373 Assignment 1

Thomas Emo, 20239238

February 2024

Discussion

Here's a discussion for the single server queue simulation I made.

In 1 we see that the simulated and theoretical results are very closely matched, but are not perfect. Small deviations from the theoretical results are a result of the randomness in the system. Increasing the run time from 10**6 onwards will eliminate some of this randomness with the cost of runtime. Not also there is an asymptote as the arrival rate approaches the service rate, representing the queue becoming overloaded with users.

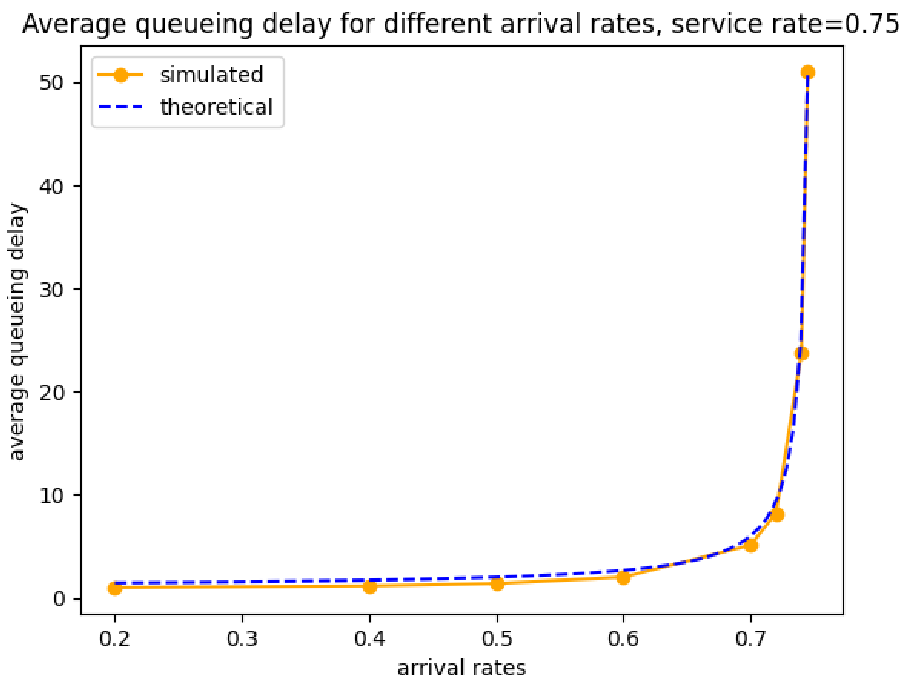

Figure 1: Average queuing delay vs arrival rate, with service rate parameter 0.75

To be brief, I will only include the code to run the server queue, the rest is just setting up plots, and collecting data.

To save time, not every time section of the server is simulated, instead we can just consider the arrival and departure times relative to the current time. Starting the clock at 0, generate the arrival and departure time, and depending on which occurs first, the clock will jump to that time, take the appropriate action by updating queue size (and account for edge cases), and then regenerate the respective arrival/departure time while leaving the other unchanged. This process is repeated until the clock is greater than the stop time (in this case 10**6).

Here is the code for the queue:

```
def single_server_queue(arrival_rate, service_rate, num_time_slots):
    clock = 0
    arrival_time = np.random.geometric(arrival_rate)
    departure_time = np.random.geometric(service_rate)
    queue_length = 0
    total_queue_length = 0

    while clock < num_time_slots:
        # arrivals
        if arrival_time < departure_time:
            clock = arrival_time
            arrival_time = clock + np.random.geometric(arrival_rate)
            if queue_length == 0:
                departure_time = clock + np.random.geometric(service_rate)
            queue_length += 1

        # departures
        elif departure_time < arrival_time:
            clock = departure_time
            if queue_length > 0:
                departure_time = clock + np.random.geometric(service_rate)
                queue_length -= 1
            else:
                departure_time = float('inf')

        else: # depart = arrival time, dont change len of q
            clock = departure_time
            arrival_time = clock + np.random.geometric(arrival_rate)
            departure_time = clock + np.random.geometric(service_rate)

        total_queue_length += queue_length

    average_queue_length = total_queue_length / num_time_slots
    return average_queue_length
}
```