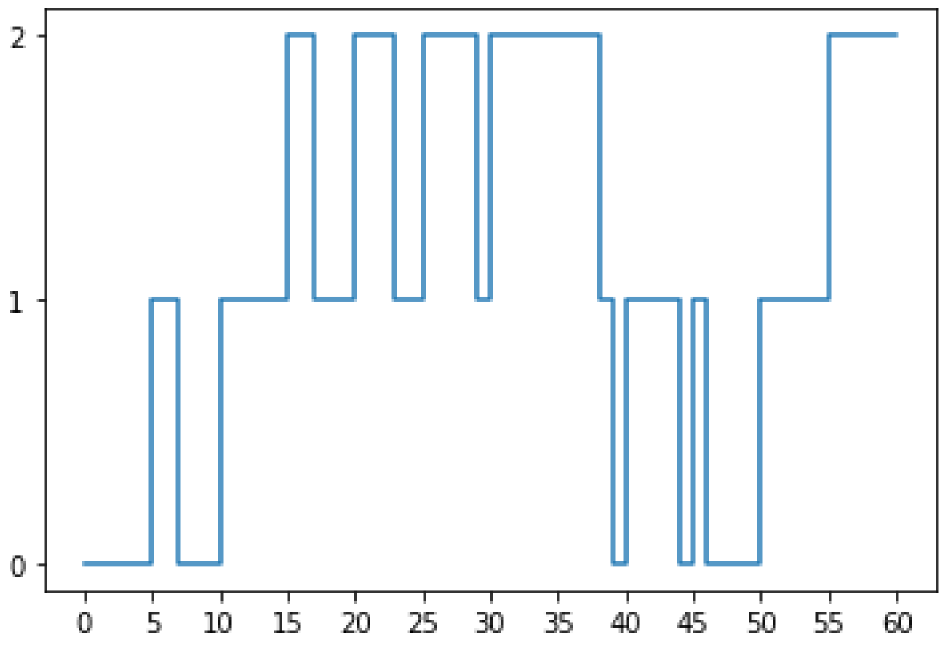
```
[5, 2, 3, 5, 2, 3, 3, 2, 4, 1, 5, 3, 1, 1, 4, 1, 1, 4, 5, 5]
68
```

```
60
1.1333333333333333
```

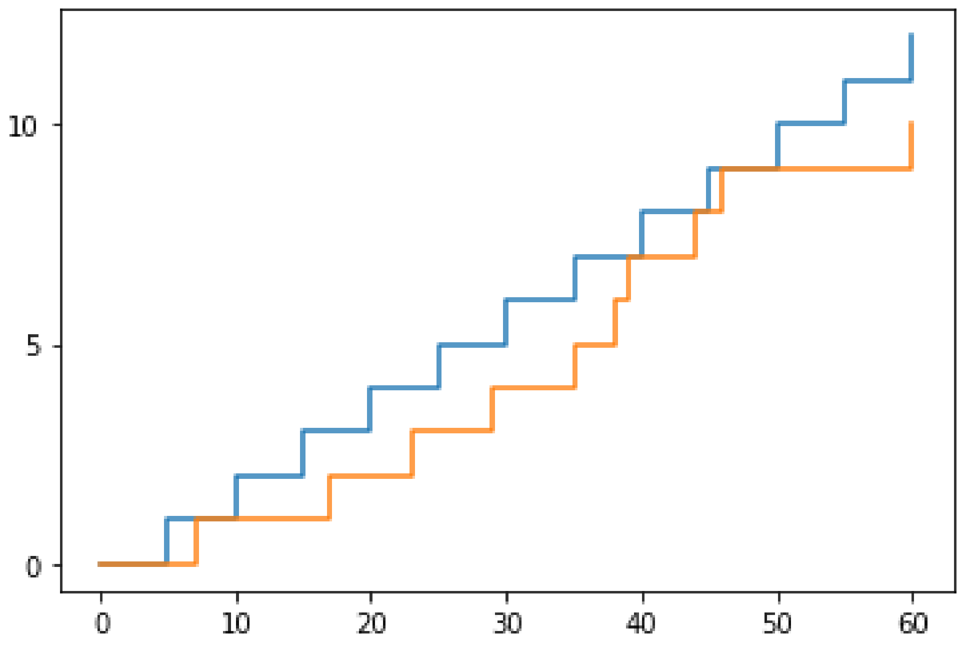
[37]:
```
W_i = sum([n*t for n,t in zip([n==0 for n in N],np.diff(Ti))])/D[-1]
print([n*t for n,t in zip([n==0 for n in N],np.diff(Ti))])
print(list(np.diff(Ti)))
print(sum([n*t for n,t in zip([n==0 for n in N],np.diff(Ti))]))
print(W_i)
```

```
[5, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 4, 0, 0]
[5, 2, 3, 5, 2, 3, 3, 2, 4, 1, 5, 3, 1, 1, 4, 1, 1, 4, 5, 5]
14
0.23333333333333334
```

[38]:
```
plt.plot(Ti,N,drawstyle='steps-post')
plt.yticks(range(max(N)+1))
plt.xticks(list(set([i*(i%5==0) for i in range(max(Ti)+1)])))
plt.show()
```



[39]:
```
plt.plot([0]+A,range(len(A)+1),drawstyle='steps-post')
plt.plot([0]+D,range(len(D)+1),drawstyle='steps-post')
#plt.plot([0]+D,range(len(D)+1),drawstyle='steps-post')
#plt.plot(Ti,N,drawstyle='steps-post')
#plt.plot(Ti,N_q,drawstyle='steps-post')
plt.yticks(list(set([i*(i%5==0) for i in range(len(A)+1)])))
plt.show()
```



3 Problem 3 (1.19)

[40]:
```
A = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
print(A)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

[41]:
```
S = [2.22,1.76,2.13,0.14,0.76,0.70,0.47,0.22,0.18,2.41,0.41,0.46,1.37,0.27,0.27]
print(S)
```

```
[2.22, 1.76, 2.13, 0.14, 0.76, 0.7, 0.47, 0.22, 0.18, 2.41, 0.41, 0.46, 1.37,
0.27, 0.27]
```

[42]:
```
D = [0 for i in range(len(A))]
U = D.copy()
U[0] = A[0]
for i in range(len(A)):
    D[i] = U[i] + S[i]
    if i < len(A)-1:
        U[i+1] = max(D[i], A[i+1])
print(U)
print(D)
```

```
[1, 3.22, 4.98, 7.11, 7.25, 8.01, 8.709999999999999, 9.18, 9.4, 10, 12.41,
12.82, 13.280000000000001, 14.650000000000002, 15]
```

```
[3.22, 4.98, 7.11, 7.25, 8.01, 8.709999999999999, 9.18, 9.4, 9.58, 12.41, 12.82,
13.280000000000001, 14.650000000000002, 14.920000000000002, 15.27]
```

[43]:
```
Ti = list(set(A+U))
Ti.sort()
print(Ti)
```

```
[1, 2, 3, 3.22, 4, 4.98, 5, 6, 7, 7.11, 7.25, 8, 8.01, 8.709999999999999, 9,
9.18, 9.4, 10, 11, 12, 12.41, 12.82, 13, 13.280000000000001, 14,
14.650000000000002, 15]
```

[44]:
```
N_q=[sum([a <= t for a in A])-sum([u <= t for u in U]) for t in Ti]
print(N_q)
```

```
[0, 1, 2, 1, 2, 1, 2, 3, 4, 3, 2, 3, 2, 1, 2, 1, 0, 0, 1, 2, 1, 0, 1, 0, 1, 0,
0]
```

[45]:
```
L_q = sum([n*t for n,t in zip(N_q,np.diff(Ti))])/D[-1]
print([round(t,4) for t in np.diff(Ti)])
print(sum([n*t for n,t in zip(N_q,np.diff(Ti))]))
print(L_q)
```

```
[1.0, 1.0, 0.22, 0.78, 0.98, 0.02, 1.0, 1.0, 0.11, 0.14, 0.75, 0.01, 0.7, 0.29,
0.18, 0.22, 0.6, 1.0, 1.0, 0.41, 0.41, 0.18, 0.28, 0.72, 0.65, 0.35]
17.020000000000007
1.1146037982973154
```

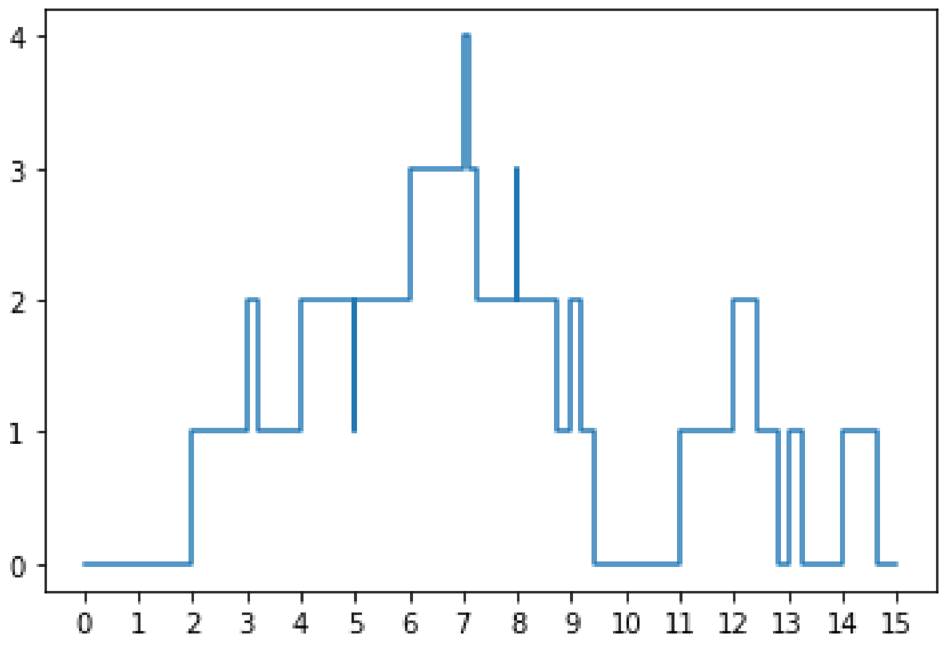
[46]:
```
N_q_A = [N_q[i+1]-1 if N_q[i+1]>N_q[i] else 0 for i in range(len(N_q[:-1]))]
print(N_q_A)
```

```
[0, 1, 0, 1, 0, 1, 2, 3, 0, 0, 2, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
```

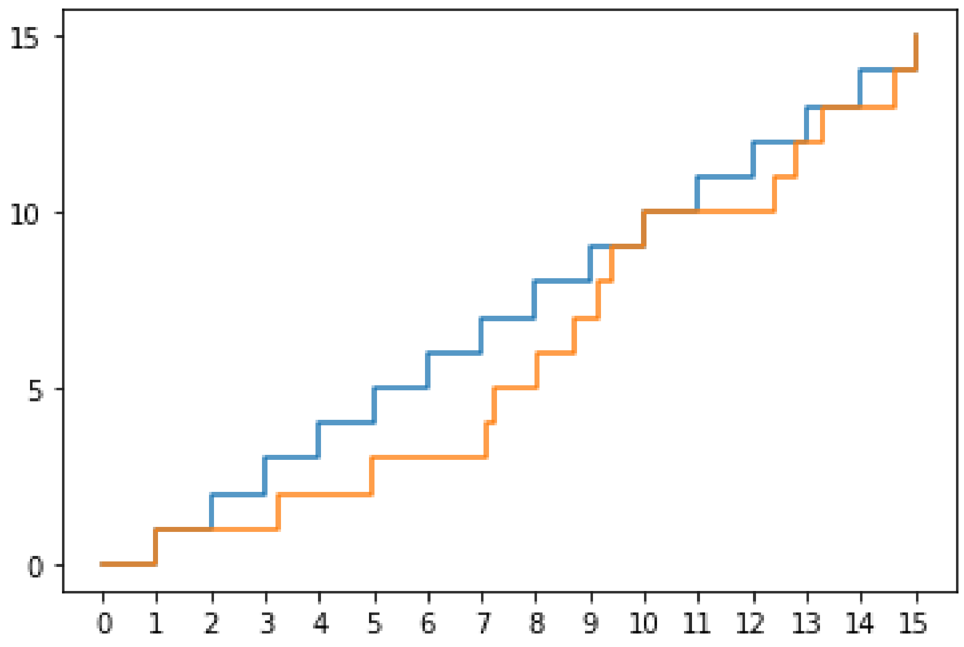
[47]:
```
L_q_A = sum(N_q_A)/sum([n>0 for n in N_q_A])
print(sum(N_q_A))
print(sum([n>0 for n in N_q_A]))
print(L_q_A)
```

```
12
8
1.5
```

[48]:
```
plt.plot([0]+Ti,[0]+N_q,drawstyle='steps-post')
plt.yticks(range(max(N_q)+1))
plt.xticks(range(16))
plt.show()
```



```
[49]: plt.plot([0]+A,range(len(A)+1),drawstyle='steps-post')
      plt.plot([0]+U,range(len(U)+1),drawstyle='steps-post')
      #plt.plot([0]+D,range(len(D)+1),drawstyle='steps-post')
      #plt.plot(Ti,N,drawstyle='steps-post')
      #plt.plot(Ti,N_q,drawstyle='steps-post')
      plt.yticks(list(set([i*(i%5==0) for i in range(len(A)+1)])))
      plt.xticks(range(16))
      plt.show()
```



```
[50]: df = pd.DataFrame(dict(A=A,S=S,U=U,D=D))
      print(df.to_latex(index=False))
```

```
\begin{tabular}{rrrr}
\toprule
  A &     S &      U &      D \\
\midrule
  1 &  2.22 &   1.00 &   3.22 \\
  2 &  1.76 &   3.22 &   4.98 \\
  3 &  2.13 &   4.98 &   7.11 \\
  4 &  0.14 &   7.11 &   7.25 \\
  5 &  0.76 &   7.25 &   8.01 \\
  6 &  0.70 &   8.01 &   8.71 \\
  7 &  0.47 &   8.71 &   9.18 \\
  8 &  0.22 &   9.18 &   9.40 \\
  9 &  0.18 &   9.40 &   9.58 \\
 10 &  2.41 &  10.00 &  12.41 \\
 11 &  0.41 &  12.41 &  12.82 \\
 12 &  0.46 &  12.82 &  13.28 \\
 13 &  1.37 &  13.28 &  14.65 \\
 14 &  0.27 &  14.65 &  14.92 \\
 15 &  0.27 &  15.00 &  15.27 \\
\bottomrule
\end{tabular}
```

4 Problem 4

4.1 4 (a)

[51]:
```
A=[0,6,9,10,15,17,19,23,29,35]
S=[6,4,6,1,2,1,3,5,8,6]
```

[52]:
```
D = [0 for i in range(len(A))]
U = D.copy()
U[0] = A[0]
for i in range(len(A)):
    D[i] = U[i] + S[i]
    if i < len(A)-1:
        U[i+1] = max(D[i], A[i+1])
print(U)
print(D)
```

```
[0, 6, 10, 16, 17, 19, 20, 23, 29, 37]
[6, 10, 16, 17, 19, 20, 23, 28, 37, 43]
```

[53]:
```
D_4a = D[-1]
print(D_4a)
```

```
43
```

[54]:
```
Ti = list(set(A+U))
Ti.sort()
print(Ti)
```

```
[0, 6, 9, 10, 15, 16, 17, 19, 20, 23, 29, 35, 37]
```

[55]:
```
N_q=[sum([a <= t for a in A])-sum([u <= t for u in U]) for t in Ti]
print(N_q)
```

```
[0, 0, 1, 1, 2, 1, 1, 1, 0, 0, 0, 1, 0]
```

[56]:
```
W_q = [u-a for u,a in zip(U,A)]
W_q_ave = sum(W_q)/len(W_q)
print(W_q)
print(W_q_ave)
```

```
[0, 0, 1, 6, 2, 2, 1, 0, 0, 2]
1.4
```

[57]:
```
L_q = sum([n*t for n,t in zip(N_q,np.diff(Ti))])/D_4a
print(L_q)
```

```
0.32558139534883723
```

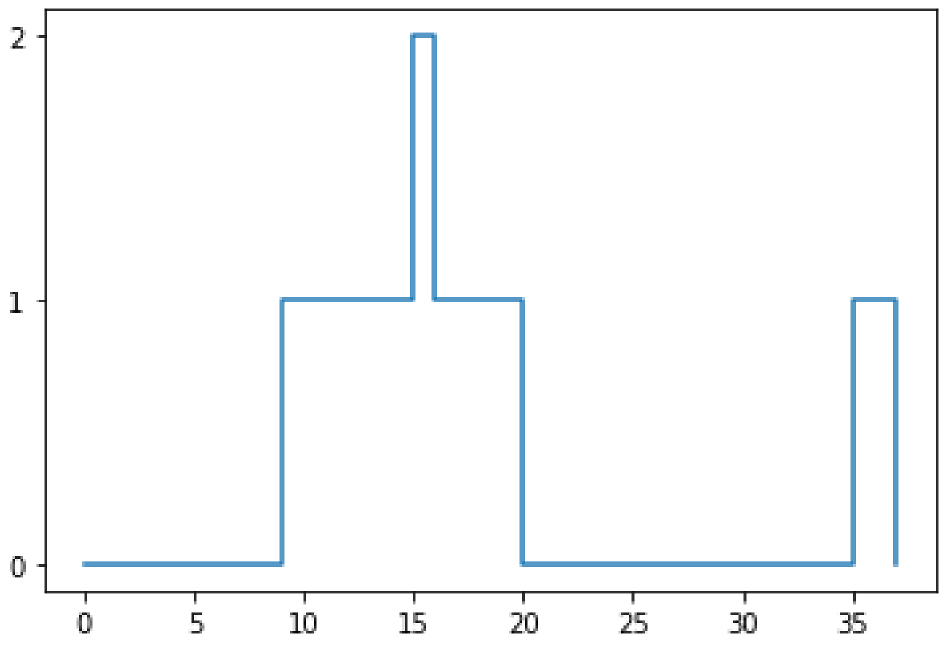
```
[58]: df = pd.DataFrame(dict(A=A,S=S,U=U,D=D,W_q=W_q))
      print(df.to_latex(index=False))
```

```
\begin{tabular}{rrrrr}
\toprule
  A &  S &   U &   D &  W\_q \\
\midrule
  0 &  6 &   0 &   6 &     0 \\
  6 &  4 &   6 &  10 &     0 \\
  9 &  6 &  10 &  16 &     1 \\
 10 &  1 &  16 &  17 &     6 \\
 15 &  2 &  17 &  19 &     2 \\
 17 &  1 &  19 &  20 &     2 \\
 19 &  3 &  20 &  23 &     1 \\
 23 &  5 &  23 &  28 &     0 \\
 29 &  8 &  29 &  37 &     0 \\
 35 &  6 &  37 &  43 &     2 \\
\bottomrule
\end{tabular}
```

```
[59]: W_q_ave = sum(W_q)/len(W_q)
      print(W_q_ave)
```

```
1.4
```

```
[60]: plt.plot([0]+Ti,[0]+N_q,drawstyle='steps-post')
      plt.yticks(range(max(N_q)+1))
      plt.xticks(list(set([i*(i%5==0) for i in range(max(Ti)+1)])))
      plt.show()
```



4.2 4 (b)

```
[61]: A=[0,6,9,10,15,17,19,23,29,35]
      S=[6,4,6,1,2,1,3,5,8,6] # From 4 (a)
      S=[2*s for s in S]
```

```
[62]: D = [0 for i in range(len(A))]
      U = D.copy()
      U[0] = A[0]
      U[1] = A[1]
      D[0] = U[0] + S[0]
      for i in range(1, len(A)):
          D[i] = U[i] + S[i]
          if i < len(A)-1:
              U[i+1] = max(min(max(D), max(D[:D.index(max(D))]+D[D.index(max(D))+1:
       ↪])), A[i+1])
      print(U)
      print(D)
```

```
[0, 6, 12, 14, 16, 20, 22, 24, 29, 35]
[12, 14, 24, 16, 20, 22, 28, 34, 45, 47]
```

```
[63]: Ti = list(set(A+U))
      Ti.sort()
      print(Ti)
```

```
[0, 6, 9, 10, 12, 14, 15, 16, 17, 19, 20, 22, 23, 24, 29, 35]
```

[64]:
```
N_q=[sum([a <= t for a in A])-sum([u <= t for u in U]) for t in Ti]
print(N_q)
```

```
[0, 0, 1, 2, 1, 0, 1, 0, 1, 2, 1, 0, 1, 0, 0, 0]
```

[65]:
```
L_q = sum([n*t for n,t in zip(N_q,np.diff(Ti))])/D_4a
print(L_q)
```

```
0.3488372093023256
```

[66]:
```
W_q = [u-a for u,a in zip(U,A)]
print(W_q)
```

```
[0, 0, 3, 4, 1, 3, 3, 1, 0, 0]
```

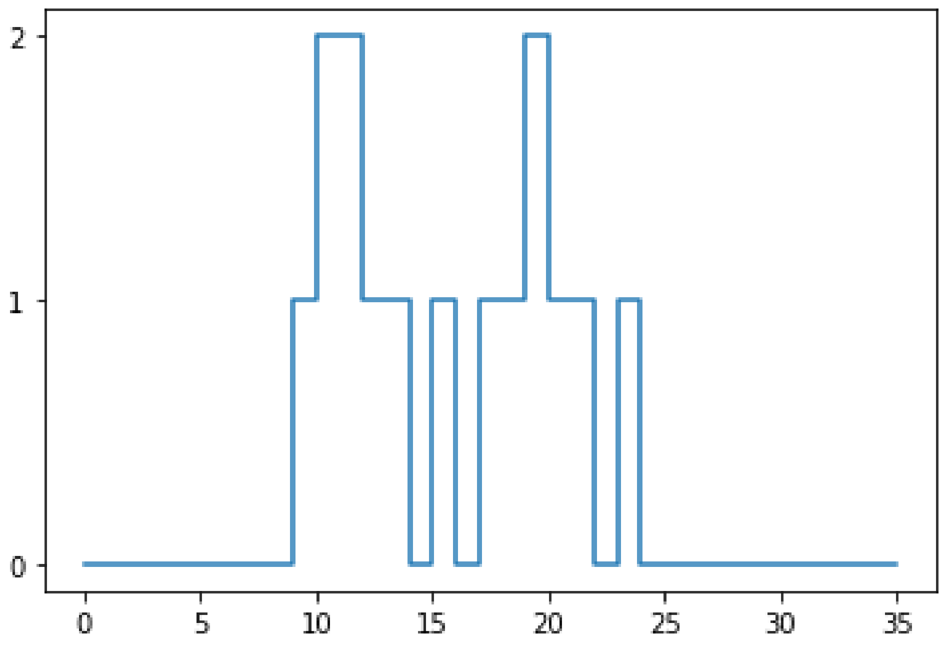
[67]:
```
df = pd.DataFrame(dict(A=A,S=S,U=U,D=D,W_q=W_q))
print(df.to_latex(index=False))
```

```
\begin{tabular}{rrrrr}
\toprule
  A &   S &   U &   D &  W\_q \\
\midrule
  0 &  12 &   0 &  12 &    0 \\
  6 &   8 &   6 &  14 &    0 \\
  9 &  12 &  12 &  24 &    3 \\
 10 &   2 &  14 &  16 &    4 \\
 15 &   4 &  16 &  20 &    1 \\
 17 &   2 &  20 &  22 &    3 \\
 19 &   6 &  22 &  28 &    3 \\
 23 &  10 &  24 &  34 &    1 \\
 29 &  16 &  29 &  45 &    0 \\
 35 &  12 &  35 &  47 &    0 \\
\bottomrule
\end{tabular}
```

[68]:
```
W_q_ave = sum(W_q)/len(W_q)
print(W_q_ave)
```

```
1.5
```

[69]:
```
plt.plot([0]+Ti,[0]+N_q,drawstyle='steps-post')
plt.yticks(range(max(N_q)+1))
plt.xticks(list(set([i*(i%5==0) for i in range(max(Ti)+1)])))
plt.show()
```



5 Problem 5

[70]:
```python
arr_frac = 0.5
dep_frac = 1 - arr_frac
arr_hold_mean = 1
dep_hold_mean = 1.5
rate_hr = 50
rate = rate_hr/60
service_mean = arr_frac * arr_hold_mean + dep_frac * dep_hold_mean
rho = rate*service_mean
print(rate, service_mean)
```

```
0.8333333333333334 1.25
```

[71]:
```python
num_ops = 400
N = 10000
```

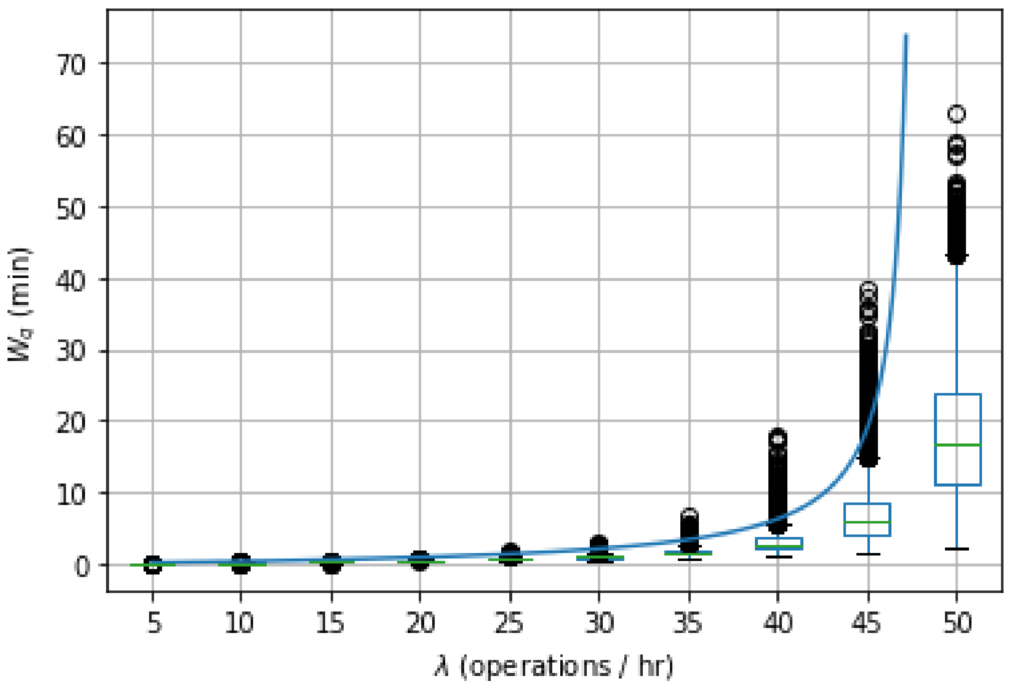
[72]:
```python
def sim(arr_frac, arr_hold_mean, dep_hold_mean, rate_hr, num_ops, exp_service =␣
 ↪True):
    rate = rate_hr/60
    T = np.random.exponential(1/rate,num_ops)
    op_type = np.random.binomial(1, arr_frac, num_ops)
    if exp_service:
        S = [np.random.exponential(arr_hold_mean) if b==1
            else np.random.exponential(dep_hold_mean) for b in op_type]
    else:
        S = [arr_hold_mean if b==1 else dep_hold_mean for b in op_type]
    A = [0] + list(np.cumsum(T[:-1]))
    D = [0 for i in range(len(A))]
    D[0] = S[0] + A[0]
    for i in range(1,len(A)):
        D[i] = S[i] + max(D[i-1], A[i])
    W_q_n = [d-s-a for d,s,a in zip(D,S,A)]
    return sum(W_q_n)/num_ops
```

[73]:
```
rates = [i*5 for i in range(1,11)]
```

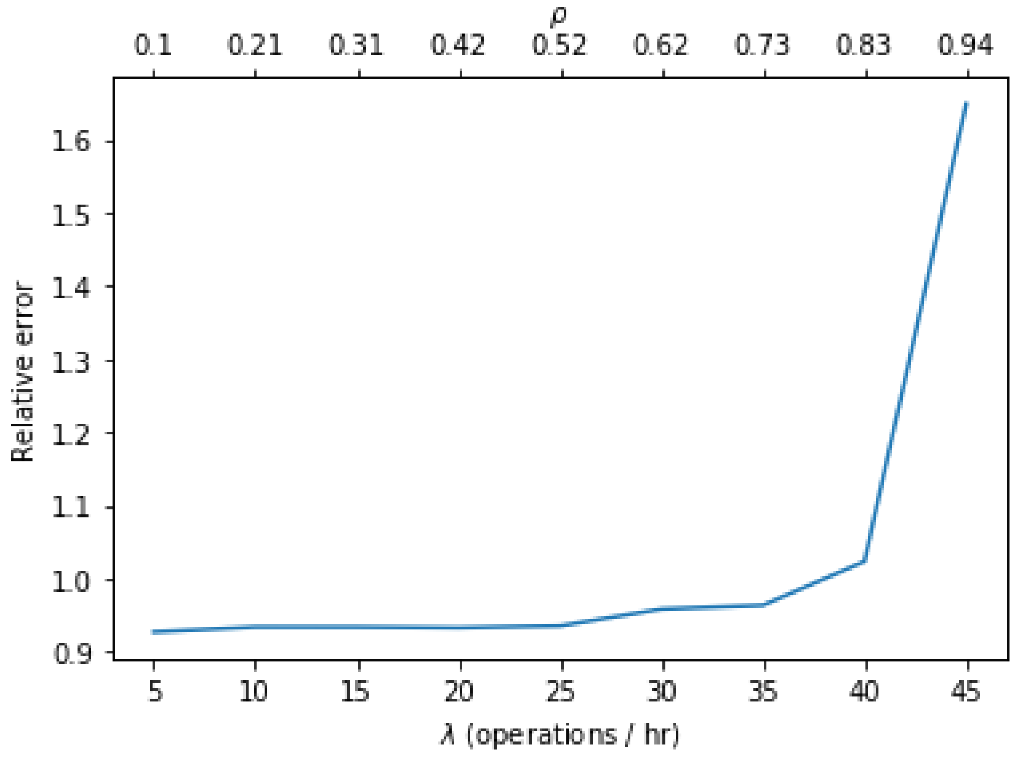
[74]:
```
data = [[sim(arr_frac, arr_hold_mean, dep_hold_mean, r, num_ops, False) for i␣
 ↪in range(N)] for r in rates]
```

[75]:
```
df = pd.DataFrame(data)
df = df.transpose()
df = df.rename(columns=dict(zip([i for i in range(10)],rates)))
```

[76]:
```
plt.figure()
df.boxplot()
x = np.arange(1, 9.45, 0.01)
y = [i*5*service_mean/60/(1/service_mean-i*5/60) for i in x]
plt.plot(x,y)
#plt.title('Average queue-waiting time')
plt.xlabel(r'$\lambda$'+' (operations / hr)')
plt.ylabel(r'$W_q$'+' (min)')
plt.savefig('W_q1.pdf')
plt.show()
```



```
[77]: m = [rate*service_mean/60/(1/service_mean-rate/60) for rate in rates][:-1]
      d = np.array(df.mean())[:-1]
      rho = np.array(rates)*service_mean/60
      err = abs((d-m)/d)
      #plt.title('Relative error by arrival rate')
      plt.ylabel('Relative error')
      plt.plot(rates[:-1],err)
      plt.xlabel(r'$\lambda$'+' (operations / hr)')
      plt.twiny()
      plt.xlabel(r'$\rho$')
      plt.plot(rates[:-1],err, alpha=0)
      plt.gca().set_xticklabels([0]+[round(r,2) for r in rho][:-1])
      plt.savefig('er1.pdf')
      plt.show()
```



[78]:
```
arr_frac = 0.25
dep_frac = 1 - arr_frac
arr_hold_mean = 1
dep_hold_mean = 1.5
rate_hr = 50
rate = rate_hr/60
service_mean = arr_frac * arr_hold_mean + dep_frac * dep_hold_mean
rho = rate*service_mean
print(rate, service_mean)
```

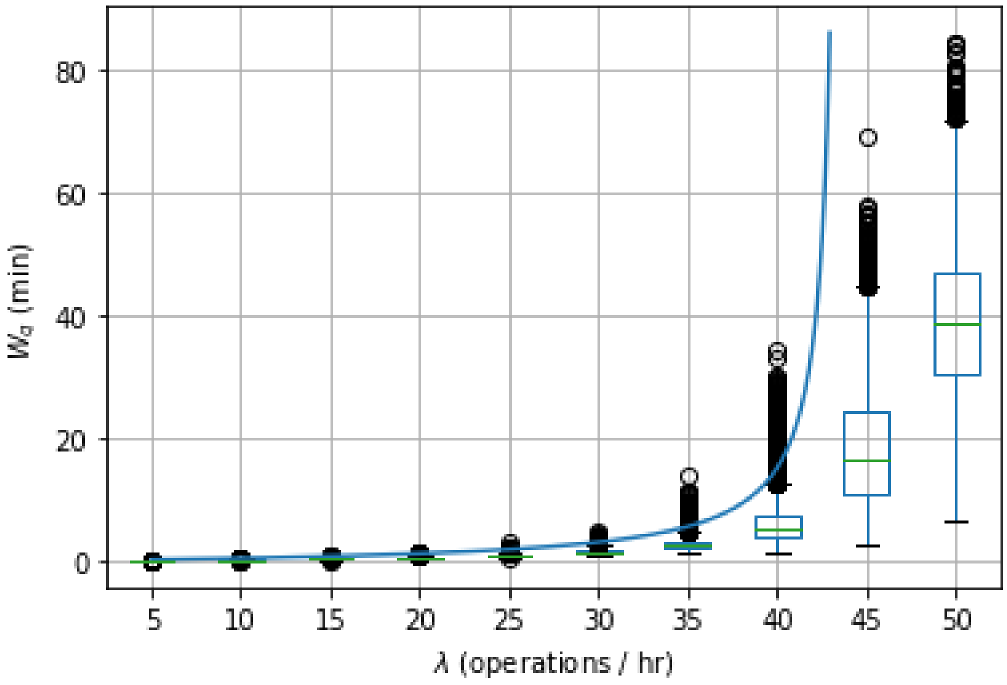
```
0.8333333333333334 1.375
```

[79]:
```
data = [[sim(arr_frac, arr_hold_mean, dep_hold_mean, r, num_ops, False) for i␣
 ↪in range(N)] for r in rates]
```

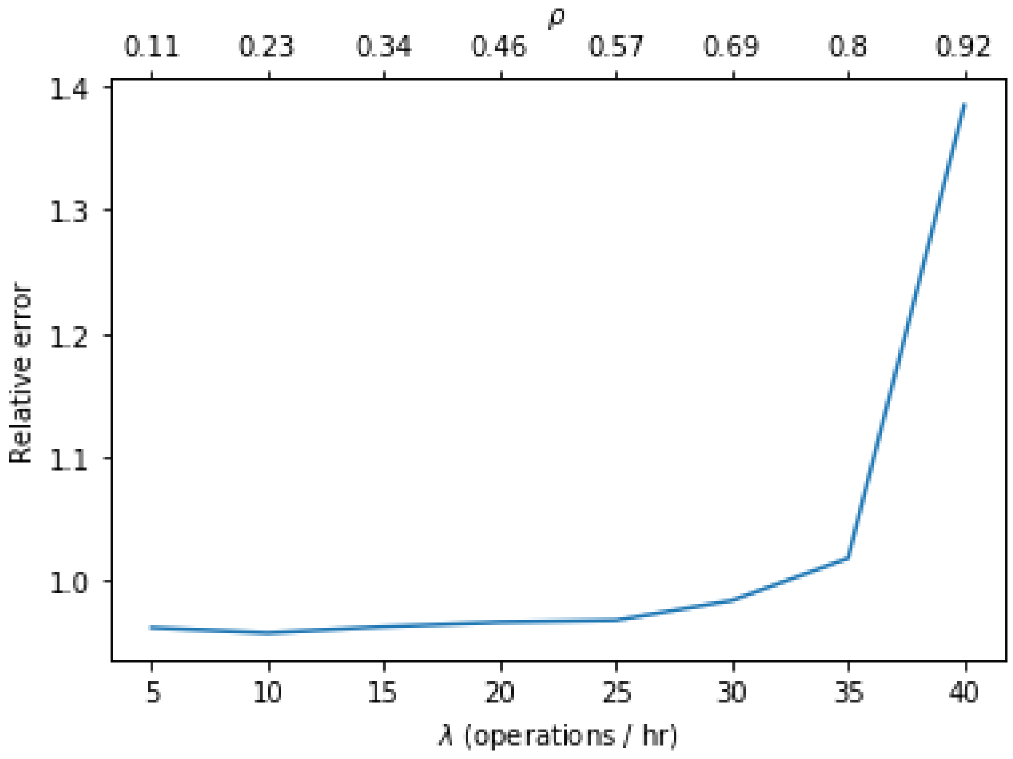
[80]:
```
df = pd.DataFrame(data)
df = df.transpose()
df = df.rename(columns=dict(zip([i for i in range(10)],rates)))
```

[81]:
```
df.boxplot()
x = np.arange(1, 8.6, 0.01)
y = [i*5*service_mean/60/(1/service_mean-i*5/60) for i in x]
```

```
plt.plot(x,y)
#plt.title('Average queue-waiting time')
plt.xlabel(r'$\lambda$'+' (operations / hr)')
plt.ylabel(r'$W_q$'+' (min)')
plt.savefig('W_q2.pdf')
plt.show()
```



[82]:
```
m = [rate*service_mean/60/(1/service_mean-rate/60) for rate in rates][:-2]
d = np.array(df.mean())[:-2]
rho = np.array(rates)*service_mean/60
err = abs((d-m)/d)
#plt.title('Relative error by arrival rate')
plt.ylabel('Relative error')
plt.plot(rates[:-2],err)
plt.xlabel(r'$\lambda$'+' (operations / hr)')
plt.twiny()
plt.xlabel(r'$\rho$')
plt.plot(rates[:-2],err, alpha=0)
plt.gca().set_xticklabels([0]+[round(r,2) for r in rho])
plt.savefig('er2.pdf')
plt.show()
```



```
[83]: arr_frac = 0.5
      dep_frac = 1 - arr_frac
      arr_hold_mean = 1
      dep_hold_mean = 1.5
      rate_hr = 50
      rate = rate_hr/60
      service_mean = arr_frac * arr_hold_mean + dep_frac * dep_hold_mean
      rho = rate*service_mean
      print(rate, service_mean)
```

```
0.8333333333333334 1.25
```

```
[84]: num_ops = 400
      N = 1000
```

```
[85]: def sim(arr_frac, arr_hold_mean, dep_hold_mean, rate_hr, num_ops, exp_service =␣
       ↪True):
          rate = rate_hr/60
          T = np.random.exponential(1/rate,num_ops)
          op_type = np.random.binomial(1, arr_frac, num_ops)
          if exp_service:
              S = [np.random.exponential(arr_hold_mean) if b==1
```

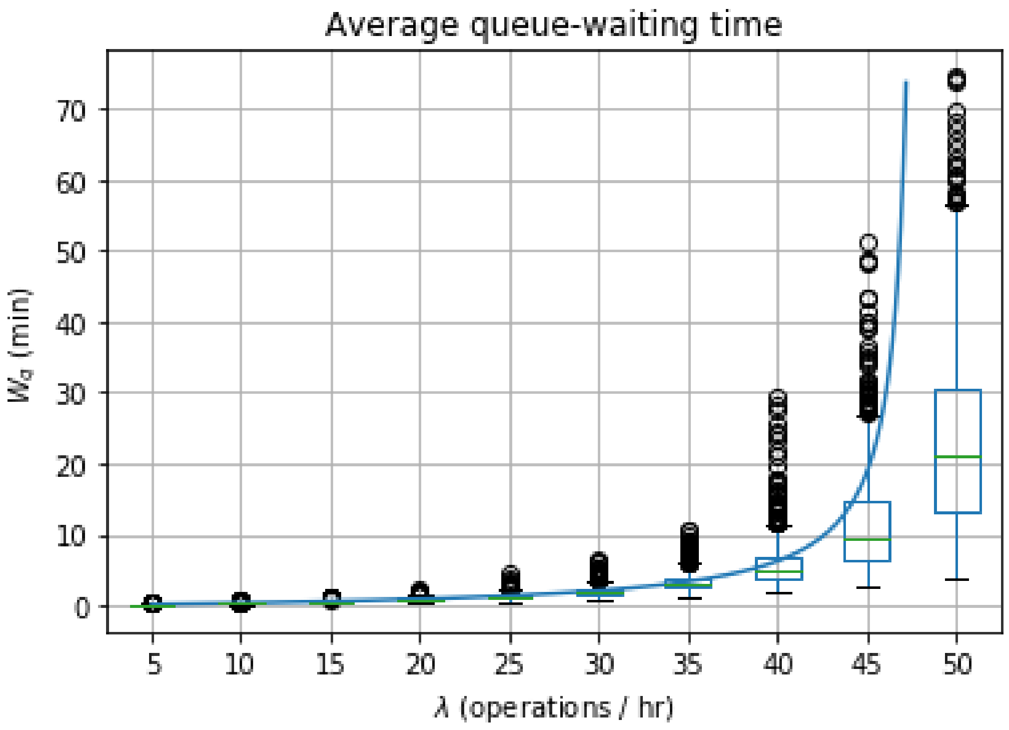
```
            else np.random.exponential(dep_hold_mean) for b in op_type]
    else:
        S = [arr_hold_mean if b==1 else dep_hold_mean for b in op_type]
    A = [0] + list(np.cumsum(T[:-1]))
    D = [0 for i in range(len(A))]
    D[0] = S[0] + A[0]
    for i in range(1,len(A)):
        D[i] = S[i] + max(D[i-1], A[i])
    W_q_n = [d-s-a for d,s,a in zip(D,S,A)]
    return sum(W_q_n)/num_ops
```

[86]:
```
rates = [i*5 for i in range(1,11)]
```

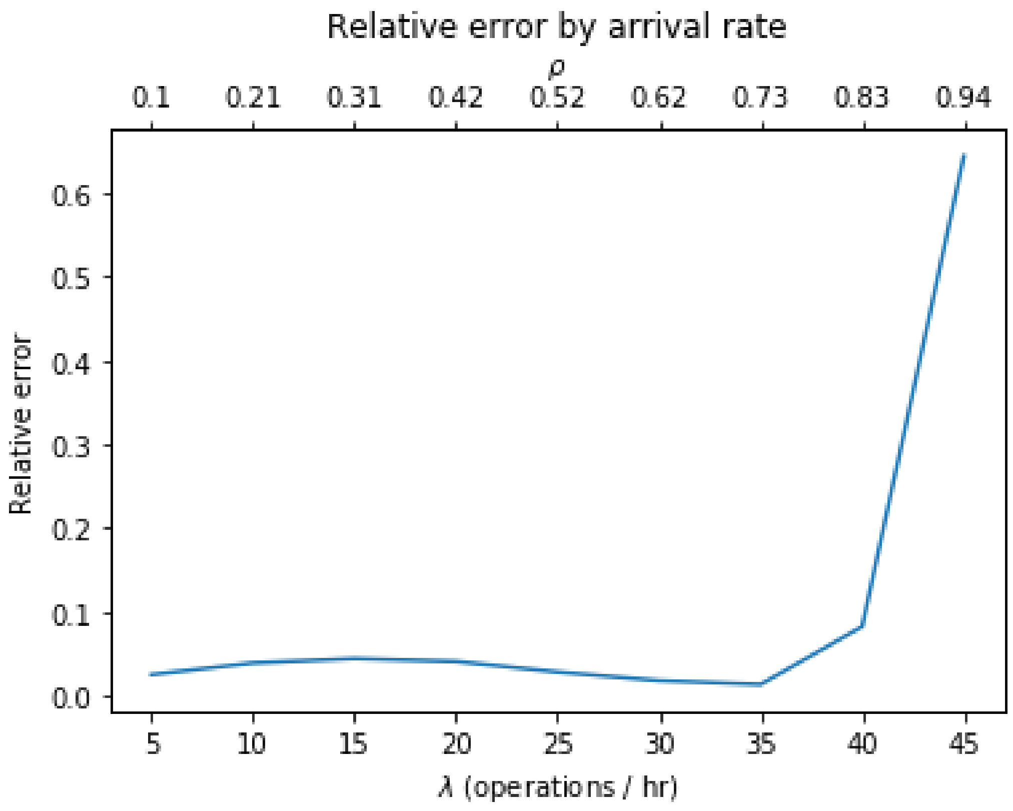
[87]:
```
data = [[sim(arr_frac, arr_hold_mean, dep_hold_mean, r, num_ops, True) for i in␣
 ↪range(N)] for r in rates]
```

[88]:
```
df = pd.DataFrame(data)
df = df.transpose()
df = df.rename(columns=dict(zip([i for i in range(10)],rates)))
```

[89]:
```
plt.figure()
df.boxplot()
x = np.arange(1, 9.45, 0.01)
y = [i*5*service_mean/60/(1/service_mean-i*5/60) for i in x]
plt.plot(x,y)
plt.title('Average queue-waiting time')
plt.xlabel(r'$\lambda$'+' (operations / hr)')
plt.ylabel(r'$W_q$'+' (min)')
plt.show()
```



[90]:
```
m = [rate*service_mean/60/(1/service_mean-rate/60) for rate in rates][:-1]
d = np.array(df.mean())[:-1]
rho = np.array(rates)*service_mean/60
err = abs((d-m)/d)
plt.title('Relative error by arrival rate')
plt.ylabel('Relative error')
plt.plot(rates[:-1],err)
plt.xlabel(r'$\lambda$'+' (operations / hr)')
plt.twiny()
plt.xlabel(r'$\rho$')
plt.plot(rates[:-1],err, alpha=0)
plt.gca().set_xticklabels([0]+[round(r,2) for r in rho][:-1])
plt.show()
```



[91]:
```
arr_frac = 0.25
dep_frac = 1 - arr_frac
arr_hold_mean = 1
dep_hold_mean = 1.5
rate_hr = 50
rate = rate_hr/60
service_mean = arr_frac * arr_hold_mean + dep_frac * dep_hold_mean
rho = rate*service_mean
print(rate, service_mean)
```

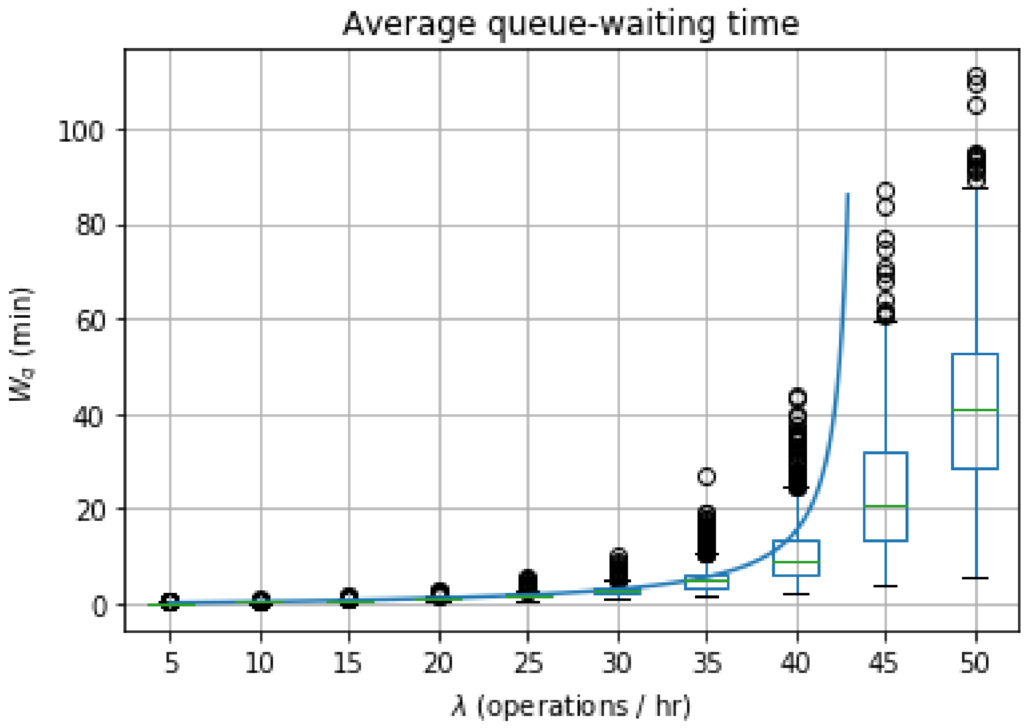
```
0.8333333333333334 1.375
```

[92]:
```
data = [[sim(arr_frac, arr_hold_mean, dep_hold_mean, r, num_ops, True) for i in␣
 ↪range(N)] for r in rates]
```

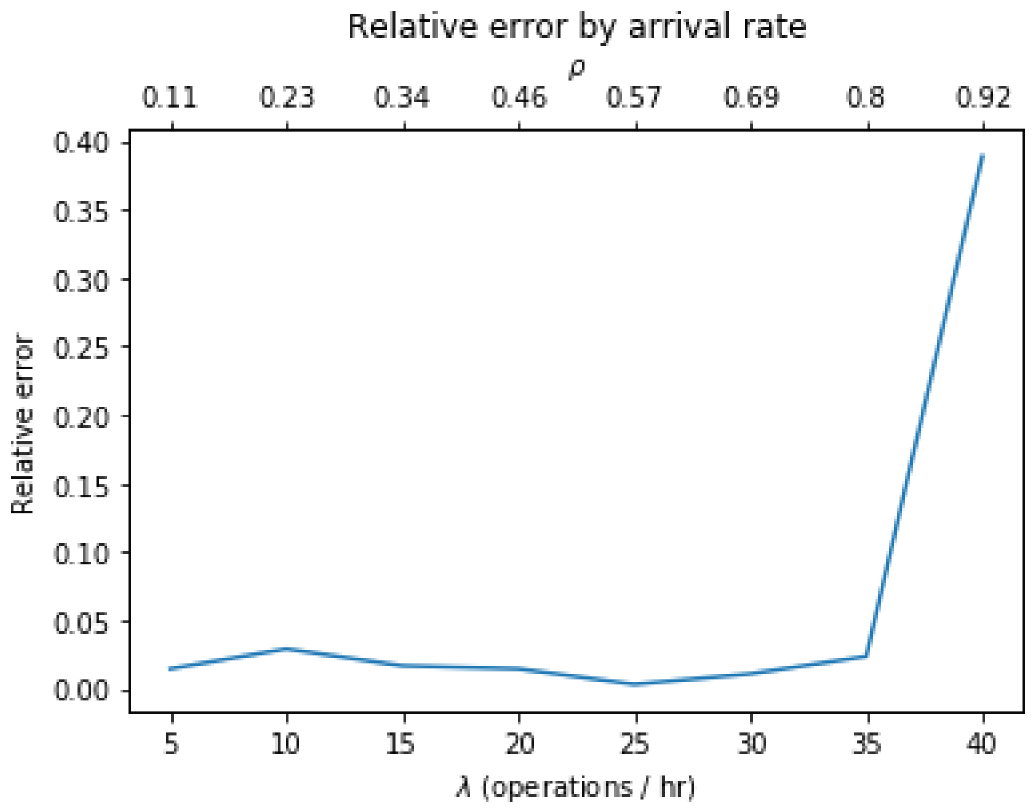
[93]:
```
df = pd.DataFrame(data)
df = df.transpose()
df = df.rename(columns=dict(zip([i for i in range(10)],rates)))
```

[94]:
```
df.boxplot()
x = np.arange(1, 8.6, 0.01)
```

```
y = [i*5*service_mean/60/(1/service_mean-i*5/60) for i in x]
plt.plot(x,y)
plt.title('Average queue-waiting time')
plt.xlabel(r'$\lambda$'+' (operations / hr)')
plt.ylabel(r'$W_q$'+' (min)')
plt.show()
```



[95]:
```
m = [rate*service_mean/60/(1/service_mean-rate/60) for rate in rates][:-2]
d = np.array(df.mean())[:-2]
rho = np.array(rates)*service_mean/60
err = abs((d-m)/d)
plt.title('Relative error by arrival rate')
plt.ylabel('Relative error')
plt.plot(rates[:-2],err)
plt.xlabel(r'$\lambda$'+' (operations / hr)')
plt.twiny()
plt.xlabel(r'$\rho$')
plt.plot(rates[:-2],err, alpha=0)
plt.gca().set_xticklabels([0]+[round(r,2) for r in rho])
plt.show()
```



[96]:
```
arr_frac = 0.5
dep_frac = 1 - arr_frac
arr_hold_mean = 1
dep_hold_mean = 1.5
rate_hr = 50
rate = rate_hr/60
service_mean = arr_frac * arr_hold_mean + dep_frac * dep_hold_mean
rho = rate*service_mean
print(rate, service_mean)
```

```
0.8333333333333334 1.25
```

[97]:
```
T = np.random.exponential(1/rate,num_ops)
np.mean(T)
```

[97]:
```
1.3176689143665428
```

[98]:
```
op_type = np.random.binomial(1, arr_frac, num_ops)
np.mean(op_type)
```

[98]:
```
0.5025
```

```
[99]: S = [arr_hold_mean if b==1 else dep_hold_mean for b in op_type]
      np.mean(S)
```

```
[99]: 1.24875
```

```
[100]: #S = [np.random.exponential(arr_hold_mean)
       #              if b==1
       #              else np.random.exponential(dep_hold_mean) for b in op_type]
       #np.mean(S)
```

```
[101]: A = [0] + list(np.cumsum(T[:-1]))
```

```
[102]: D = [0 for i in range(len(A))]
       D[0] = S[0] + A[0]
       for i in range(1,len(A)):
           D[i] = S[i] + max(D[i-1], A[i])
```

```
[103]: D[-1]
```

```
[103]: 551.5958091442158
```

```
[104]: W_q_n = [d-s-a for d,s,a in zip(D,S,A)]
```

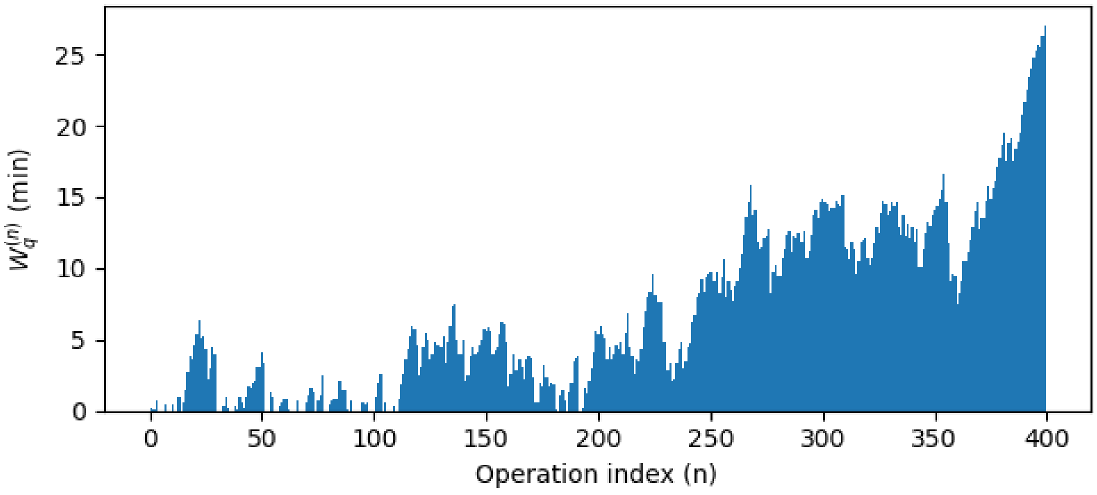
```
[105]: W_q = sum(W_q_n)/D[-1]
       print(W_q)
```

```
4.952185617364083
```

```
[106]: [i in range(len(W_q_n))]
```

```
[106]: [True]
```

```
[107]: plt.figure(figsize=(6.4,3),dpi=100)
       plt.bar([i for i in range(len(W_q_n))],W_q_n,1)
       #plt.title('Sample queue-waiting times')
       plt.ylabel(r'$W_q^{(n)}$'+' (min)')
       plt.xlabel('Operation index (n)')
       plt.tight_layout()
       #plt.savefig('path.pdf')
       plt.show()
```



6 Problem 6

[109]:
```python
c = 3

lam = 45/60
mu = 1/3
print(lam, mu)
```

```
0.75 0.3333333333333333
```

[110]:
```python
r = lam / mu
print(r)
```

```
2.25
```

[111]:
```python
rho = r / c
print(rho)
```

```
0.75
```

[112]:
```python
n = r**c
d = factorial(c)*(1-rho)
s = sum([r**i/factorial(i) for i in range(c)])
p_0 = 1 / (n/d + s)
print(p_0)
```

```
0.07476635514018691
```

[113]:
```
L_q = n*rho/(d *(1-rho))*p_0
print(L_q)
```

```
1.703271028037383
```

[114]:
```
W = (1/mu) + L_q/lam
print(W)
```

```
5.271028037383177
```

7 Problem 7

[115]:
```
c = 5
lam = 75/60
mu = 1/2
print(lam, mu)
```

```
1.25 0.5
```

[116]:
```
r = lam / mu
print(r)
```

```
2.5
```

[117]:
```
rho = r / c
print(rho)
```

```
0.5
```

7.1 7 (b)

[118]:
```
n = r**c
d = factorial(c)*(1-rho)
s = sum([r**i/factorial(i) for i in range(c)])
p_0 = 1 / (n/d + s)
print(p_0)
```

```
0.08010012515644557
```

[119]:
```
p_n = sum([lam**i/factorial(i)/mu**i*p_0 for i in range(4)])
print(p_n)
```

```
0.7392574050896956
```

7.2 7 (c)

```
[120]: L_q = n*rho/(d *(1-rho))*p_0
       print(L_q)
```

```
0.1303712974551523
```

8 Problem 8

```
[121]: lam = 18
       mu = 20
       print(lam, mu)
```

```
18 20
```

```
[122]: r = lam / mu
       print(r)
```

```
0.9
```

8.1 8 (a)

```
[123]: c = 1
       rho = r / c
       print(rho)
```

```
0.9
```

```
[124]: n = r**c
       d = factorial(c)*(1-rho)
       s = sum([r**i/factorial(i) for i in range(c)])
       p_0 = 1 / (n/d + s)
       print(p_0)
```

```
0.09999999999999998
```

```
[125]: L_q = n*rho/(d *(1-rho))*p_0
       print(L_q)
```

```
8.100000000000003
```

```
[126]: L_q = rho**2/(1-rho)
       print(L_q)
```

```
8.100000000000003
```

```
[127]: r * 40 + (c - r) * 10 + L_q * 2
```

```
[127]: 53.2
```

8.2 8 (b)

```
[128]: c = 2
       rho = r / c
       print(rho)
```

```
0.45
```

```
[129]: n = r**c
       d = factorial(c)*(1-rho)
       s = sum([r**i/factorial(i) for i in range(c)])
       p_0 = 1 / (n/d + s)
       print(p_0)
```

```
0.37931034482758624
```

```
[130]: L_q = n*rho/(d *(1-rho))*p_0
       print(L_q)
```

```
0.22852664576802512
```

```
[131]: r * 40 + (c - r) * 10 + L_q * 2
```

```
[131]: 47.45705329153605
```

9 Problem 9 (3.32)

```
[132]: def approx(alpha, beta):
           return norm.pdf(beta)/(norm.pdf(beta) + beta * norm.cdf(beta)) - alpha
```

9.1 9 (a)

```
[133]: lam = 500 / 60
       mu = 1 / 2
       print(lam, mu)
```

```
8.333333333333334 0.5
```

```
[134]: r = lam / mu
       print(r)
```

```
16.666666666666668
```

```
[135]: beta = newton(lambda beta: approx(0.10, beta), 1)
       print(beta)
```

```
1.4201868881004411
```

```
[136]: c_approx = r + beta * sqrt(r)
       print(c_approx)
```

```
22.464555358728653
```

9.2 9 (b)

```
[137]: lam = 500 / 60 * 1.6
       mu = 1 / 2
       print(lam, mu)
```

```
13.333333333333336 0.5
```

```
[138]: r = lam / mu
       print(r)
```

```
26.66666666666667
```

```
[139]: c_approx = r + beta * sqrt(r)
       print(c_approx)
```

```
34.00048022148687
```

9.3 9 (c)

```
[140]: lam = 500 / 60
       mu = 1 / 3
       print(lam, mu)
```

```
8.333333333333334 0.3333333333333333
```

```
[141]: r = lam / mu
       print(r)
```

```
25.000000000000004
```

```
[142]: c_approx = r + beta * sqrt(r)
       print(c_approx)
```

```
32.10093444050221
```

9.4 9 (d)

[143]:
```
lam = 500 / 60
mu = 1 / 2
print(lam, mu)
```

```
8.333333333333334 0.5
```

[144]:
```
r = lam / mu
print(r)
```

```
16.666666666666668
```

[145]:
```
c = 24
rho = r / c
print(rho)
```

```
0.6944444444444445
```

[146]:
```
n = r**c
d = factorial(c)*(1-rho)
s = sum([r**i/factorial(i) for i in range(c)])
p_0 = 1 / (n/d + s)
print(p_0)
```

```
5.714528177635283e-08
```

[147]:
```
C_c_r = n*p_0/d
print(C_c_r)
```

```
0.06361426515341066
```

[148]:
```
L_q = n*rho/(d*(1-rho))*p_0
print(L_q)
```

```
0.14457787534866065
```

[149]:
```
W_q = L_q / lam
print(W_q)
```

```
0.017349345041839277
```

[150]:
```
lam = 500 / 60 * 1.6
mu = 1 / 2
print(lam, mu)
```

```
13.333333333333336 0.5
```

[151]:
```
r = lam / mu
print(r)
```

```
26.666666666666667
```

[152]:
```
c = 35
rho = r / c
print(rho)
```

```
0.7619047619047621
```

[153]:
```
n = r**c
d = factorial(c)*(1-rho)
s = sum([r**i/factorial(i) for i in range(c)])
p_0 = 1 / (n/d + s)
print(p_0)
```

```
2.5787312412854715e-12
```

[154]:
```
C_c_r = n*p_0/d
print(C_c_r)
```

```
0.08498227504737443
```

[155]:
```
L_q = n*rho/(d *(1-rho))*p_0
print(L_q)
```

```
0.2719432801515984
```

[156]:
```
W_q = L_q / lam
print(W_q)
```

```
0.020395746011369877
```

10 Problem 10 (3.37)

[157]:
```
beta = newton(lambda beta: approx(0.05, beta), 1)
print(beta)
```

```
1.7398362717905036
```

10.1 10 (a)

[158]:
```
lam = 300 / 60
mu = 1 / 2
print(lam, mu)
```

```
5.0 0.5
```

[159]:
```
r = lam / mu
print(r)
```

```
10.0
```

[160]:
```
c_approx = r + beta * sqrt(r)
print(c_approx)
```

```
15.50184537463375
```

10.2 10 (b)

[161]:
```
lam = 300 / 60 * 2
mu = 1 / 2
print(lam, mu)
```

```
10.0 0.5
```

[162]:
```
r = lam / mu
print(r)
```

```
20.0
```

[163]:
```
c_approx = r + beta * sqrt(r)
print(c_approx)
```

```
27.780784346886733
```