

COMP9334-Project

ZHENG FUDI 5126586

Part1:Correctness of simulation code

1. the correctness of the inter arrival probability distribution

There is one module named random which implements pseudo-random number generators for various distributions. The library of random.expovariate(lambd) can give us random floating point numbers, exponentially distributed.

```
lambd = arrival_rate
u = service_rate
# distribution_arrival = []
next_arrival = 0
while next_arrival < time_end:
    inter_arrival_time=random.expovariate(lambd)
    # distribution_arrival.append(inter_arrival_time)
    next_arrival = next_arrival + inter_arrival_time
    arrivals_list.append(next_arrival)
# plt.hist(distribution_arrival, bins=50, edgecolor='k')
# plt.show()
```

Figure 1: generate the arrival time list before time end

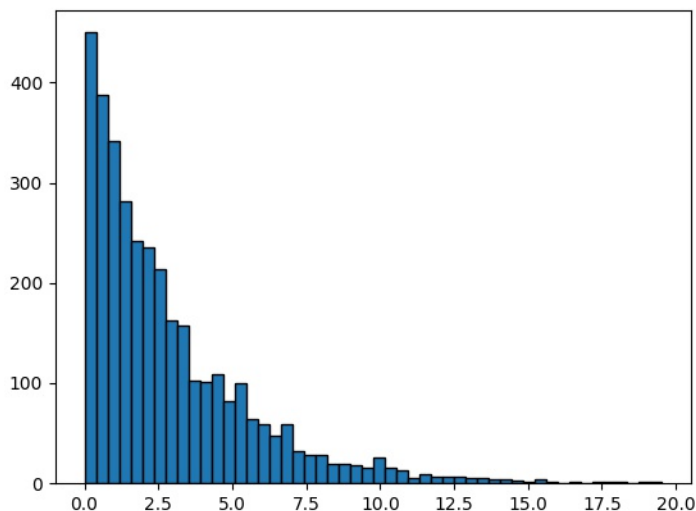


Figure 2:Histogram of exponentially distributed psuedo-random numbers

2. the correctness of the service time distribution

Independent exponentially distributed numbers can also be generated through `random.expovariate(lambd)`. The easiest way to generate the service time is to generate three independent exponentially distributed numbers and add them up.

```
77 import matplotlib.pyplot as plt
78 import random
79 for i in range(len(arrivals_list)):
80     service_time = 0
81     for j in range(3):
82         x = random.expovariate(u)
83         service_time += x
84     service_list.append(service_time)
85 plt.hist(service_list, bins=50, edgecolor='k')
86 plt.show()
87
```

Figure 3: generate the service time list before time end

line 80-83: these three lines are used to generate service time. Because we need to generate three independent exponentially distributed numbers, we need to loop 3 times to generate every independent exponentially distributed number and add them up, which can be achieved by line 81 and line 83.

After adding independent exponentially distributed random variables, we will get phase type distribution.

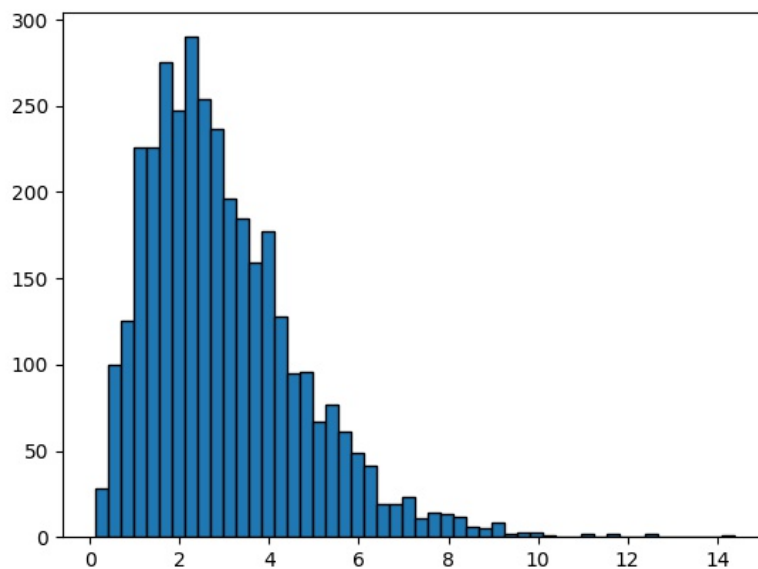


Figure 4: Histogram of phase type distribution

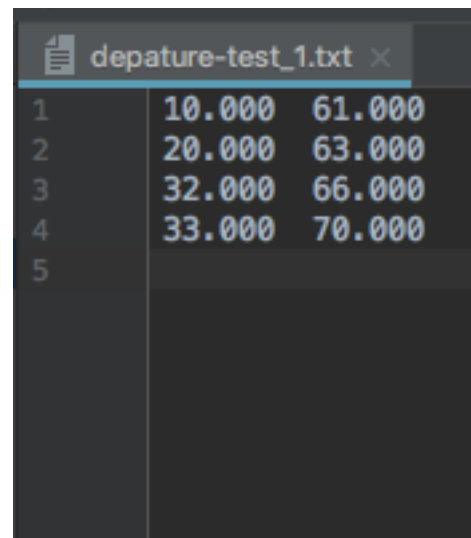
3. verified the correctness of your simulation code

Example1:

Number of servers	Setup time	Tc
3	50	100

Arrival time	Service time
10	1
20	2
32	3
33	4

arrival time	service time	departure time
10	1	61
20	2	63
32	3	66
33	4	70



	arrival time	service time	departure time
1	10.000	1	61.000
2	20.000	2	63.000
3	32.000	3	66.000
4	33.000	4	70.000
5			

Example2:

Number of servers	Setup time	Tc
5	5	10

Arrival time	Service time
10	2
18	4
20	14
23	5
28	6
32	21
33	2
34	16
35	9

derive test cases

master clock	queue	s1	s2	s3	s4	s5
10	10,2,marked	setup,15				
15		busy,17(10)				
17		delayoff,27				
18		busy,22(18)				
20	20,14,marked	busy,22(18)	setup,25			
22		busy,36(20)	off			
23	23,5,marked	busy,36(20)	setup,28			
28	28,6,marked	busy,36(20)	busy,33(23)	setup,33		
32	28,6,marked	busy,36(20)	busy,33(23)	setup,33	setup,37	
	32,21,marked					
33	33,2,marked	busy,36(20)	busy,39(28)	busy,54(32)	setup,37	
34	33,2,marked	busy,36(20)	busy,39(28)	busy,54(32)	setup,37	setup,39
	34,16,marked					
35	33,2,marked	busy,36(20)	busy,39(28)	busy,54(32)	setup,37	setup,39

	34,16,marked					
	35,9,unmarked					
36	34,16,marked	busy,38(33)	busy,39(28)	busy,54(32)	setup,37	setup,39
	35,9,marked					
37	35,9,marked	busy,38(33)	busy,39(28)	busy,54(32)	busy,53(34)	setup,39
38		busy,47(35)	busy,39(28)	busy,54(32)	busy,53(34)	off
39		busy,47(35)	delayoff,49	busy,54(32)	busy,53(34)	off
47		delayoff,57	delayoff,49	busy,54(32)	busy,53(34)	off
49		delayoff,57	off	busy,54(32)	busy,53(34)	off
53		delayoff,57	off	busy,54(32)	delayoff,63	off
54		delayoff,57	off	delayoff,64	delayoff,63	off
57		off	off	delayoff,64	delayoff,63	off
63		off	off	delayoff,64	off	off
64		off	off	off	off	off

arrival time	service time	departure time
10	2	17
18	4	22
20	14	36
23	5	33
28	6	39
32	21	54
33	2	38
34	16	53
35	9	47

depature-test_3.txt		
1	10.000	17.000
2	18.000	22.000
3	23.000	33.000
4	20.000	36.000
5	33.000	38.000
6	28.000	39.000
7	35.000	47.000
8	34.000	53.000
9	32.000	54.000
10		

Part2: determining a suitable value of T_c

1. Find baseline system mean response time and appropriate time (T_c)

less than that of the baseline system

λ	μ	Number of serves	Setup time	Time end	seed
0.35	1	5	5	10000	1

T_c	Mean response time
0.1	6.053
0.2	6.018
1	5.734
2	5.348
3	5.100
4	4.863
5	4.681
6	4.530
7	4.393
8	4.247
9	4.134
10	4.086
11	4.007
12	3.947
13	3.913
14	3.862
15	3.801

Assume the following parameter values: the number of servers is 5, setup time is 5, $\lambda = 0.35$, $\mu = 1$. baseline system uses $T_c = 0.1$

From the above table, we can conclude that the mean response time is 6.053.

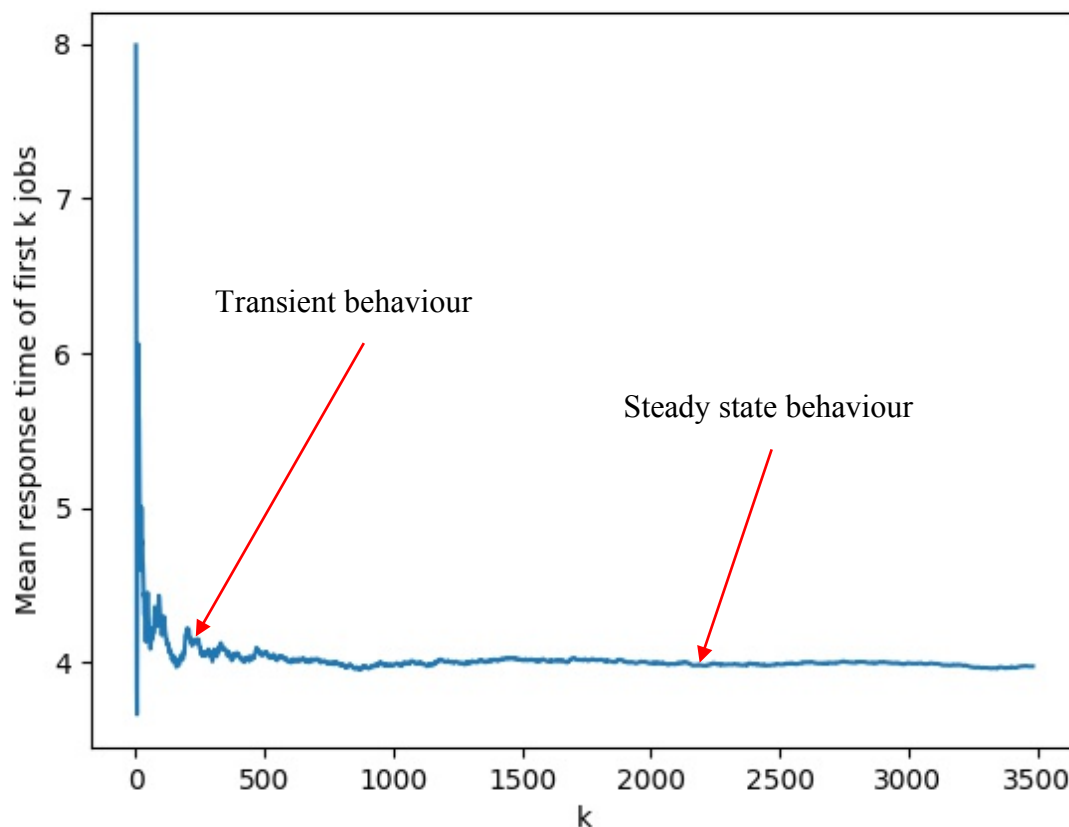
Because the aim is to design an improved system which uses a higher value of T_c so as to reduce the response time. Our aim is to determine a value of T_c (or a range for

T_c) so that the improved system's response time must be 2 units less than that of the baseline system.

It can be seen from the table that $T_c=11$ is appropriate time, which can be 2 units less than that of the baseline system.

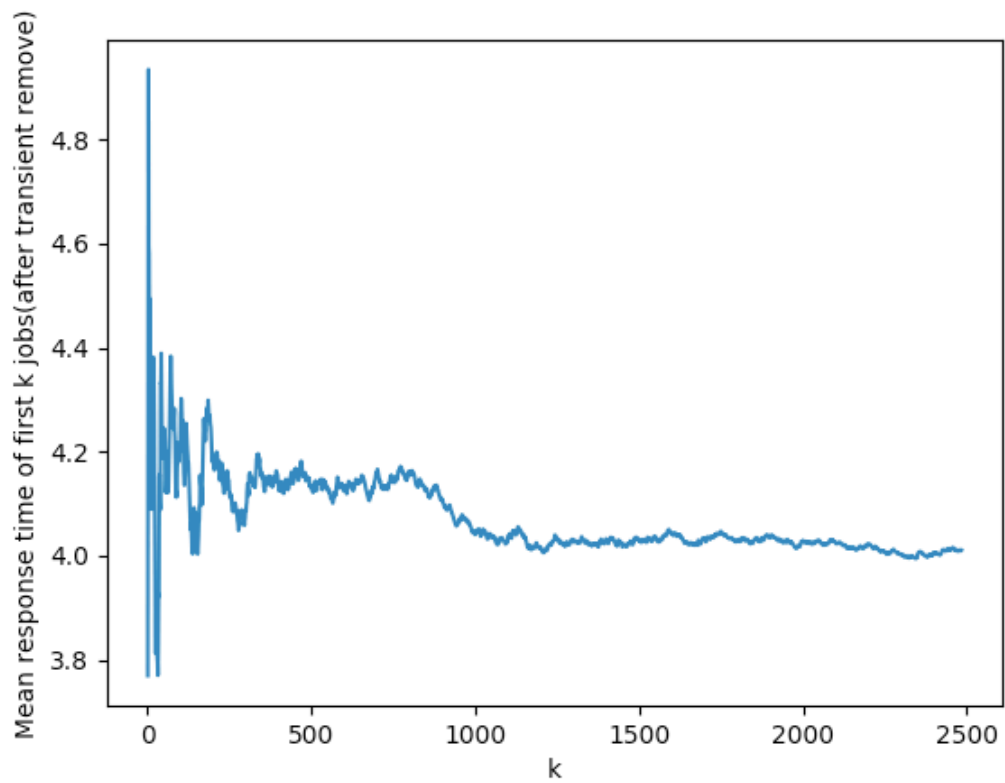
2. transient removals

λ	μ	Number of serves	Setup time	Time end	seed	T_c
0.35	1	5	5	10000	1	11



The early part of the simulation displays transient. The later part of the simulation converges or fluctuates around the steady state value. Since we are interested in the steady state value, we should not use the transient part of the data to compute the steady state value. We should remove the transient part and only use the steady state part to compute the mean. One method to identify the transient part is to use visual inspection

It can be seen from the line chart that first 1000 jobs can be removed.

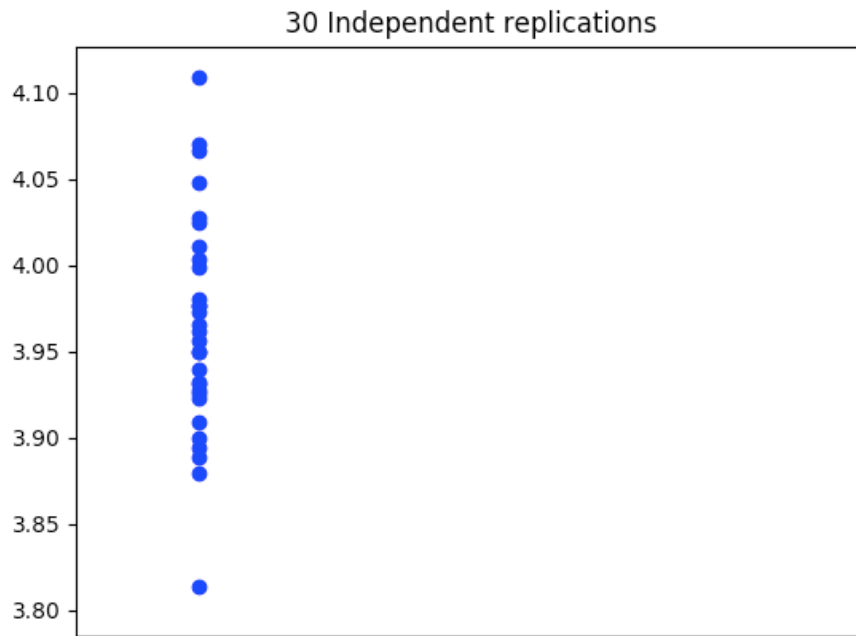


3. 30 independent replications

λ	μ	serves	Setup time	Time end	Tc	Jobs remove
0.35	1	5	5	10000	11	1000

Seed	Mean response time
1	4.011
2	3.926
3	4.028
4	3.879
5	3.956
6	3.999
7	3.928
8	3.909
9	3.980

10	3.900
11	3.950
12	3.940
13	4.048
14	3.977
15	4.070
16	3.962
17	4.004
18	3.932
19	3.894
20	3.950
21	4.109
22	4.067
23	3.931
24	3.814
25	3.889
26	3.923
27	3.966
28	4.025
29	3.973
30	3.977



```
import matplotlib.pyplot as plt
fig = plt.figure()
ax1 = fig.add_subplot(111)
y=[4.011,3.926,4.028,3.879,3.956,3.999,3.928,3.909,3.980,3.900,
    3.950,3.940,4.048,3.977,4.070,3.962,4.004,3.932,3.894,3.950,
    4.109,4.067,3.931,3.814,3.889,3.923,3.966,4.025,3.973,3.977]
x=[]
for i in range(len(y)):
    x.append(0.1)
#set title
ax1.set_title('30 Independent replications')
#set xlabel
plt.xlabel('X')
#set ylabel
plt.ylabel('Y')
#draw scatter
ax1.scatter(x,y,c='b',marker='o')
#draw graph
plt.show()
```

The blue circles show the estimated mean response time from the 30 independent experiments.

4.Computing the confidence interval

In each replication, After removing the transient part and compute an estimate of the mean steady state response time equal= 3.9639

Let us call your estimate from the kth replication, $T(k)$

$$\hat{T} = \frac{\sum_{i=1}^n T(i)}{n}$$

the sample standard deviation

$$\hat{S} = \sqrt{\frac{\sum_{i=1}^n (\hat{T} - T(i))^2}{n - 1}}$$

```
import statistics
y=[4.011,3.926,4.028,3.879,3.956,3.999,3.928,3.909,3.980,3.900,
  3.950,3.940,4.048,3.977,4.070,3.962,4.004,3.932,3.894,3.950,
  4.109,4.067,3.931,3.814,3.889,3.923,3.966,4.025,3.973,3.977]
mean_mrt=statistics.mean(y)
s=statistics.stdev(y)
```

`statistics.mean(data)`

Return the sample arithmetic mean of *data* which can be a sequence or iterator.

The arithmetic mean is the sum of the data divided by the number of data points. It is commonly called “the average”, although it is only one of many different mathematical averages. It is a measure of the central location of the data.

`statistics.stdev(data, xbar=None)`

Return the sample standard deviation (the square root of the sample variance).

See `variance()` for arguments and other details

T=3.9639(the sample arithmetic mean)

S=0.06389557752760522(the sample standard deviation)

There is a probability $(1-\alpha)$ that the mean response time that you want to estimate lies in the interval

$$\left[\hat{T} - t_{n-1, 1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}}, \hat{T} + t_{n-1, 1-\frac{\alpha}{2}} \frac{\hat{S}}{\sqrt{n}} \right]$$

- The sample mean of (n =) 30 replications =3.964

- The sample standard deviation of 30 replications is 0.0639
- If we want to compute the 95% confidence interval, $\alpha = 0.05$
- Since we did 30 independent experiments and want 95% confidence interval, we use $t(29,0975)$.

From the t-distribution table, the value of 2.0452, the 95% confidence interval is $[3.964 - 2.0452 * (0.0639 / \sqrt{30}), 3.964 + 2.0452 * (0.0639 / \sqrt{30})] = [3.940, 3.988]$

```
>>> 3.964-2.0452*(0.0639/(30**(1/2)))
3.940139693680816
```

```
>>> 3.964+2.0452*(0.0639/(30**(1/2)))
3.9878603063191838
```

λ	μ	serves	Setup time	Time end	Tc	Jobs remove
0.35	1	5	5	10000	11	1000

There is a 95% probability that the true mean response time that we want to estimate is in the interval $[3.940, 3.988]$ under the conditions from the above table.