Simulation Project Report

ZHU Jiarui
WU Yuheng
ZHANG Wenkai
CHEN Xiangyue

Assignment1:

Question 1: name at least two other blocking disciplines

1. Clear the queue if there is blocking
2. Skip the current event if there is blocking

Question 2: Write a DES simulation of the system

To formulate the question as a DES algorithm:

- We define a class called “station” and define all the variables we need.
- Four single-server stations. Staionid = {0,1,2,3}. n_server = 1 for each station.
- Each station has a different buffer capacity. n_buffer = {5,6,6,3} for each station.
- Each station’s event processing time follows exponential distribution with different parameters. Process_param = {2.9,2.2,2.2,2.8}
- The arriving process of each event follows Poisson distribution with □=2.1
- S = {□,□} . □-event stands for ‘finish consulting’; □-event stands for ‘new coming’
- Algorithm:
    - At the beginning, we set the number of customers as 20, 20 □-events are generated with arrival process being Poisson distribution.
    - A timer list is created to store all the following events and each element in the list is in the format (time point, event, staionid, serverid).
    - We sort those events by their arrival time length in ascending order and let those events come to the server one by one.
    - If the coming event is a □-event.
        - When the waiting line is not empty and still has capacity. Then the current status of the server puls one.
        - When the waiting line is not empty and it is full. Return False
        - When the waiting line is empty. Then we check whether the server is in service or not. If it is in service, the new arrival will be placed into waiting line. On the contrary, we add an □-event into the timer list.
    - If the coming event is an □-event
        - If the station doesn't have a next station, i.e. the consumer can leave immediately after □-event happens. Then check whether there is someone waiting in the queue. When the queue is empty, set the server to be empty; when the queue is noe empty, move one into the server which was just cleaned up. The first number of current status minus one, and a new □-event is added into the timeline with a new time of occurrence accordingly.
        - If the station has a next station, i.e. the system need to check whether the consumer could enter the next station before nove it out of current station. When the beta event at current time happens successfully in next station, the process is the same as □-event in a station without next station. When

the beta event for next station is not successful, check the timeline and find out the earliest time when the next station will have an empty space. reset the □-event to that time. Because the server is still occupied, it cannot keep on moving toward.

Question 3: Run your simulation and estimate the throughput

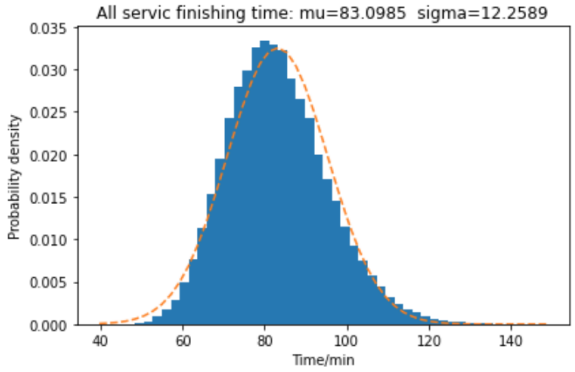
The estimate output of running a single simulation

```
In [3]: 1 simulation.run(n_customers=20)
        time:72.6794    station 2: [2, [1]]
        [(75.60810054159828, 'alpha', 3, 0), (75.72405058727752, 'alpha', 2, 0)]
        time:75.6081    station 3: [0, [0]]
        [(75.72405058727752, 'alpha', 2, 0)]
        time:75.7241    station 3: [0, [1]]
        time:75.7241    station 2: [1, [1]]
        [(76.54421069106735, 'alpha', 2, 0), (78.13727371297966, 'alpha', 3, 0)]
        time:76.5442    station 3: [1, [1]]
        time:76.5442    station 2: [0, [1]]
        [(78.13727371297966, 'alpha', 3, 0), (78.763931502669, 'alpha', 2, 0)]
        time:78.1373    station 3: [0, [1]]
        [(78.48664244998979, 'alpha', 3, 0), (78.763931502669, 'alpha', 2, 0)]
        time:78.4866    station 3: [0, [0]]
        [(78.763931502669, 'alpha', 2, 0)]
        time:78.7639    station 3: [0, [1]]
        time:78.7639    station 2: [0, [0]]
        [(79.5660020431916, 'alpha', 3, 0)]
        time:79.5660    station 3: [0, [0]]

Out[3]: 79.5660020431916
```

Question 4: Output analysis
Iteration: 100000 (without batches)

```
(59.0709087949697, 107.12600661621123)
```

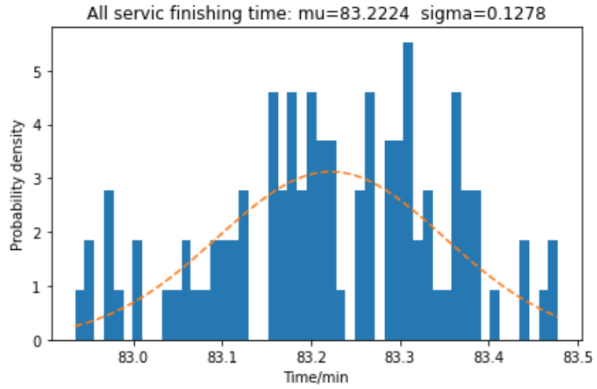


The orange line in the graph is the standard normal distribution which has the same mean and variable as the data collected. From the graph, we could see that it almost perfectly matches the line. So we think it is approximately normal. The 95% CI for 20 person in total is (59.0709, 107.1260), and we could get the 95% CI for time per person is (2.8535, 5.3563).

Question 5: simulation experiment and exact first two digits after the comma

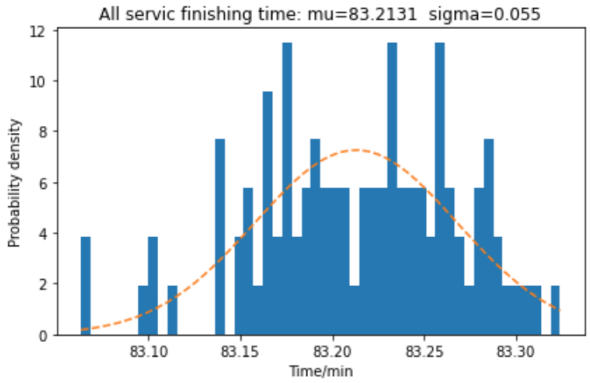
Iteration: 100 batches. 10000 samples in one batch.

```
(82.97068200496257, 83.47419839562467)
```



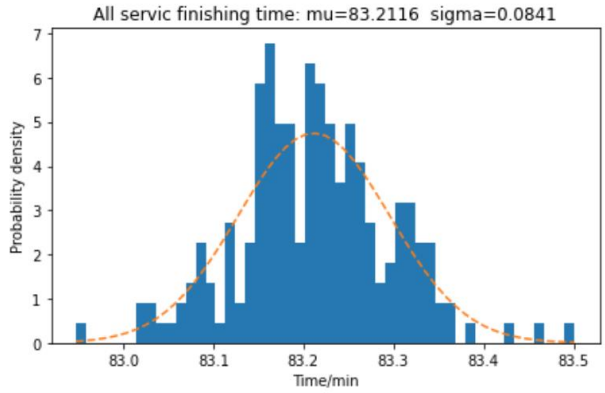
Iteration: 100*50000 (with batches)

```
(83.10478986266048, 83.32134928099629)
```



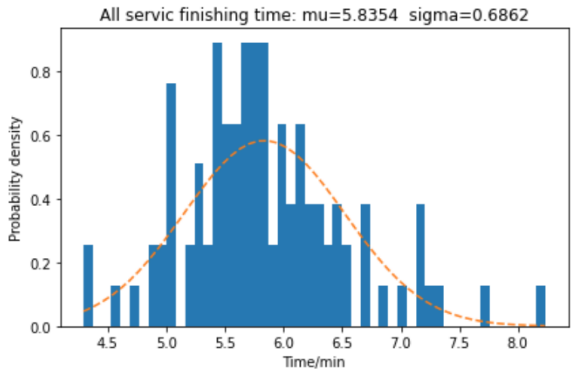
Iteration: 200*20000 (most opitmized in aspect of computing power and result accuracy)

```
(83.04623336276656, 83.37692063175457)
```



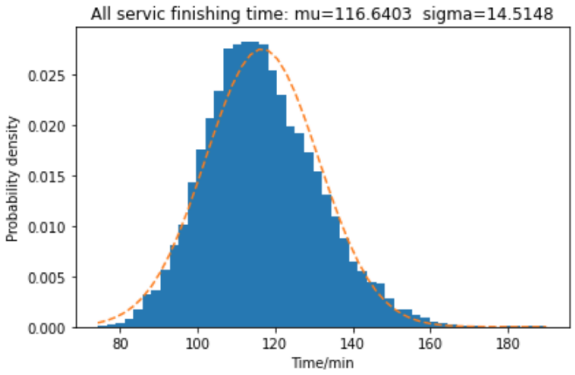
In this question, we apply the warmup to our algorithm.
Iteration: 100

```
(4.483621909868701, 7.187092009235645)
```



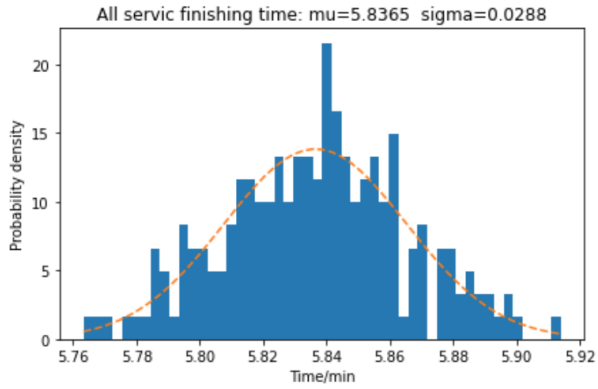
Iteration: 10000

```
(88.18981234802138, 145.09083589626974)
```



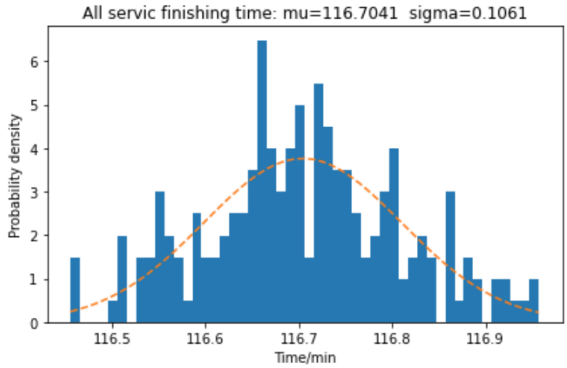
Iteration: 200*500

```
(5.779915020018829, 5.89313414712091)
```



Iteration: 200*20000

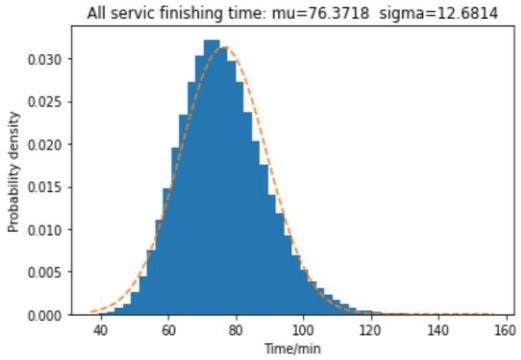
```
(116.49564395289848, 116.91252792883346)
```



Question 6: mutli-server stations

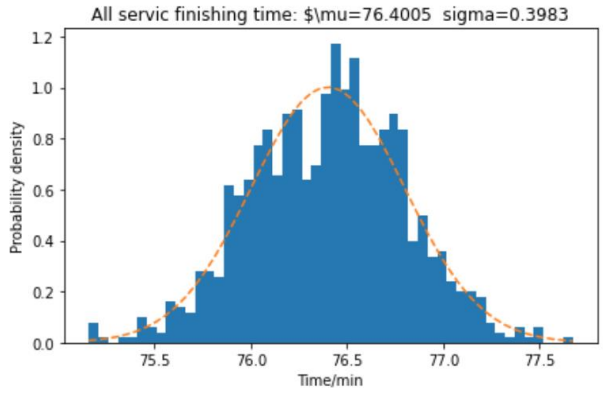
Iteration: 100000(without batches)

```
(51.516242148658066, 101.22741201159319)
```



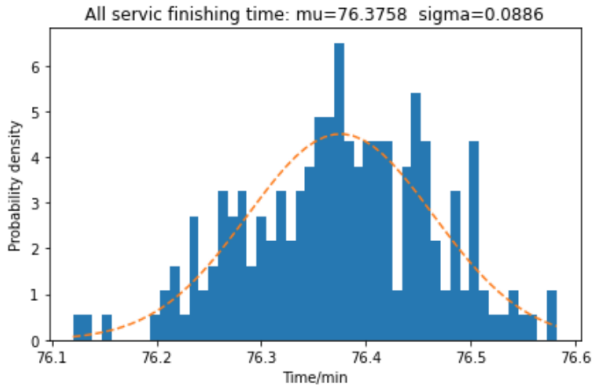
Iteration: 1000*1000

```
(75.61944649102337, 77.18148282972571)
```



Iteration: 200*20000

```
(76.20184612199239, 76.54985320779181)
```



Assignment 2:

Question 1: Why do we need the constraint lamda + 0:025?

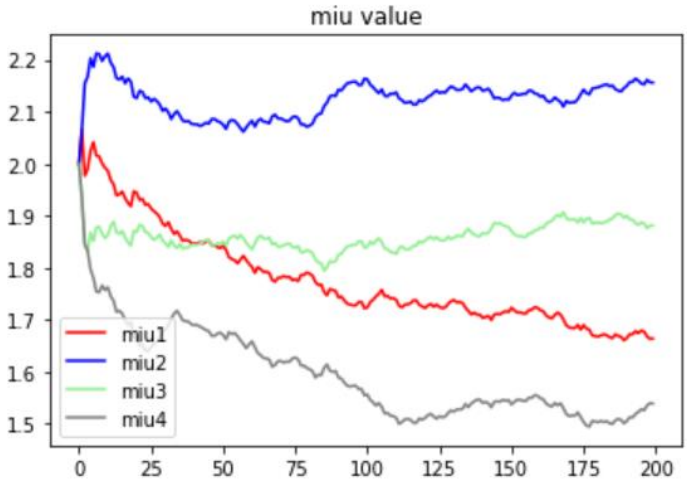
If lamda is too small, the mean value of exponential distribution will be quite large, which cannot happen in real life and running time for the simulation program will be quite long.

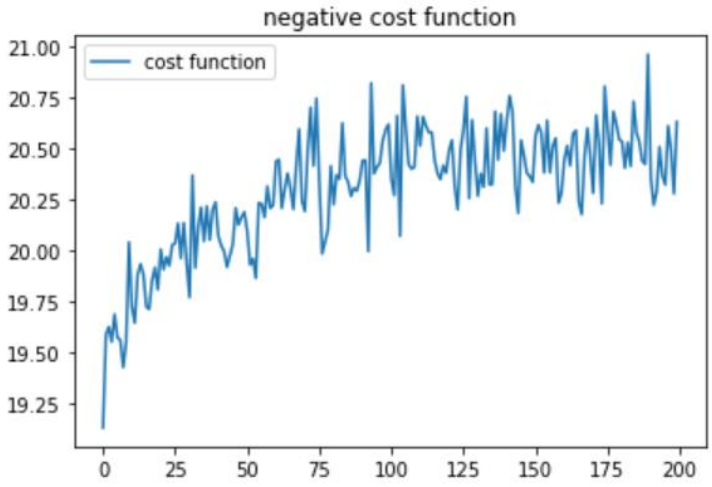
Question 2: the problem is well-posed.

By observing the behaviour of our algorithm. The opitimization problem has a solution and the solution is unique. The set of points confined to a bounded set. So the problem is well-posed.

Question 3: Apply SPSA for solving the optimization problem by simulation.

Suppose the expression we want to maximize to be J(μ), then the cost function of SPSA is -J(μ). For function $\tau(\mu_1, \ldots, \mu_I)$, we measure it by 60 divided by average process time per person (uses the same process as in Assignment 1 to simulate), which is the number of products the system can generate in one hour. We tried different set of parameters and here is the results.






```
Final values  theta & cost [1.66407313 2.15617943 1.88150158 1.53892947] -20.34355356071255
```

Question 4: how to terminate the optimization algorithm

The moving direction is right, and the final value of the mius and J(μ) seems to be stable in a small range. So we think it is right.