**CST-305: Project 7 – Code Errors and the Butterfly Effect**

Jordan Scott and Angel Velazquez

Grand Canyon University

CST-305: **Principles of Modeling and Simulation Lecture & Lab**

Ricardo Citro

April 21, 2024

**CST-305: Project 7 – Code Errors and the Butterfly Effect**

**Responsibilities:**

We both worked on the development in the code for part 2 and both took part in working on the documentation.

**System Performance in Context:**

Queueing Theory is the mathematical study that delves into waiting lines, focusing on modeling and analyzing systems that cater to various demands. Such systems apply this theory by employing models that depict the physical setups, determining the quantity of servers, understanding the nature of demands, variability in arrival processes, and other relevant factors.

**Specific Problem Solved:**

A model will be developed to illustrate the dynamic nature of a system that exhibits the chaos and attributes of a self-organized file system. Through this, we can gauge the durability of the file system and predict when it will fall below the critical threshold.

**Mathematical Approach:**

The mathematical method utilized involved employing a range of formulas from Queueing theory, encompassing probabilities of packet loss, average gateway processing times, mean wait times, and job sizes, all relevant to their respective inquiries.

**Implementation in Code:**

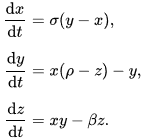
This code implements the Lorenz system, and the other codes a mathematical model describing a chaotic, deterministic dynamical system. We utilize NumPy for numerical computations and Matplotlib for plotting. We first define the `lorenz` function (system's dynamics through a set of differential equations) based on parameters “x”, “y”, “z”, “r”, “s”, and “b”. Next it does the calculation for “x\_dot”, “y\_dot” and “z\_dot” and returns all of those values. The simulation is conducted over a specified number of time steps, with initial values set for the state variables “x”, “y”, and “z”.

The user is prompted to input three different values of “r”, corresponding to low, medium, and high values. For each `r` value, the system's evolution is computed and plotted individually for the `x`, `y`, and `z` axes, as well as in 3D space to visualize the Lorenz attractor. This allows for the exploration of the system's behavior under various conditions and parameter values, providing insight into its chaotic dynamics.

For Part 2 question: we first define the arrival time, and service duration. Then we find the service start time, exit time , and time in queue. Then we initialize the customer in system and customer in queue. Next, we simulate the queue for the rest of customers. Now we calculate the Lq and LAq. Then we plot the five graphs, for the variables we initialize variables.

For Part 2 Question 3: We first define the constants for mu and lamda. Then we find range for k. Next we do the calculations for Utilization, ρ, Throughput, X, Mean number in the system, E[N] and Mean time in system, E[T]. Then we plot each one on a graph.

**Part 1:**



**Non-Chaotic:**

**A screenshot of a graph

Description automatically generatedA group of graphs showing different sizes and numbers

Description automatically generated with medium confidence**

Lorenz Attractor with r value of 5.   
  
Here we see that the system is not very chaotic. The 3d graph shows the spiral but, on the x(t), y(t), and z(t) as time moves forward the system is stable due to the flat line on the graphs.

**Semi-Chaotic:**

A graph of an attraction

Description automatically generated

**A group of graphs showing different types of data

Description automatically generated with medium confidence**

Lorenz Attractor with r value of 10.

The system got a little more chaotic, but it is still not fully chaotic. The 3d graph shows the line spiraling towards the center but the x(t), y(t), and z(t) graphs say something different. At the beginning the system jumps around but as time goes on the line goes flat, stabilizing.

**Chaotic:**

**A graph with a blue line

Description automatically generated**

**A group of blue lines

Description automatically generated**

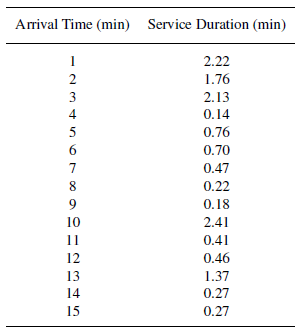
Lorenz Attractor with r value of 28.

Here the system is completely chaotic. The 3d graph shows two spirals that are merged. On the x(t), y(t), and z(t) graphs, the jumps are erratic and over time they do not stabilize and are

In summary, as the r value rises, the system transitions into chaos due to the spiral converging into another spiral. The graphs of x(t), y(t), and z(t) exhibit increased chaotic behavior with higher r values, as the system's response becomes increasingly erratic over time.

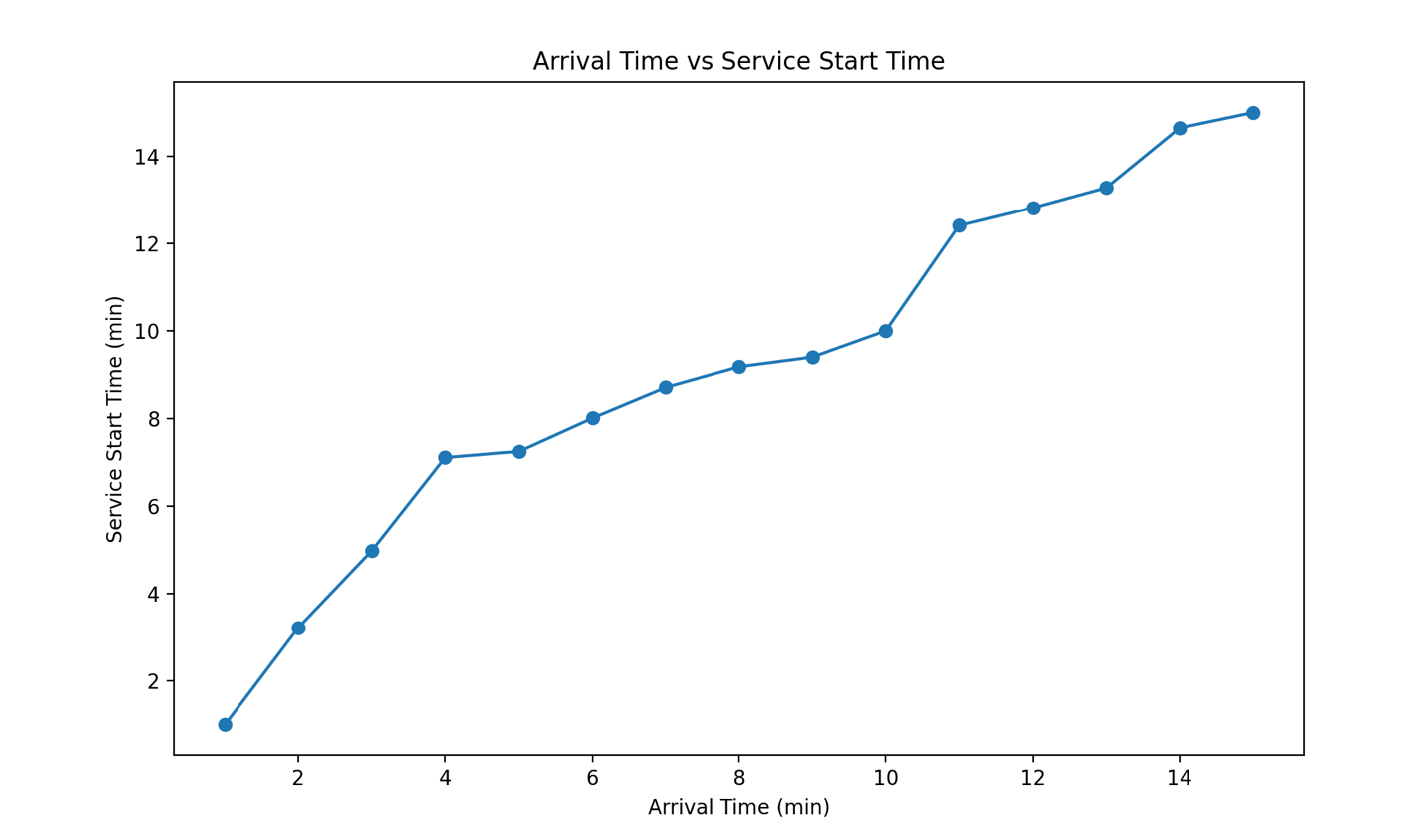
**Part 2:**

1. The Table below lists the arrival times and service durations for customers in a FCFS single server queue. From this data, compute (the time average number in queue) and (the average number in queue as seen by arriving customers). For , use a time horizon of , where is the time that the last customer exits the system. Assume the system is empty at t = 0. Calculate by hand for each inter-arrival time and write a Python code and generate 5 plots, that is, (1) the customer arrival time as a function of service start time, (2) the customer arrival time as a function of exit time, (3) the customer arrival time as a function of time in queue, (4) the customer arrival time as a function of the number of customers in system and (5) the customer arrival time as a function of number of customers in queue.



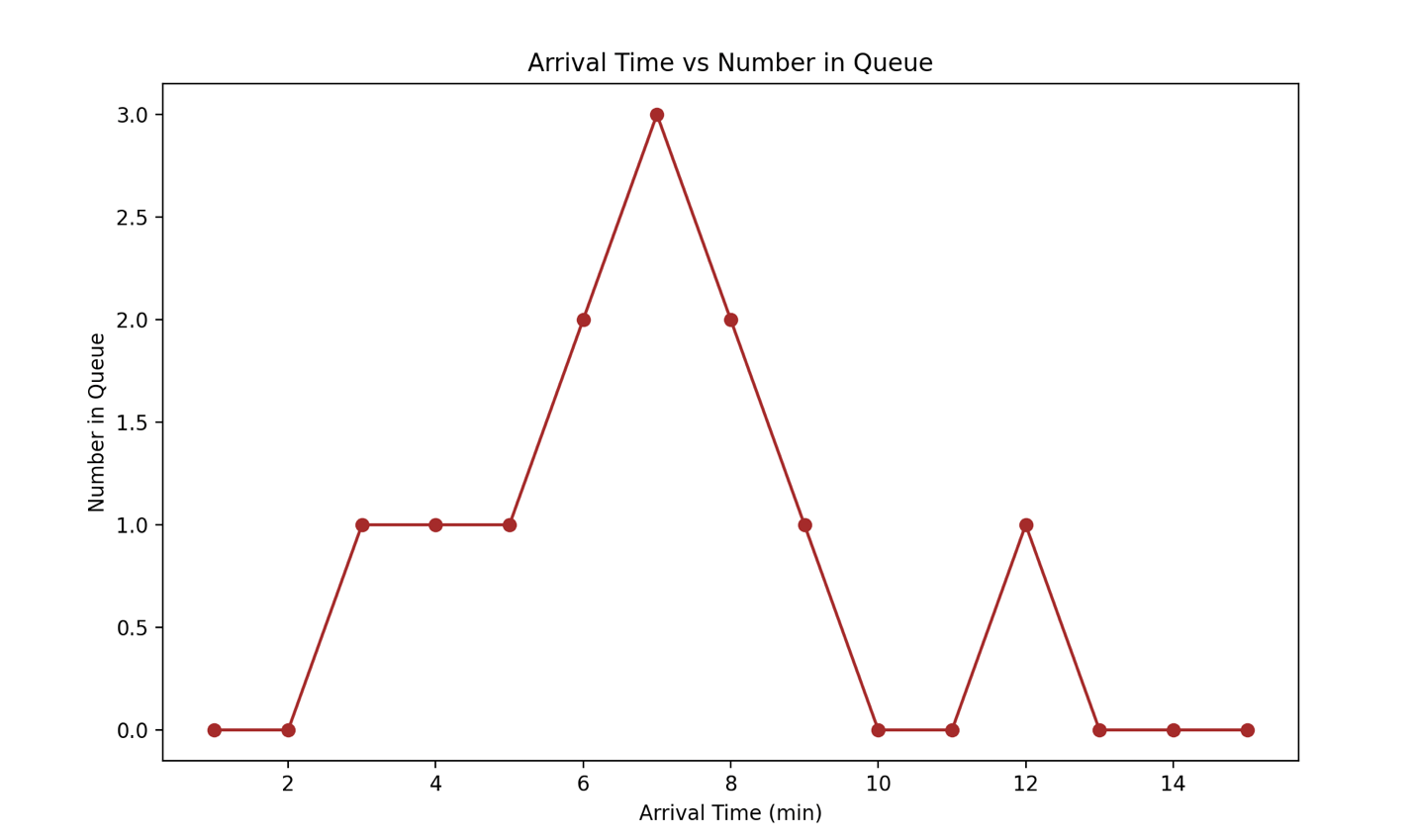
|  |  |  |
| --- | --- | --- |
| **Arrival Time (min)** | Service Duration | **Service Start Time (min)** |
| 1 | 2.22 | 1 |
| 2 | 1.76 | 3.22 |
| 3 | 2.13 | 4.98 |
| 4 | 0.14 | 7.11 |
| 5 | 0.76 | 7.25 |
| 6 | 0.7 | 8.01 |
| 7 | 0.47 | 8.71 |
| 8 | 0.22 | 9.18 |
| 9 | 0.18 | 9.4 |
| 10 | 2.41 | 10 |
| 11 | 0.41 | 12.41 |
| 12 | 0.46 | 12.82 |
| 13 | 1.37 | 13.28 |
| 14 | 0.27 | 14.65 |
| 15 | 0.27 | 15 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Exit Time (min)** | **Time in Queue (min)** | **Number in System** | **Number in Queue** |
| 3.22 | 0 | 0 | 0 |
| 4.98 | 1.22 | 1 | 0 |
| 7.11 | 1.98 | 2 | 1 |
| 7.25 | 3.11 | 2 | 1 |
| 8.01 | 2.25 | 2 | 1 |
| 8.71 | 2.01 | 3 | 2 |
| 9.18 | 1.71 | 4 | 3 |
| 9.4 | 1.18 | 3 | 2 |
| 9.58 | 0.4 | 2 | 1 |
| 12.41 | 0 | 0 | 0 |
| 12.82 | 1.41 | 1 | 0 |
| 13.28 | 0.82 | 2 | 1 |
| 14.65 | 0.28 | 1 | 0 |
| 14.92 | 0.65 | 1 | 0 |
| 15.27 | 0 | 0 | 0 |

A line graph with green dots

Description automatically generatedA graph with red dots

Description automatically generatedA graph with purple lines and numbers

Description automatically generated

Lq (Time average number in queue): 1.114603798297315

Lq(A) (Average number in queue as seen by arriving customers): 0.8

1. On a network gateway, measurements show that the packets arrive at a mean rate of 125 packets per second and the gateway takes about 2 milliseconds to forward them. Using an M/M/1 model, analyze the gateway. What is the probability of buffer overflow if the gateway had only 12 buffers? How many buffers do we need to keep packet loss below one packet per million?

λ (Arrival rate) = 125pps

µ (Service rate) = 1/.002 = 500pps

p (Gateway Utilization) = λ/ µ= 125/500 = 0.25

Probability of n packets in gateway =

Mean number of packets in gateway =

Mean time spent in gateway =

Probability of buffer overflow =

57 packets per billion packets

Limit less than 10^-6

n > log (10^-6) / log(0.25) = 9.96

So, if we round 9.96 to the nearest whole number it would be 10. So we will need 10 buffers.

1. Given an M/M/1 system (with λ < μ), suppose that we increase the arrival rate λ and the service rate μ by a factor of k each. How are the following affected?
   1. Utilization, ρ?
   2. Throughput, X?
   3. Mean number in the system, E[N]?
   4. Mean time in system, E[T]?

Show mathematically each a-d case. Using python, code and visualize your answer for each a-d.

1. p’ = So, the utilization 𝜌 remains unchanged.
2. Throughput in an M/M/1 system is the same as the arrival rate, assuming the system is stable. So if we multiply *λ* by 𝑘*k*, the new throughput *X*′ is:

X’=

1. The mean number of customers in the system 𝐸[𝑁]*E*[*N*] in an M/M/1 queue is given by:

𝐸[𝑁]=𝜌1−𝜌*E*[*N*]=1−*ρρ*​

Since 𝜌*ρ* remains unchanged, 𝐸[𝑁] also remains unchanged.

1. The mean time a customer spends in the system 𝐸[𝑇]*E*[*T*] is the inverse of the difference between the service rate and the arrival rate:

𝐸[𝑇]=1/𝜇−𝜆

If both 𝜆 and 𝜇 are increased by a factor of 𝑘, then:

𝐸[𝑇]′= = ​*E*[*T*]

So, the mean time in the system is inversely proportional to *k*.

A graph of a function

Description automatically generated with medium confidence

1. Given an M/M/1 server, what is the maximum allowable arrival rate of jobs if the mean job size (service demand) is 3 minutes and the mean waiting time (E[TQ]) must be kept under 6 minutes?

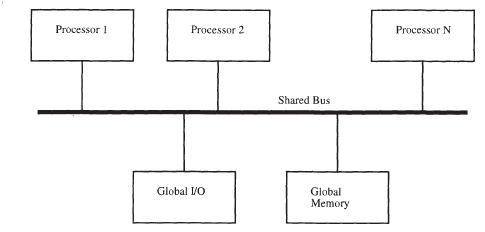
Mean job size = 3 minutes

Mean waiting time = 6 minutes

E(w) =

So that mean job size =

1. A tightly coupled multiprocessor (TCMP) is defined as a distributed computer system where all the processors communicate through a single (global) shared memory. A typical physical layout for such a computer structure is shown below. Draw a queueing model that illustrates this single bus tightly coupled multiprocessor (SBTCMP) architecture. Each processing element, PE (with identifying index ), is modeled as a finite set of tasks (the task pool), a CPU, a bus interface unit (BIU) that allows the PE to access the shared bus, and a set of queues associated with the CPU and BIU. Each CPU and BIU has mean service rate and , respectively. Tasks are assumed to have a mean sleep time in the task pool of time units. We also assume the CPU and BIU operate independently of each other, and that all the BIU's in the multiprocessor can be lumped together into a single "equivalent BIU". The branching probabilities are . The branching probability is interpreted as the probability that a task associated with PE will join the CPU queue at PE after using the BIU; is then the probability that a task associated with PE will wait for an intenupt acknowledgment at PE . It is not the probability that a task will migrate from one PE to another. Interrupts to the PE are assumed to occur at a mean rate . In general, the interrupt rate to PE will be a function of task location, inter task communication probability, speeds of resources on other PE's, global I/O devices, etc. We will assume that the workload is fixed.



A diagram of a computer program

Description automatically generated

**Flowcharts:**

**Part 2 Question 1’s code flowchart:**

A screenshot of a computer

Description automatically generated

**Part 2 Question 3’s code flowchart:**

A screenshot of a computer

Description automatically generated

GitHub link: